Linear Programming Decoding for Non-Uniform Sources and for Binary Channels With Memory

by

Adam Cohen

A thesis submitted to the

Department of Mathematics and Statistics

in conformity with the requirements for

the degree of Master of Science (Engineering)

Queen's University

Kingston, Ontario, Canada

December 2008

Abstract

Linear programming (LP) decoding of low-density parity-check codes was introduced by Feldman *et al.* in [1]. In his formulation it is assumed that communication takes place over a memoryless channel and that the source is uniform. Here, we extend the LP decoding paradigm by studying its application to scenarios with source non-uniformity and to decoding over channels with memory. We develop two decoders for the scenario of non-uniform memoryless sources transmitted over memoryless channels. The first decoder uses a modified linear cost function which incorporates the *a-priori* source information and works with systematic codes. The second decoder differs by using non-systematic codes obtained by puncturing lower rate systematic codes and using an extended decoding polytope. Simulations show that the modified decoders yield gains over the standard LP decoder. Next, LP decoding is considered for two channels with memory: the binary additive Markov noise channel and the infinite-memory non-ergodic Polya-contagion channel. For the Markov channel, no linear cost function corresponding to maximum likelihood (ML) decoding could be obtained and hence it is unclear how to proceed. For the Polya channel, two LP-based decoders are developed. The first is derived in a straightforward manner from the ML decoding rule of [2]. The second decoder relies on a simplification of the same ML decoding rule which holds for codes containing the all-ones codeword. Simulations are performed for both decoders with regular and irregular LDPC codes and demonstrate relatively good performance with respect to the channel ϵ-capacity.

Acknowledgments

First and foremost, I would like to thank my supervisors, Dr. Fady Alajaji, Dr. Navin Kashyap and Dr. Glen Takahara. Their teaching, supervision, and direct contributions to this work have been invaluable. They have invested a great deal of time and energy into my work and development, and for this I am very grateful.

I would also like to thank my friends and colleagues in the math library, Pantelis, Mark, Daniel and Amy, for many interesting discussions and distractions. In addition, I would like to thank my friends and collaborators in music, Julian, Eric and Mike (and more recently in business, Julian and Eric) for providing a fulfilling creative outlet during my time here.

I would like to thank my family and my girlfriend Daniela for all that they have given me in life. Finally, I would like to dedicate this work to my mother.

Contents

Abstract **ii**

Acknowledgments **iii**

List of Tables **vii**

List of Figures **viii**

Chapter 1: Introduction **1**

1.1 Description of the Problem 1

1.2 Review of Literature 4

1.3 Thesis Overview 7

1.4 Thesis Contribution 8

Chapter 2: Channel Coding and Modelling **10**

2.1 Channel Coding 10

2.1.1 Binary Linear Codes 12

2.1.2 Low-Density Parity-Check Codes 15

2.2 Channel Modelling 16

2.2.1 Channel Decoding 18

2.2.2 Binary Symmetric Channel 19

2.2.3 Additive White Gaussian Noise Channel 20

2.2.4 First-Order Markov Noise Channel 22

2.2.5 Non-ergodic Polya Channel 23

Chapter 3: Linear Programming Decoding **29**

3.1 Linear Programming 29

3.1.1 Definition 29

3.1.2 Solving Linear Programs 31

3.2 Decoding as an LP 32

3.2.1 Linear Cost Function 33

3.2.2 Region of Optimization 35

3.2.3 Relaxing the Problem 36

3.2.4 Complexity 38

3.2.5 Adaptive LP Decoding 41

Chapter 4: LP Decoding for Non-Uniform Sources **43**

4.1 LP Decoding for Non-Uniform Sources with Systematic Codes 44

4.2 LP Decoding for Non-Uniform Sources with Non-Systematic Codes . 47

4.3 The AWGN Channel 49

4.4 Simulation Results 51

Chapter 5: LP Decoding for Channels with Memory **59**

5.1 First-Order Markov Noise Channel 60

5.2 Polya Channel 67

5.3 Simplified ML Decoding for the Polya Channel 69

5.4 Simulation Results 72

Chapter 6: Conclusions and Future Work **80**

Bibliography **83**

Appendix A: Implementation Details **90**

Appendix B: Iterative Decoders for the Polya Channel **94**

List of Tables

5.1 Relaxed MDD performance for Polya channel 74

5.2 Relaxed MMDD performance for Polya channel 75

List of Figures

1.1 Typical model for a communication system. 2

2.1 Visual representation of the binary symmetric channel (BSC). 20

2.2 The additive white Gaussian noise (AWGN) channel. 21

2.3 State transition diagram for the first-order Markov noise channel. . . 24

4.1 Standard LP Decoder vs. Modified LP Decoders for Non-Uniform Source $n = 200$, $p_1 = 0.9$, $R = \frac{1}{2}$ over the BSC. 54

4.2 Standard LP Decoder vs. Modified LP Decoders for Non-Uniform Source $n = 200$, $p_1 = 0.8$, $R = \frac{1}{2}$ over the BSC. 55

4.3 Standard LP Decoder vs. Modified LP Decoders for Non-Uniform Source $n = 200$, $p_1 = 0.7$, $R = \frac{1}{2}$ over the BSC. 56

4.4 Systematic $(200, 100)$ LDPC code vs. Systematic $(1000, 500)$ LDPC code at $R = \frac{1}{2}$, $p_1 = 0.7$ over the BSC. 57

4.5 LP JSC decoders for the BSC and AWGN channel with systematic $(200, 100)$ LDPC code and $p_1 = 0.8$ 58

5.1 $\delta = 10$: $(200, 100)$ $(3, 6)$-regular LDPC code under relaxed MDD decoding and irregular $(200, 100)$ LDPC code under relaxed MMDD. Curves representing the ϵ-capacity and the case of ideal interleaving (BSC) are also included for comparison. 76

5.2 $\delta = 4$: $(200, 100)$ $(3, 6)$-regular LDPC code under relaxed MDD decoding and irregular $(200, 100)$ LDPC code under relaxed MMDD. Curves representing the ϵ-capacity and the case of ideal interleaving (BSC) are also included for comparison. 77

5.3 $\delta = 2$: $(200, 100)$ $(3, 6)$-regular LDPC code under relaxed MDD decoding and irregular $(200, 100)$ LDPC code under relaxed MMDD. Curves representing the ϵ-capacity and the case of ideal interleaving (BSC) are also included for comparison. 78

5.4 $\delta = 10$: $(200, 100)$ $(3, 6)$-regular LDPC code and irregular $(200, 100)$ LDPC code under relaxed MDD and irregular $(200, 100)$ LDPC code under relaxed MMDD. Curve representing ϵ-capacity is also included. 79

1 Word error rate of a Rate-$\frac{1}{2}$ $(3, 6)$ LDPC code of length 200 over a range of BSC crossover probabilities, corresponding to [1, Fig. 7]. . . 93

2 Word error rate of a Rate-$\frac{1}{2}$ $(3, 6)$ LDPC code of length 200 over the non-ergodic Polya channel with $\delta = 2$ using both a message-passing decoder and relaxed MDD over a range of values of the Polya channel BER ρ. 95

Chapter 1

Introduction

1.1 Description of the Problem

Shannon's seminal work [3], "A Mathematical Theory of Communication," laid the foundation for information and communication theory. In his paper, Shannon defined a novel method for measuring information, and proved a fundamental result about the ability to reliably transmit that information over noisy channels.

We begin by clarifying what is meant by "communication over a noisy channel." A typical model for a communication system is shown in Figure 1.1. In this model, the source attempts to communicate some message to the destination, and must do so through a channel which has the potential to introduce corruption. The source and channel encoders and decoders depicted in Figure 1.1 compose the pre- and post-transmission processing components of the system. Typically, the source encoder and

decoder are responsible for data compression and decompression, that is, the process of representing the message as efficiently as possible, and then restoring it to its original form. The channel encoder, on the other hand, introduces structured redundancy into the message and the channel decoder attempts to use that redundancy to detect and correct errors introduced by the channel. This work focusses primarily on the channel coding component of the system, and even more specifically, on the channel decoder. For a comprehensive and rigorous introduction to information and communication theory, the reader is directed to [4] or [5].

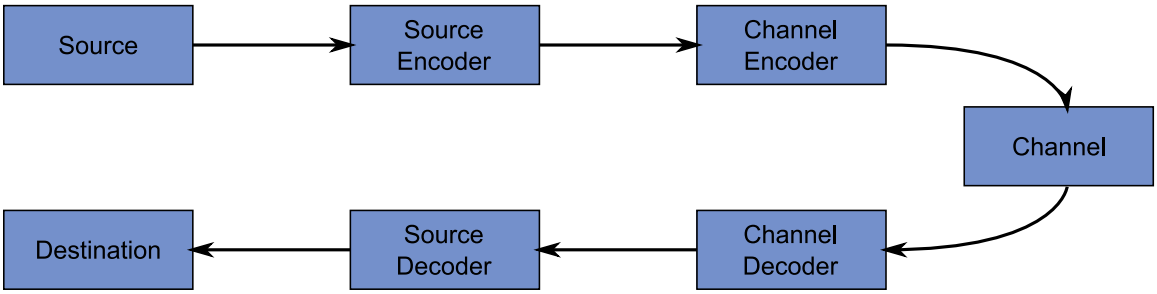

Figure 1.1: Typical model for a communication system.

Although Shannon's work demonstrated the existence and achievability of fundamental performance limits associated with transmitting information, it did not show how these limits could be reached using practical systems. Thus, the development of effective channel codes used for communication over noisy channels has been an integral focus in information and communication theory since Shannon's work in 1948. From an application standpoint, one of the most notable classes of channel codes to date is the class of low-density parity-check (LDPC) codes. Although

originally proposed by Gallager in 1960 [6], LDPC codes, along with powerful and efficient sub-optimal iterative decoders, were not popularized until 1999, when they were rediscovered by MacKay [7]. The excellent (near Shannon limit) performance and practicality of LDPC codes quickly made them a primary focus in the field of coding theory [8].

Very recently, Feldman [9] demonstrated that the problem of decoding LDPC codes could be expressed as a linear program. This decoding approach, termed LP decoding, yields similar error performance as iterative decoders and, in addition, is an analytically tractable decoding technique. Feldman's LP decoders were designed to decode for uniform sources transmitting over discrete input memoryless symmetric channels. Although these source and channel assumptions are common in the literature, they do not always apply well to real communication systems. The source uniformity assumption is often inaccurate, as source data is frequently uncompressed, and, even when it is compressed, it is rarely compressed ideally. Further, channel memory is most commonly addressed in practice by interleaving, a strategy which strives to make the channel "appear" memoryless to the decoder. Although useful in making the channel better-fit the memoryless assumption, this approach is not optimal as memory has a positive effect on a channel's ability to reliably carry information. Thus, if accounted for correctly at the decoder, channel memory can serve to improve performance. In light of these issues, there has been a substantial amount of research into the modification of iterative decoders of LDPC codes for decoding

in situations with source non-uniformity (see *e.g.*, [10],[11]) and with channel with memory (see *e.g.*, [12],[13],[14]). Thus, it is natural to investigate the modification of LP decoders to the same end, and this is the problem which is considered herein.

1.2 Review of Literature

LP decoding of binary linear codes (BLCs), and in particular, LDPC codes, was introduced by Feldman *et al.* in his 2005 paper "Using linear programming to decode binary linear codes" [1]. The paper was based on work from Feldman's PhD thesis, published in 2003 [9], and demonstrates how the problem of decoding BLCs transmitted over discrete input memoryless symmetric channels can be formulated as an LP. The proposed LP decoder is sub-optimal[1] and has polynomial complexity with respect to the code-length. For LDPC codes, its performance is tightly related to that of the iterative min-sum decoder, a fact that was noted by Feldman and has been studied by Vontobel and Koetter in [15]. Another important characteristic of the proposed LP decoder is the so-called "ML certificate property," which provides a guarantee on the optimality of the LP decoding result. Prior to the work regarding BLCs, and also based on work from Feldman's thesis, Feldman *et al.* [16] demonstrated how LP decoding could be applied to (turbo-like) repeat accumulator codes, with bounded error performance. In [17], Kashyap proves that for a certain nontrivial class of binary linear codes, relaxed LP decoding is in fact equivalent to ML decoding,

[1]If it is not stated explicitly when discussing decoder optimality, we are referring to optimality in the sense of minimizing the probability of *codeword* error.

thus providing a polynomial-time algorithm for the ML decoding of those codes. LP decoding has also been studied in the context of non-binary linear codes [18].

Since Feldman's initial publications, a number of developments have occurred in the area. One of the most interesting features of LP decoding is that, in direct contrast to iterative decoding techniques, it is relatively well-suited to mathematical analysis. As a result, there exist some important theorems on the properties of LP decoding. The first such result, due to Feldman *et. al.*, is regarding the error correction capability of LP decoding [19]. Specifically, it is shown that for an LDPC code satisfying certain conditions on its Tanner graph and code rate, LP decoding can correct a number of errors that is proportional to the length of the code. We note that the conditions required for the theorem to apply are satisfied with high probability by a random LDPC code. Some additional results and analysis regarding the error performance of LP decoding have been published in [20] and [21]. Another important analytical result regarding LP decoding, due to Feldman and Stein, is that LP decoding achieves capacity [22]. It is shown that when applied to expander codes (a class of codes constructed from expander graphs), LP decoding achieves the capacity of memoryless symmetric log-likelihood bounded channels.

The main draw-back of LP decoding is that it is much less efficient than iterative decoding and so is not presently considered to be a practical decoding technique. However, important progress has been made in this area. In [23], Taghavi and Siegel demonstrate how major increases in the efficiency of LP decoding can be obtained

by using adaptive LP techniques. The idea is to solve a series of less complex LPs, adding constraints along the way, until the desired LP is solved. In the paper, they show substantial experimental efficiency gains, especially in the case of codes with more dense parity-check matrices. In addition, they show how the adaptive LP technique can be extended so as to improve decoding performance. Another interesting publication "Toward low complexity LP decoding", [24], explores the development of algorithms designed specifically for LP decoding, as opposed to using standard LP solvers. In this paper, Vontobel and Koetter show that algorithms which have complexity similar to that of the min-sum algorithm are feasible for solving the decoding LP, a very promising result. In a related work by Burshtein [25], an approximate LP decoder is developed which is proved to correct the constant fraction of errors promised in [19] with complexity that is *linear* in the length of the code.

There has also been research which demonstrates how the error performance of LP decoding can be improved. Draper *et al.* extend the above-mentioned approach of Taghavi and Siegel even further, and modify the adaptive LP decoder to incorporate integer constraints [26]. By allowing for integer constraints, they are able to implement an ML decoder which executes in a reasonable amount of time for codes of moderate block-length. Another improvement to the performance of LP decoding was proposed by Dimakis and Wainwright [27] who develop a decoding algorithm which provably outperforms the standard LP decoder for expander codes at the expense of a modest increase in complexity. Again, similar to parts of the work of Taghavi

and Siegel, enhanced performance is achieved by adding additional constraints to the original LP and, in this case, those extra constraints are selected by guessing in a certain way.

There is also work which explores the more general question of decoding via convex optimization, a superset of linear programming. In [28], the authors demonstrate that ML decoding of LDPC codes transmitted over linear vector channels can be relaxed to a convex optimization problem, a result which is relevant to this work as this class of channels has memory. A list of publications relating to LP decoding including those mentioned herein and some others is found on the website of Vontobel [29].

1.3 Thesis Overview

In Chapter 2 we will provide an introduction to coding theory. Included are an introduction to BLCs and LDPC codes, and a description of the communication channels considered in this work. Specifically, we will discuss the binary symmetric channel (BSC), the additive white Gaussian noise (AWGN) channel, the first order additive Markov noise channel, and the infinite-memory non-ergodic Polya contagion channel.

Next, Chapter 3 provides the background required to understand and extend Feldman's work on LP decoding of LDPC codes. We will discuss linear programming, linear cost functions for decoding, decoding polytopes and adaptive LP decoding.

With the required background, in Chapter 4 we explore the application of LP decoding to the scenario of decoding LDPC-coded non-uniform sources sent over the BSC and AWGN channels. A linear cost function is derived which exploits source non-uniformity, and two decoders are developed based on this cost function. One decoder is designed for systematic codes and one uses punctured systematic codes. Simulation results are provided to demonstrate the decoders' performance.

In Chapter 5, we investigate the application of LP decoding to channels with memory; specifically, the Markov noise channel and the infinite-memory non-ergodic Polya contagion channel. For the Markov noise channel, a quadratic cost function is derived, but no decoder is developed. For the Polya channel, two separate decoders are developed and simulation results are included to demonstrate their performance.

Finally, in Chapter 6 we will summarize our results and suggest future avenues for work in this area.

1.4 Thesis Contribution

The following is a list of the contributions presented in this work:

- Developed an LP decoder for non-uniform sources and systematic codes for the BSC and AWGN channel. Simulation results for these decoders are provided.

- Developed an LP decoder for non-uniform sources and non-systematic codes based on puncturing and the use of an "extended codeword polytope" for the

BSC. Simulation results for the decoder are provided.

- Derived a quadratic cost function corresponding to maximum likelihood (ML) decoding for the Markov noise channel.
- Proved the equivalence of ML decoding to minimum or maximum Hamming distance decoding (depending on a channel parameter) for the non-ergodic Polya channel when using codes with the all-ones codeword.
- Developed a relaxed LP decoder for the Polya channel.
- First simulation results for the Polya channel with good performance with respect to the channel ϵ-capacity.

The results mentioned above have been published in part in [30].

Chapter 2

Channel Coding and Modelling

2.1 Channel Coding

Channel coding is the process of adding redundancy to a message with the aim of protecting that message against error. Before diving into definitions, we will provide a very simple example of channel coding to demonstrate the concept.

Suppose that a source would like to transmit a message consisting of the binary symbol "1" across a channel which flips the symbol to a "0" with probability $p = 0.1$. If the message is sent as-is, then clearly it will be received correctly with probability $1 - p = 0.9$. Now, suppose that we repeat the symbol three times, that is, transmit

"111" instead. Let us analyze the possible error sequences:

$$\begin{aligned} P[\text{0 errors}] = (1-p)^3 &= 0.729, \\ P[\text{1 errors}] = 3(1-p)^2 p &= 0.243, \\ P[\text{2 errors}] = 3(1-p)p^2 &= 0.027, \text{ and} \\ P[\text{3 errors}] = p^3 &= 0.001. \end{aligned}$$

Now, suppose that in order to decode this message we use a majority decoding rule, that is, we decode to 1 if there are more 1's than 0's in the received bit sequence, and vice versa. By this rule, we will correctly decode the received bits with probability $0.729 + 0.243 = 0.972$, *i.e.*, the probability that one or fewer errors occurred. If we were to have repeated the symbol five times instead of three, used a majority rule and left $p = 0.1$, we would increase the probability of successful decoding to 0.99144.

The example above demonstrates the classical trade-off between redundancy and probability of error. This trade-off characterizes channel coding: we were able to significantly reduce the probability of error at the cost of transmitting information at a lower rate. Channel coding techniques are primarily focused on methods of adding redundancy in a highly structured manner so as to obtain the maximum benefit in terms of error protection. The content contained herein is standard in the literature, and for a more thorough treatment of the subject the reader is directed to [31] or [32].

2.1.1 Binary Linear Codes

An (n, k), $n > k$, BLC C is a k-dimensional subspace $\mathbb{F}_2^n$. Any element $\underline{c} \in C$ is called a *codeword*. We define the rate R of an (n, k) BLC to be $R = \frac{k}{n}$. An encoder for C is a 1-to-1 and onto map, f, of the form

$$f : \mathbb{F}_2^k \rightarrow C.$$

In other words, the encoder maps each binary k-tuple to some binary n-tuple in C. Redundancy is added here as $n > k$, and the rate of the code R represents the ratio of the number of bits going into the encoder to that coming out.

The *dual* of a BLC C is denoted $C^{\perp}$ and is defined as

$$C^{\perp} = \{\underline{h} \in \mathbb{F}_2^n : \underline{h} \cdot \underline{c} = 0, \ \forall \ \underline{c} \in C\},$$

where the dot product is performed over $\mathbb{F}_2$.

An (n, k) BLC can be described by a set of k basis vectors in $\mathbb{F}_2^n$. A *generator matrix* of an (n, k) BLC is a $k \times n$ matrix over $\mathbb{F}_2$ with its rows forming a basis of the code. A *parity-check matrix* of an (n, k) BLC is an $(n - k) \times n$ matrix over $\mathbb{F}_2$ with its rows forming a basis of $C^{\perp}$. We note that a parity-check matrix for C is a generator matrix of $C^{\perp}$, and vice versa.

A BLC can be completely specified using a parity-check matrix or a generator

matrix. Given a parity-check matrix H for an (n, k) code C, we have that

$$C = \{\underline{c} \in \mathbb{F}_2^n : H\underline{c}^t = \underline{0}\},$$

where $\underline{0}$ is the all-zeros vector in $\mathbb{F}_2^k$ and addition is performed modulo-2. Similarly, given a generator matrix G for an (n, k) code C, we have

$$C = \{\underline{c} \in \mathbb{F}_2^n : \underline{s}G = \underline{c} \text{ for some } \underline{s} \in \mathbb{F}_2^k\},$$

where, again, addition is performed modulo-2. A generator matrix also serves as an encoder for a BLC, as it maps vectors from $\mathbb{F}_2^k$ into the appropriate subspace of $\mathbb{F}_2^n$. It is worth noting that for a particular code, there are many distinct generator and parity-check matrices.

We now provide an example of a linear block code. The $(7, 4)$ *Hamming code*, denoted $\mathbb{H}_3$, is a BLC with parity-check matrix

$$H = \begin{bmatrix} 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 \end{bmatrix},$$

and generator matrix

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

A generator matrix is said to be *systematic* if it contains the $k \times k$ identity matrix as a submatrix, as in the example above. Strictly speaking, this is a property of generator matrices; however, it is often the case that a *code* will be described as systematic. In this case, it is tacitly assumed that the code is being discussed in tandem with a corresponding systematic generator matrix which is being used as the encoder.

Given a parity-check matrix for some BLC C, it is always possible to obtain a corresponding systematic generator matrix for C. First, the parity-check matrix must be row-reduced and have its columns permuted so as to take following block form

$$H = [A|I_{n-k}]\,,$$

where A is an $(n-k) \times k$ matrix, and I_{n-k} is the $(n-k) \times (n-k)$ identity matrix. Given such a parity-check matrix,

$$G = \left[I_k|A^t\right]$$

is always a systematic generator matrix for C, where I_k is the $k \times k$ identity matrix, and A^t is the transpose of A.

2.1.2 Low-Density Parity-Check Codes

LDPC codes were originally proposed by Gallager in his PhD thesis in 1960 [6]; however, the codes were mostly forgotten until 1999 when they were revisited by MacKay [7]. From this point forth, the codes have been studied heavily due to their excellent (capacity-approaching) performance achieved using practical iterative decoders [8].

LDPC codes are a class of BLCs which are represented by *sparse* parity-check matrices. Roughly speaking, if we let d_r^* and d_c^* be the maximum number of 1's in any given row and column of some parity-check matrix H, respectively, then we say that H is sparse if d_r^* and d_c^* are small constants relative to the number of columns and rows in H, respectively. For example, in practice, a rate-$\frac{1}{2}$ LDPC code might have a 30000×60000 parity-check matrix with maximum row and column degrees of 6 and 3, respectively.

A BLC is said to be regular if it is derived from a parity-check matrix in which every row and column has d_r^* and d_c^* 1's, respectively. A length-n, dimension k regular BLC with row degree d_r^* and column degree d_c^* will be referred to as an (n, k) (d_c^*, d_r^*)-regular code. A BLC is said to be irregular if it is not regular.

Below we give an example of a parity-check matrix corresponding to a $(20, 10)$ $(3, 6)$-regular code

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

Generally speaking, an LDPC code is designed by creating a sparse parity-check matrix with desired row- and column-degree properties. From the parity-check matrix, a systematic generator matrix can always be obtained in the manner described in 2.1.1. Both the regular and irregular LDPC codes used in the simulations of this work were generated using the software of Neal [33], as described in Appendix A.

2.2 Channel Modelling

In the physical world, if one wishes to communicate a message of any form, that message must travel through space and/or time before arriving at the intended destination (*e.g.*, one's voice through the air, or bits stored on a cd-rom). The spatial/temporal steps between transmitter and receiver, which have the potential to introduce corruption, are referred to as a communication channel. A channel is formally defined as a

triple $(\mathcal{X}, P[\cdot|\cdot], \mathcal{Y})$, where $\mathcal{X}$ and $\mathcal{Y}$ are the sets of possible channel input and output symbols, respectively, and $P[\cdot|\cdot]$ describes the channel transition probabilities. More specifically, given $(x_1, \ldots, x_n) \in \mathcal{X}^n$ and $(y_1, \ldots, y_n) \in \mathcal{Y}^n$, $P[(y_1, \ldots, y_n)|(x_1, \ldots, x_n)]$ gives the probability (or likelihood, if $\mathcal{Y}$ is continuous) that $(y_1, \ldots, y_n)$ will be received given that $(x_1, \ldots, x_n)$ was transmitted, for any $n \geq 1$. A channel is said to be *memoryless* if its transition probabilities are independent from symbol to symbol, that is, if

$$P[(y_1, \ldots, y_n)|(x_1, \ldots, x_n)] = \prod_{i=1}^{n} P[y_i|x_i].$$

One of the most important properties of a channel, although not central to this work, is its *information capacity*. The information capacity of a channel was shown by Shannon [3] to represent the maximum rate at which, given sufficiently large block length, information can be transmitted over the channel with arbitrarily small probability of error. The existence of this quantity, having such a strong operational significance, is one of the most important concepts in information and communication theory.

Next, we give a brief introduction to channel decoding. The remainder of the section will be devoted to describing the four specific communication channels which will be considered in this work: the binary symmetric channel, the additive white Gaussian noise channel, the first order additive Markov noise channel, and the infinite-memory non-ergodic Polya contagion channel.

2.2.1 Channel Decoding

Given a binary channel $(\mathbb{F}_2, P[\cdot|\cdot], \mathcal{Y})$[1] and an (n, k) BLC C, a complete[2] channel decoder for C is a map of the form

$$g : \mathcal{Y}^n \to C.$$

Suppose that a codeword $\underline{c} \in C$ is transmitted through the channel and $\underline{y} \in \mathcal{Y}^n$ is received at the channel output. It is the job of the channel decoder to determine, as best possible, which codeword of C was transmitted given that $\underline{y}$ was received. Generally, the optimal decoding rule, in terms of minimizing the probability of codeword error (PCE), is the maximum *a-posteriori* probability (MAP) decoding rule. Given a received word $\underline{y}$, the rule is to select as the decoded vector some $\hat{\underline{y}} \in C$ which maximizes

$$P[\hat{\underline{y}}]P[\underline{y}|\hat{\underline{y}}],$$

where $P[\hat{\underline{y}}]$ represents the *a-priori* probability that $\hat{\underline{y}}$ was transmitted and $P[\underline{y}|\hat{\underline{y}}]$ represents the probability (or likelihood if $\mathcal{Y}$ is continuous) that $\underline{y}$ was received given that $\hat{\underline{y}}$ was transmitted.

[1] We restrict ourselves to the consideration of binary codes and binary input channels to simplify the presentation. Further, throughout this work the channel output alphabet $\mathcal{Y}$ will either be $\mathbb{R}$ (for the AWGN channel) or $\mathbb{F}_2$ (for all other channels considered here).

[2] "Complete" here specifies that the decoder always outputs a codeword. We note that many practical decoders are not complete, and, in particular, relaxed LP decoders are not complete since they can output pseudocodewords.

If every codeword is equiprobable (*i.e.*, the source is uniform) then the MAP decoding rule reduces to the maximum likelihood (ML) decoding rule. Given a received word $\underline{y}$, the ML decoding rule is to select as the decoded vector some $\hat{\underline{y}} \in C$ which maximizes

$$P[\underline{y}|\hat{\underline{y}}],$$

where, again, $P[\underline{y}|\hat{\underline{y}}]$ represents the probability (or likelihood if $\mathcal{Y}$ is continuous) that $\underline{y}$ was received given that $\hat{\underline{y}}$ was transmitted.

2.2.2 Binary Symmetric Channel

The binary symmetric channel (BSC) is one of the simplest and well-studied communication channels. It is a binary additive noise channel in which the noise process $\{Z_i\}$ consists of independent and identically distributed (i.i.d.) Bernoulli random variables. The channel behaviour can be described by

$$Y_i \;=\; X_i \oplus Z_i,$$

where $\oplus$ represents addition modulo-2 and the binary variables X_i, Y_i and Z_i are the channel input, output and noise, respectively, at time i. As the noise process is i.i.d., the channel can be fully described by the single parameter $P[Z_i = 1] \doteq p$. It is also instructive to describe the channel visually, as in Figure 2.1.

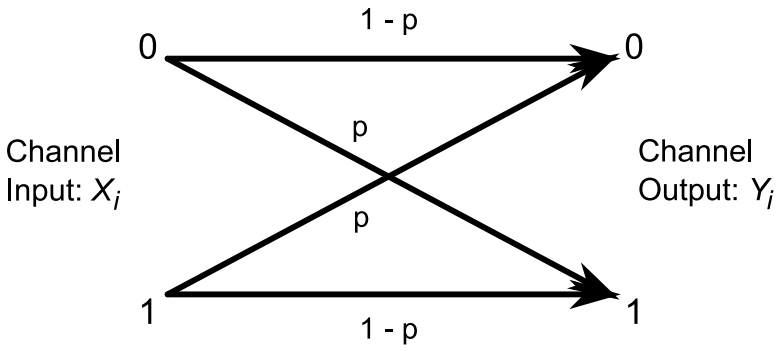

Figure 2.1: Visual representation of the binary symmetric channel (BSC).

2.2.3 Additive White Gaussian Noise Channel

Here we will provide a basic discussion of the additive white Gaussian noise (AWGN) channel in the context of binary phase-shift keying (BPSK) modulation. Generally speaking, modulation is the process of representing a digital message with an analog signal. We can model BPSK modulation as a mapping of messages from $\mathbb{F}_2$ to messages in $\mathbb{R}$. Specifically, the binary symbols 0 and 1 are mapped to the real numbers -1 and 1, respectively. In the context of BPSK modulation, we can model the AWGN channel as an additive noise channel over the real numbers. We let the real numbers X, Y and Z represent the channel input, output and noise, respectively, and model the channel's behaviour by

$$Y = X + Z$$

where X is either -1 or 1, and Z is a Gaussian random variable with mean 0 and variance σ^2. The channel is often represented visually as in Figure 2.2.

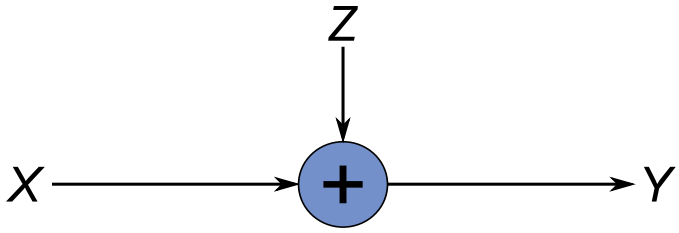

Figure 2.2: The additive white Gaussian noise (AWGN) channel.

The probability density function (pdf) for Z is given by

$$p_Z(z) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-z^2}{2\sigma^2}}. \tag{2.1}$$

We define the channel signal-to-noise ratio (SNR) as

$$SNR = \frac{\epsilon_s}{\sigma^2},$$

where ϵ_s denotes the average signal power per channel bit. We note that σ^2, the variance of the noise Z, is in fact equal to the average noise power and that $\epsilon_s = 1$, as we are always transmitting either -1 or 1. Due to the shape of the Gaussian distribution, optimal single bit (hard-decision) decoding, in terms of minimizing the probability of decoding error for each bit, is performed by minimizing the Euclidean distance between the received and decoded signal. This rule simplifies to decoding to 1 when the received signal is greater than or equal to 0, and decoding to -1 otherwise. Letting $\hat{X}$ represent the decoded signal, we can calculate the probability of bit error,

p_e:

$$\begin{aligned} p_e &= P[X=-1]P[\hat{X}=1|X=-1] + P[X=1]P[\hat{X}=-1|X=1] \\ &= P[X=1]P[Z>1] + P[X=-1]P[Z<-1] \\ &= P[Z<-1], \end{aligned}$$

where the last equality follows by symmetry of the normal pdf and the fact that $P[X=1]+P[X=-1]=1$. The bit-error probability under hard-decision decoding will be used later in order to compare performance over the BSC to performance over the AWGN channel by equating the BSC's cross-over probability to p_e.

2.2.4 First-Order Markov Noise Channel

An additive noise channel with memory is a channel in which the noise variables $\{Z_i\}$ are not independent of one another. The binary first-order Markov noise channel is a binary (modulo-2) additive noise channel with memory, and is one of the simplest examples of such a channel. In this channel, the binary-valued noise process $\{Z_i\}$ forms a Markov chain, which is to say that

$$P[Z_{i+1}|Z_i, Z_{i-1}, \ldots, Z_1] = P[Z_{i+1}|Z_i].$$

If the Markovian noise process is stationary, the channel can be fully described using two parameters, α and β, which define the channel's noise characteristics in the

following way:

$$P[Z_{i+1} = 1|Z_i = 0] = \alpha \quad \text{and} \quad P[Z_{i+1} = 1|Z_i = 1] = 1 - \beta.$$

It follows that

$$P[Z_{i+1} = 0|Z_i = 0] = 1 - \alpha \quad \text{and} \quad P[Z_{i+1} = 0|Z_i = 1] = \beta.$$

Alternately, we can describe the channel as having two states, S_0 and S_1, corresponding to the scenarios in which the channel noise at the previous time was 0 or 1, respectively. In each state, the channel has a distinct noise distribution according to the probabilities given above. This perspective is best described using the state transition diagram shown in Figure 2.3. If the channel is in the state S_0 at time i, then at time $i + 1$ it will be in state S_0 with probability $1 - \alpha$ (*i.e.*, $Z_i = 0$) or state S_1 with probability α (*i.e.*, $Z_i = 1$). On the other hand, if the channel is in state S_1 at time i, then at time $i + 1$ it will be in state S_1 with probability $1 - \beta$ (*i.e.*, $Z_i = 1$) or state S_0 with probability β (*i.e.*, $Z_i = 0$).

2.2.5 Non-ergodic Polya Channel

The Polya-contagion communication channel was introduced by Alajaji and Fuja in [2, Sections II-V]. It is an infinite memory, non-ergodic binary additive channel in which the noise is modelled by the Polya contagion urn scheme of [34]. It is shown in

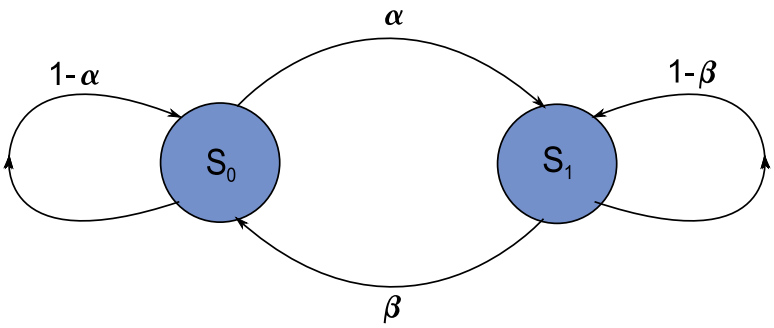

Figure 2.3: State transition diagram for the first-order Markov noise channel.

[2] that this channel belongs to the class of averaged channels with memory, admits a closed-form expression for its ϵ-capacity (defined later), and has a straightforward formulation for ML decoding. Thus, the infinite-memory Polya channel provides an interesting tool for modelling non-ergodic fading channels since the class of averaged channels with memory has recently been actively studied in the context of (non-ergodic) wireless fading channels and their outage capacity (studied in [35, Section VI]). It should be noted that the finite-memory version of the Polya channel and its generalization based on a finite queue have been recently shown to accurately model ergodic Rician fading channels, *e.g.*, see [36].

We denote the channel input, output and noise at time i by X_i, Y_i and Z_i, respectively. As a binary additive channel, its behaviour at time i is governed by

$$Y_i \;=\; X_i \oplus Z_i.$$

The distinguishing feature of the channel is the process by which the noise variables $\{Z_i\}$ are generated. These variables are statistically modelled by supposing that they

are generated by a random process involving drawing balls from an urn. Before giving a concise definition of the channel, we provide a general idea of this process.

A hypothetical urn contains some number of red and black balls. At each time instance i, a ball is randomly drawn from the urn. If that ball is red, then the noise, Z_i, at that time is 1, and otherwise it is 0. After a ball is selected, it is returned to the urn and an additional number of balls Δ of the same colour are added, hence changing the channel statistics at the next time instance. The noise process $\{Z_i\}$ is generated by repeating this process indefinitely. The infinite memory results from the cumulative nature of this process, *i.e.*, balls added at any given iteration remain for all successive iterations, hence influencing all subsequent ball selections.

We now introduce some notation to describe the process more precisely. We can define the state of the urn at time i as a pair $S_i = (R_i, B_i)$, where R_i and B_i represent the number of red and black balls, respectively, at time i. We represent the total number of balls at time i by $T_i = R_i + B_i$. The process is considered to start at time 1 with an initial state $S_1 = (R_1, B_1)$. At any time instance i we have the following noise distribution:

$$P[Z_i = 1] = \frac{R_i}{T_i}, \qquad P[Z_i = 0] = \frac{B_i}{T_i}.$$

Recalling that Δ is the number of *extra* balls returned to the urn after each draw, we

can define the channel state transition probabilities as follows:

$$\begin{aligned} P[S_{i+1} = (R_i + \Delta, B_i) \,|\, S_i = (R_i, B_i)] &= \frac{R_i}{T_i} \\ P[S_{i+1} = (R_i, B_i + \Delta) \,|\, S_i = (R_i, B_i)] &= \frac{B_i}{T_i}. \end{aligned}$$

We note that the sequence of channel states forms a Markov chain, as the channel state distribution at time $i+1$ depends only on the channel's state at time i.

To discuss the channel further, we introduce the following channel parameters, used in [2]. Firstly, $\delta = \frac{\Delta}{T_1}$, represents the ratio of extra balls added at each iteration as compared to the number of balls initially in the urn. Intuitively, this parameter is a measure of the degree of volatility of the channel. When δ is small, the channel's noise statistics will remain relatively stable, and vice versa. At the extreme end of this, we note that in the case where $\delta = 0$ ($\Delta = 0$), the channel reduces to a BSC with crossover probability $\frac{R_1}{T_1}$, as the number and ratio of balls stays constant. We also define $\rho = \frac{R_1}{T_1}$ and $\sigma = 1 - \rho$, the initial proportion of red and black balls, respectively. The expression for the block transition probability (the probability that the received block $\underline{y} = (y_1, \ldots, y_l)$ was received given that $\underline{x} = (x_1, \ldots, x_l)$ was transmitted and starting from the initial channel parameters) is given in [2]:

$$P[\underline{y}|\underline{x}] = \frac{\Gamma(\frac{1}{\delta})\Gamma(\frac{\rho}{\delta} + d)\Gamma(\frac{\sigma}{\delta} + l - d)}{\Gamma(\frac{\rho}{\delta})\Gamma(\frac{\sigma}{\delta})\Gamma(\frac{1}{\delta} + n)}$$

where d is the Hamming distance between $\underline{x}$ and $\underline{y}$, and $\Gamma(\cdot)$ is the gamma function

defined as $\Gamma(x) = \int_0^x t^{x-1}e^{-t}\,dt$.

In [2], a formulation for ML decoding over the infinite-memory non-ergodic Polya channel was derived in which either the minimum or maximum Hamming distance codeword is selected depending on the channel parameters and the distances of the minimum and maximum distance codewords. More precisely, if $\underline{y}$ is the received vector, we define

$$\begin{aligned} d_0 &= \frac{n}{2} + \frac{1-2\rho}{2\delta} \\ d_{min} &= \min_{\underline{c}\in C} d(\underline{y}, \underline{c}) \\ d_{max} &= \max_{\underline{c}\in C} d(\underline{y}, \underline{c}) \end{aligned}$$

where $d(\underline{y}, \underline{c})$ represents the Hamming distance between vectors $\underline{y}$ and $\underline{c}$. Now, if

$$|d_{max} - d_0| \leq |d_{min} - d_0|\,, \tag{2.2}$$

then ML decoding is equivalent to minimum distance decoding (MDD). Otherwise, if

$$|d_{max} - d_0| > |d_{min} - d_0|\,, \tag{2.3}$$

ML decoding is equivalent to maximum distance decoding. This formulation will be important later, when we develop an LP decoder for this channel. Further, we will see that in certain cases this formulation can be simplified.

The last point of discussion for the Polya channel is regarding its capacity. It is shown in [2] that the information capacity of the infinite memory non-ergodic Polya channel is 0. However, the channel has a non-zero ϵ-capacity. For a given $\epsilon > 0$, the ϵ-capacity, C_ϵ, of a channel is defined as the maximum ϵ-achievable rate. That is, the maximum rate, R, for which there exist, given sufficiently large block length, codes having rate arbitrarily close to R and probability of codeword error at most ϵ [2]. [3]

A formula for the ϵ-capacity of the Polya channel is given in [2]

$$C_\epsilon = 1 - F_V^{-1}(1-\epsilon), \tag{2.4}$$

where $F_V^{-1}(\cdot)$ is the inverse of the cumulative distribution function of the random variable $V \doteq h_b(U)$. U is defined as a beta-distributed $(\frac{\rho}{\delta}, \frac{\sigma}{\delta})$ random variable, and

$$h_b(x) \doteq -x\log_2(x) - (1-x)\log_2(1-x) \text{ for } x \in [0,1].$$

The ϵ-capacity is a theoretical lower-bound of error performance for a given code rate, and hence is a benchmark against which we can compare our simulation results. Because C_ϵ represents the maximum rate with which communication can occur at an error level of ϵ, determining the ϵ for which $C_\epsilon = R'$ gives us the smallest possible achievable error rate for a code of rate R'. More concisely, given a code rate, R', and channel parameters ρ and δ, we use (2.4) to determine the ϵ for which $C_\epsilon = R'$.

[3] The channel capacity is the $\lim_{\epsilon\to 0} C_\epsilon$.

Chapter 3

Linear Programming Decoding

3.1 Linear Programming

3.1.1 Definition

Linear programming (LP)[1] is a type of optimization problem which arises naturally in a variety of contexts, especially economic planning [37]. It can be defined as an optimization problem in which a linear cost function is minimized (or maximized) with respect to some set of variables under the restriction that the values taken by those variables satisfy a set of linear inequality constraints. More formally, we can express an LP as follows

$$\min\{\underline{d}^T \underline{x} | A\underline{x} \leq \underline{b}\}$$

[1]LP is used throughout to stand for both "Linear Programming" and "Linear Program," and this will be clear from the context.

where $\underline{d}^T = (d_1, d_2, \ldots, d_n)$ and $\underline{x}^T = (x_1, x_2, \ldots, x_n)$ are vectors in $\mathbb{R}^n$, $\underline{b}^T = (b_1, b_2, \ldots, b_n)$ is a vector in $\mathbb{R}^m$, and A is a real $m \times n$ matrix [38]. Here, $\underline{x}$ represents the set of variables over which the linear cost function, represented by $\underline{d}$, is being minimized. The matrix inequality $A\underline{x} \leq \underline{b}$ represents the set of m linear constraints on the variables $(x_1, x_2, \ldots, x_n)$. For example, if $n = 3$, $m = 2$,

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \quad \text{and} \quad \underline{b} = \begin{bmatrix} 7 \\ 8 \end{bmatrix},$$

then the matrix inequality $A\underline{x} \leq \underline{b}$ corresponds to the inequalities

$$x_1 + 2x_2 + 3x_3 \leq 7 \quad \text{and} \quad 4x_1 + 5x_2 + 6x_3 \leq 8.$$

We also note that this definition allows for equality constraints, which can be obtained by means of a pair of inequality constraints.

For our purposes, it is important that we introduce the concepts of *polyhedra* and *polytopes*. A subset of $\mathbb{R}^n$ of the form $\{\underline{x} | A\underline{x} \leq \underline{b}\}$ is called a polyhedron. Thus, it is equivalent to consider an LP to be the optimization of a linear cost function over a polyhedron. Further, we can define a polytope as the *convex hull* of some finite set of points in $\mathbb{R}^n$:

$$\mathcal{P}(S) = \left\{ \sum_{\underline{s} \in S} \lambda_{\underline{s}} \underline{s} : \lambda_{\underline{s}} \geq 0, \sum_{\underline{s} \in S} \lambda_{\underline{s}} = 1 \right\},$$

where S is a finite subset of $\mathbb{R}^n$. Finally, a subset of $\mathbb{R}^n$ is a polytope if and only if

it is a bounded polyhedron [38, Corollary 7.1c], meaning that any polytope is also a polyhedron. Thus, the minimization of a linear objective function over a polytope also constitutes an LP, and this is the form in which we will be expressing LPs.

3.1.2 Solving Linear Programs

In addition to the fact that LP arises naturally in a variety scenarios, one of its most important features, which makes it a very applicable paradigm for optimization, is that LPs are efficiently solvable. Here, we will briefly discuss the simplex algorithm and the ellipsoid algorithm, two well-known algorithms for solving the general LP problem.

The simplex method, due to Dantzig [39] in 1951, was the first algorithm designed to solve LPs and to this day remains the most important practical algorithm for solving LPs [38, p. 129]. The algorithm works under the principle that over any polytope, a linear function admits its minimum (or maximum) at a vertex. So, it follows that in order to find a global optimum of a linear function it suffices to explore the set of vertices of the region of optimization. This is exactly what the simplex algorithm does; it traverses the vertices of the polytope in a prescribed manner so that it is guaranteed to converge on a globally optimum vertex [38, p. 129-130]. There exist various sets of rules used by the simplex algorithm in order to traverse the polytope, and each of these corresponds to a unique version of the simplex method. The simplex algorithm has proved through empirical evidence to be very fast [38, p.

139]. However, despite its practical efficiency, for all known versions of the simplex method, examples can be constructed in which the algorithm requires an exponential amount of time to converge with respect to the size of the input. These examples, although not representative of naturally occurring problems, leave a theoretical gap in the sense that they provide an undesirable upper-bound (which can be reached) on the complexity of solving LPs.

This issue was resolved by Khachiyan [40] in 1979 who showed that LPs can be solved in polynomial time using an extension of the ellipsoid method, an algorithm used to solve nonlinear programming problems [38, p. 163]. The importance of the ellipsoid method for LPs is primarily a theoretical one, as it was the first algorithm which proved that LPs are solvable in polynomial time. However, from a practical standpoint, the simplex method is much more efficient, despite its weaker theoretical complexity [38, p. 170].

3.2 Decoding as an LP

In this section, we review the recent work of Feldman [1, 9] in which the problem of decoding linear error correcting codes transmitted over memoryless channels is formulated as an LP. An LP has two primary components: a linear cost function and a set of linear constraints defining a region of optimization (a polytope). So, in order to formulate decoding as an LP we will next consider each of these components.

3.2.1 Linear Cost Function

We herein assume that a binary source symbol $\underline{s} = (s_1, s_2, \ldots, s_k)$ is generated by a memoryless uniform source and is encoded by a linear (n, k) code C. The encoder output is transmitted over a DMC, and the binary n-vector $\underline{y} = (y_1, y_2, \ldots, y_n)$ is received. In this scenario, the optimal decoding rule, in terms of minimizing the probability of codeword error, is the maximum likelihood (ML) decoding rule. The ML decoding rule is to select a codeword $\hat{\underline{y}} \in C$ which maximizes the probability that $\hat{\underline{y}}$ was transmitted given that $\underline{y}$ was received. More precisely, ML decoding is performed by selecting the decoded codeword to be in the set $\operatorname{argmax}_{\underline{c} \in C} P[\underline{y}|\underline{c}]$ or, equivalently, the set

$$\operatorname*{argmax}_{\underline{c} \in C} \prod_{i=1}^{n} P[y_i|c_i].$$

Next, we derive a linear cost function whose maximum is attained by the same set of codewords that maximize the ML decoding metric. Here, and for the remainder of this work, we let $P[y_i|c_i]$, $P[y_i|1]$ and $P[y_i|0]$ represent the probabilities that y_i was

received given that that c_i, 1 and 0, respectively, were transmitted at time i:

$$\begin{aligned}
\underline{\hat{y}} &= \underset{\underline{c}\in C}{\operatorname{argmax}} \prod_{i=1}^{n} P[y_i|c_i] \\
&= \underset{\underline{c}\in C}{\operatorname{argmax}} \log\left(\prod_{i=1}^{n} P[y_i|c_i]\right) \\
&= \underset{\underline{c}\in C}{\operatorname{argmax}} \sum_{i=1}^{n} \log\left(P[y_i|c_i]\right) \\
&= \underset{\underline{c}\in C}{\operatorname{argmax}} \sum_{i:c_i=1} \log\left(P[y_i|1]\right) c_i + \sum_{i:c_i=0} \log\left(P[y_i|0]\right) \\
&= \underset{\underline{c}\in C}{\operatorname{argmax}} \sum_{i=1}^{n} \log\left(P[y_i|1]\right) c_i + \sum_{i:c_i=0} \log\left(P[y_i|0]\right) \\
&\qquad - \sum_{i=1}^{n} \log\left(P[y_i|0]\right) c_i + \sum_{i:c_i=1} \log\left(P[y_i|0]\right) \\
&= \underset{\underline{c}\in C}{\operatorname{argmax}} \sum_{i=1}^{n} \log\left(\frac{P[y_i|1]}{P[y_i|0]}\right) c_i + \sum_{i=1}^{n} \log\left(P[y_i|0]\right) \\
&= \underset{\underline{c}\in C}{\operatorname{argmax}} \sum_{i=1}^{n} \log\left(\frac{P[y_i|1]}{P[y_i|0]}\right) c_i \\
&= \underset{\underline{c}\in C}{\operatorname{argmax}} \underline{\gamma} \cdot \underline{c}
\end{aligned} \tag{3.1}$$

where $\underline{\gamma} = (\gamma_1, \ldots, \gamma_n)$ is defined as the log-likelihood ratio (LLR)

$$\log\left(\frac{P[y_i|1]}{P[y_i|0]}\right), \tag{3.2}$$

as in [1]. So $\underline{\gamma}$ is a linear cost function whose optima among the set of codewords are the same as those for the ML decoding metric.

3.2.2 Region of Optimization

Since an LP achieves its optimum at a vertex, it follows that in order to formulate the ML decoding problem as an LP we must define a polytope which contains the set of codewords as its vertex set. The most natural way to arrive at such a polytope is to consider the *codeword polytope*, $\mathcal{P}(C)$, that is, the convex hull of the code C over $\mathbb{R}^n$:

$$\mathcal{P}(C) = \left\{ \sum_{\underline{c} \in C} \lambda_{\underline{c}} \underline{c} : \lambda_{\underline{c}} \geq 0, \sum_{\underline{c} \in C} \lambda_{\underline{c}} = 1 \right\}.$$

Thus, solving the LP

$$\max_{x \in \mathcal{P}(C)} \underline{\gamma} \cdot \underline{x} \tag{3.3}$$

where $\underline{\gamma}$ is as in (3.1), yields the solution of the ML decoding problem. This follows because we are essentially maximizing over the set of vertices of $\mathcal{P}(C)$ which is exactly equal to the discrete set C, and, as shown, the linear cost function $\underline{\gamma}$ shares its set of optimizing vectors with the ML metric. So (3.3) constitutes an LP which solves the ML decoding problem. However, it is known that the ML decoding problem for an arbitrary code is NP-hard [41]. It follows that solving the LP (3.3) is also NP-hard. We note here that although an arbitrary LP can be solved in polynomial time in terms of the size of its input, this does not guarantee anything about the efficiency of solving (3.3) (in terms of the code length n). That is, if the size of the input, namely,

the size of the description of the polytope in (3.3), is exponentially large in terms of n, then the time required to solve (3.3) is also exponential in terms of n.

3.2.3 Relaxing the Problem

In a situation where solving an LP is prohibitively complex, a common practice is to "relax" the problem. The idea of a relaxation in this context is to find a polytope that contains the code as a subset of its vertex set, but which has some property that allows an LP defined over it to be solved more easily. In this case, that property is a compact representation of the polytope in terms of linear constraints. Such a polytope is called a *relaxation* of the codeword polytope $\mathcal{P}(C)$.

A certain relaxation of the codeword polytope has received much recent attention [1], [23]. This is the polytope which, given a code C of length n, and a subset of the dual code, $H \subset C^{\perp}$, is defined as

$$\mathcal{Q}(H) = \bigcap_{\underline{h} \in H} \mathcal{P}(\underline{h}^{\perp}), \tag{3.4}$$

where $\mathcal{P}(\underline{h}^{\perp})$ is the codeword polytope of the code $\underline{h}^{\perp} = \{\underline{c} \in \{0,1\}^n : \underline{h} \cdot \underline{c} \equiv 0 \pmod 2\}$.

For any $H = \{\underline{h}_1, \underline{h}_2, \ldots, \underline{h}_l\} \subset C^{\perp}$, the polytope $\mathcal{Q}(H)$ contains C as a subset of its vertex set, $\mathcal{V}(\mathcal{Q}(H))$. This follows by noting that C is a subset of $\underline{h}_i^{\perp}$ for all i, and so $\mathcal{P}(\underline{h}_i^{\perp})$ contains C as a subset of its vertex set for all i; thus, clearly by intersecting all such polytopes, the resulting polytope will also contain C as a subset of its vertex

set. Consequently, the LP

$$\max_{\underline{x}\in\mathcal{Q}(H)} \underline{\gamma}\cdot\underline{x} \tag{3.5}$$

constitutes a relaxation of the LP that represents ML decoding. This decoder will be referred to throughout this work as the "standard LP decoder".

Now, any standard LP-solving algorithm requires that the LP to be solved has its constraints represented via linear inequalities, *i.e.*, as a subset of $\mathbb{R}^n$ in the form $\{\underline{x}|A\underline{x}\leq\underline{b}\}$. The advantage of using the relaxation $\mathcal{Q}(H)$ is that there is a straightforward representation of the constraint $\underline{x}\in\mathcal{Q}(H)$ as a set of linear inequalities. The polytope $\mathcal{Q}(H)$ can also be expressed as [1, Theorem 4]

$$\mathcal{Q}(H)=\bigcap_{\underline{h}\in H}\Pi(\mathrm{supp}(\underline{h})), \tag{3.6}$$

where for $\underline{h}=(h_1,\ldots,h_n)$, $\mathrm{supp}(\underline{h})=\{i:h_i=1\}$ and for $S\subset\{1,2,\ldots,n\}$, $\Pi(S)$ denotes the polyhedron

$$\Pi(S) \;=\; \bigcap_{\substack{J\subset S\\ |J|\text{odd}}} \left\{ \begin{array}{c} (x_1,\ldots,x_n)\in[0,1]^n: \\ \\ \sum_{j\in J}x_j-\sum_{i\in S\setminus J}x_i\leq|J|-1 \end{array} \right\}. \tag{3.7}$$

Let $\mathcal{J}(\mathcal{Q}(H))=\mathcal{V}(\mathcal{Q}(H))\cap\{0,1\}^n$ denote the set of *integral vertices* (*i.e.,* vertices whose coordinates are all integers) of $\mathcal{Q}(H)$. We discussed above that $C\subset\mathcal{V}(\mathcal{Q}(H))$ and since C consists of integral vectors we clearly have that $C\subset\mathcal{J}(\mathcal{Q}(H))$. If H

is a spanning subset (over $\mathbb{F}_2$) of $C^\perp$, so that the vectors in H form (the rows of) a parity-check matrix of C, then we in fact have $C = \mathcal{J}(\mathcal{Q}(H))$. This is because if $\underline{x} \in \{0,1\}^n$ is not in C, then $\underline{x} \notin h^\perp$ for some $h \in H$, and hence, $\underline{x} \notin \mathcal{P}(h^\perp) \supset \mathcal{Q}(H)$. The polytope $\mathcal{Q}(H)$ in this case is the "projected polytope" $\bar{\mathcal{Q}}$ of Feldman *et. al.* [1, p. 958]. The fact that $C = \mathcal{J}(\mathcal{Q}(H))$ for such a polytope $\mathcal{Q}(H)$ implies that the polytope has the following "ML certificate" property [1, Proposition 2]: if the LP (3.5) attains its minimum at some $\underline{x} \in \mathcal{J}(\mathcal{Q}(H))$, then $\underline{x}$ is guaranteed to be the ML codeword. However, it is possible that the LP attains its minimum at some non-integral vertex $\underline{x} \in \mathcal{V}(\mathcal{Q}(H)) - \mathcal{J}(\mathcal{Q}(H))$, in which case either decoding failure is declared or some heuristic is used.

3.2.4 Complexity

If we let T represent the set of linear constraints required to define the polytope $\mathcal{Q}(H)$, then we have that T contains the following constraints: for each $\underline{h} \in H$, we have the following constraint for every odd subset, J, of $\mathrm{supp}(\underline{h})$,

$$\sum_{j \in J} x_j - \sum_{i \in supp(\underline{h}) \setminus J} x_i \leq |J| - 1.$$

In addition, T contains the following constraints, ensuring that the polytope is contained within the unit cube:

$$(x_1, x_2 \ldots, x_n) \in [0,1]^n.$$

So, for each $\underline{h} \in H$ we have a set of linear constraints whose cardinality is equal to the number of odd subsets of $B = \text{supp}(\underline{h})$. The number of odd subsets of a given set B is equal to $2^{|B|-1}$. Also, it is easy to see that $2n$ constraints are required to restrict the polytope to the unit cube. In particular, if we consider a rate R, (n, k) LDPC code C and take $H \subset C^{\perp}$ corresponding to the rows of a parity-check matrix of C with constant row-degree d, then we can explicitly calculate the number of linear constraints required to represent $\mathcal{Q}(H)$:

$$\begin{aligned} |T| &= \sum_{i=1}^{n-k} \left(2^{|\text{supp}(\underline{h}_i)|-1}\right) + 2n \\ &= \sum_{i=1}^{n-k} \left(2^{d-1}\right) + 2n \\ &= (n-k)2^{d-1} + 2n \\ &= n(2 + (1-R)2^{d-1}). \end{aligned}$$

Assuming that the row-degree d is not dependent on n, as is the case in an LDPC code, then what the above shows is that the number of constraints required to represent $\mathcal{Q}(H)$ is linear in the length of the code n. Further, in the case that we do not have a constant row-degree d, but the row-degree is bounded by some constant, the above expression acts as an upper-bound on the number of constraints required to represent $\mathcal{Q}(H)$. We note here that the size of the representation of $\mathcal{Q}(H)$ is not only dependent on the code C, but the choice of H. In particular, if we are selecting H to correspond to the rows of a parity-check matrix for C, then this number will

depend on the row-degree distribution of the particular parity-check matrix selected. Generally speaking, an LDPC code is designed from a parity-check matrix which has desirable row-degree properties, and so this matrix is used to define $\mathcal{Q}(H)$.

As previously discussed, the efficiency of a particular LP solver depends on the size of the LP representation, which is proportional to the number of variables and linear inequalities forming the constraints. Following the discussion above, if we select H to be a parity-check matrix of an LDPC code, then the size of the input is linear in n. Thus, because an LP is guaranteed to be solvable in polynomial time with respect to the size of the input, we know that the relaxed LP (3.5) as defined for an LDPC code is guaranteed to be solvable in polynomial time with respect to the code-length.

It should also be noted that the number of constraints, and hence the complexity of the LP, is exponentially dependent on the maximum row-degree of the parity-check matrix used. Thus, if the maximum row-degree grows with the code-length n, then the complexity of the LP is exponentially dependent on n. So the relaxation discussed herein is efficient for LDPC codes, where the maximum row degree is small and fixed as n increases. In [1], an alternate polytope is defined which can be efficiently represented for arbitrary (dense) parity-check matrices. In addition, the problem of dense parity-check matrices can be addressed by using the adaptive LP method [23], which is discussed next.

3.2.5 Adaptive LP Decoding

In [23], Taghavi and Siegel introduce an algorithmic variation on the relaxed LP decoding scheme of [1] which serves to increase the efficiency of LP decoding. The proposed algorithm is identical to solving (3.5) in terms of performance but has lower complexity, especially when defining the relaxed polytope using a dense parity-check matrix.

The general idea is to start with some minimal subset of the constraints from (3.7), solve the corresponding LP, and then find some constraint(s) from (3.7) which make(s) the solution invalid. Those constraints (called *cuts*) are then added to the LP, and the LP is solved again. This process is repeated until the solution of the LP violates none of the constraints from (3.7), at which point the solution is the same as that which would be obtained by solving (3.5) [23, Claim 1].

The adaptive LP decoding algorithm can be described more precisely as follows. First, we solve the LP:

$$\begin{gathered} \min \underline{\gamma} \cdot \underline{x} \quad \text{subject to:} \\ 0 \leq x_i \text{ if } \gamma_i > 0, \\ x_i \leq 1 \text{ if } \gamma_i < 0, \end{gathered} \tag{3.8}$$

and obtain some vector $\underline{\hat{x}} = (\hat{x}_1, \ldots, \hat{x}_n)$. Next, the following algorithm is used to find constraints which are violated by $\underline{\hat{x}}$. We assume that the coordinates $(\hat{x}_1, \ldots, \hat{x}_n)$

are sorted in decreasing order. For each $\underline{h} \in H$ we proceed according to Algorithm 1, letting $\text{supp}(\underline{h}) = (s_1, \ldots, s_t), s_1 \leq s_2 \leq \ldots \leq s_t$.

Algorithm 1 Find Cut for $\underline{h}$

set $j = 1$ and $J = \{s_1\}$
if $\sum_{j \in J} \hat{x}_j - \sum_{i \in supp(\underline{h}) \setminus J} \hat{x}_i \leq |J| - 1$ is violated **then**
    a cut has been found; exit.
else
    set $j = j + 2$.
    if $j \leq t$ **then**
        set $J = J \cup \{s_{j-1}, s_j\}$
        if $|J| - 1 < \sum_{i \in J} \hat{x}_i \leq |J|$ **then**
            goto 2:
        else
            there is no cut associated with $\underline{h}$; exit.
        end if
    else
        there is no cut associated with $\underline{h}$; exit.
    end if
end if

If any constraints are yielded by Algorithm 1, they are added to the LP (3.8), the LP is solved again, and the process of searching for cuts is repeated. Otherwise, if no new constraints are yielded, then as stated above [23, Claim 1], the solution is the same as that of the standard relaxed LP (3.5). The reader is directed to [23] for an explanation of why this algorithm finds all cuts.

Chapter 4

LP Decoding for Non-Uniform Sources

The original formulation of LP decoding by Feldman, described in Chapter 3, is designed for use with uniform sources. Accordingly, the standard LP decoder (3.5) does not take the *a-priori* source information into account when the source is non-uniform. To address this issue, we modify the standard LP decoder to account for this *a-priori* information in order to achieve enhanced performance in this context.

4.1 LP Decoding for Non-Uniform Sources with Systematic Codes

Here, a modified version of the LP decoding scheme is applied to the scenario of decoding for a memoryless non-uniform source transmitting over a memoryless BSC. In scenarios with non-uniformity at the source, we would like to account for the *a-priori* codeword probabilities. Thus, if we want to design an LP decoder for this situation, we must somehow include the *a-priori* codeword probabilities into a linear cost function involving the codeword variables. The optimal decoder, in terms of minimizing the probability of codeword error, in this context is the maximum *a-posteriori* probability (MAP) decoding rule. For a systematic (n, k) code C transmitted over a memoryless BSC, it is possible to derive a linear cost function of the c_i variables whose set of optimizing codewords is the same as that for the (optimal) MAP metric. We assume a non-uniform binary source with the probability of 1 and 0 being p_1 and p_0, respectively and that the binary n-vector $\underline{y} = (y_1, y_2, \ldots, y_n)$ is received. We will also assume for the remainder of this chapter that the source statistics p_1 and p_0 are known to the decoder. Lastly, here, and for the remaind er of this work, we let $P[c_i]$, $P[1]$ and $P[0]$ be the probabilities that the source output is c_i, 1 and 0, respectively,

at any given time, *i.e.*, $P[1] = p_1$ and $P[0] = p_0$:

$$\begin{aligned}
\underset{\underline{c}\in C}{\text{argmax}}\ P[\underline{c}|\underline{y}] &= \underset{\underline{c}\in C}{\text{argmax}}\ P[\underline{c}]P[\underline{y}|\underline{c}] \\
&= \underset{\underline{c}\in C}{\text{argmax}} \prod_{i=1}^{k} P[c_i] \prod_{i=1}^{n} P[y_i|c_i] \\
&= \underset{\underline{c}\in C}{\text{argmax}} \sum_{i=1}^{k} \log\left(P[c_i]\right) + \underline{\gamma}\cdot\underline{c} \\
&= \underset{\underline{c}\in C}{\text{argmax}} \sum_{\substack{i:c_i=1 \\ i\le k}} \log\left(P[1]\right) c_i + \sum_{\substack{i:c_i=0 \\ i\le k}} \log\left(P[0]\right) + \underline{\gamma}\cdot\underline{c} \\
&= \underset{\underline{c}\in C}{\text{argmax}} \sum_{i=1}^{k} \log\left(P[1]\right) c_i + \sum_{i=1}^{k} \log\left(P[0]\right) \\
&\qquad - \sum_{i=1}^{k} \log\left(P[0]\right) c_i + \underline{\gamma}\cdot\underline{c} \\
&= \text{argmax}\ \underline{\gamma}\cdot\underline{c} + \sum_{i=1}^{k} \log\left(\frac{p_1}{p_0}\right) c_i + const \\
&= \text{argmax}\ \underline{\gamma}^{+}\cdot\underline{c},
\end{aligned}$$

where the second equality follows because the code is systematic [1], and the n-vector $\underline{\gamma}^{+} = (\gamma_1^{+}, \ldots, \gamma_n^{+})$ is defined as

$$\gamma_i^{+} = \begin{cases} \gamma_i + \log\left(\frac{p_1}{p_0}\right) & \text{for } 1 \le i \le k \\ \gamma_i & \text{for } k < i \le n. \end{cases}$$

where γ_i is given by 3.2). Thus, for a systematic code, we have a linear cost function that can be used to formulate an LP decoder which makes use of the *a-priori* codeword

[1] We assume without loss of generality that the systematic bits occur in the first k bit positions of the code.

information. This linear cost function is optimal, in the sense that the MAP codeword would maximize this function over the set C. We can thus use this linear cost function to formulate an LP by maximizing it over the relaxed codeword polytope (3.4). If we select $H \subset C^{\perp}$ corresponding to the rows of a parity-check matrix for C, then we can construct the relaxed codeword polytope $\mathcal{Q}(H)$ and formulate a sub-optimal MAP decoder by selecting the decoded vector $\hat{\underline{y}}$ to be in the set

$$\max_{\underline{x} \in \mathcal{Q}(H)} \underline{\gamma}^{+} \cdot \underline{x}. \tag{4.1}$$

If $\hat{\underline{y}}$ is an integral vector, then it is known to be the MAP codeword. This follows because the set of integral vertices of $\mathcal{Q}(H)$ is exactly equal to C, and so if $\hat{\underline{y}}$ is integral, then $\hat{\underline{y}} \in C$. Thus, if $\hat{\underline{y}}$ is an optimum among $\mathcal{V}(\mathcal{Q}(H)) \supseteq C$, we also have that $\hat{\underline{y}}$ is an optimum among C. This gives us a "MAP Certificate Property" analogous to the "ML Certificate Property" discussed in [1, Proposition 2] and Section 3.2.3. When the LP optimum $\hat{\underline{y}}$ is a non-integral vector, we can either declare decoding failure, or as a heuristic alternative, we can round $\hat{\underline{y}}$ and return the result. We will refer to the decoder (4.1) as the systematic joint source-channel (JSC) decoder.

4.2 LP Decoding for Non-Uniform Sources with Non-Systematic Codes

The proposed technique works; however, it has been demonstrated (*e.g.*, see [42], [10], [11] and the references therein) that in scenarios with non-uniformity at the source, non-systematic codes provide substantial performance gains when compared to systematic codes. Because the above derivation of a linear MAP cost function relies completely on the code being systematic, another approach must be considered.

One way to incorporate the *a-priori* codeword information into an LP decoder without transmitting a systematic code is to encode using a systematic code of rate lower than desired, and then puncture the systematic bits before transmission. More precisely, suppose that we wish to use an LDPC code of rate $R = \frac{k}{n}$, and blocklength n. First, we select a systematic $(n+k, k)$ LDPC code, $\widetilde{C}$. Now, suppose we encode a source symbol $\underline{s}$ using $\widetilde{C}$, but, before transmission, we strip away the first k (systematic) bits. That is, we transmit only the last n bits of the encoded block. Given that the n-vector $\underline{y}$ is received, the MAP decoding metric can be linearized in a similar

fashion

$$\begin{aligned}
\max_{\underline{c}\in\widetilde{C}} P[\underline{c}]P[\underline{y}|\underline{x}] &= \max_{\underline{c}\in\widetilde{C}} \prod_{i=1}^{k} P[c_i] \prod_{i=k+1}^{n+k} P[y_{i-k}|c_i] \\
&= \operatorname*{argmax}_{\underline{c}\in\widetilde{C}} \sum_{i=1}^{k} \log\left(P[c_i]\right) + \underline{\gamma}\cdot\underline{c'} \\
&= \operatorname*{argmax}_{\underline{c}\in\widetilde{C}} \sum_{\substack{i:c_i=1 \\ i\leq k}} \log\left(P[1]\right) c_i + \sum_{\substack{i:c_i=0 \\ i\leq k}} \log\left(P[0]\right) + \underline{\gamma}\cdot\underline{c'} \\
&= \operatorname*{argmax}_{\underline{c}\in C} \sum_{i=1}^{k} \log\left(P[1]\right) c_i + \sum_{i=1}^{k} \log\left(P[0]\right) \\
&\qquad - \sum_{i=1}^{k} \log\left(P[0]\right) c_i + \underline{\gamma}\cdot\underline{c'} \\
&= \operatorname{argmax}\ \underline{\gamma}\cdot\underline{c'} + \sum_{i=1}^{k} \log\left(\frac{p_1}{p_0}\right) c_i + const \\
&= \max_{\underline{c}\in\widetilde{C}} \underline{\gamma}^*\cdot\underline{c},
\end{aligned}$$

where $\underline{c'} = (c_{k+1}, \ldots, c_{k+n})$ and $\underline{\gamma}^* = (\gamma_1^*, \ldots, \gamma_{n+k}^*)$ is defined as

$$\gamma_i^* = \begin{cases} \log\left(\frac{p_1}{p_0}\right), & \text{for } 1 \leq i \leq k \\ \log\left(\frac{P[y_{i-k}|x_i=1]}{P[y_{i-k}|x_i=0]}\right), & \text{for } k < i \leq n+k. \end{cases} \tag{4.2}$$

Now, if we select $\widetilde{H} \subset \widetilde{C}^{\perp}$ corresponding to the rows of a parity-check matrix of $\widetilde{C}$, we can construct the relaxed polytope $\mathcal{Q}(\widetilde{H})$, as in the previous section. Decoding can then be performed by selecting $\underline{\hat{y}}$ to be in the set

$$\operatorname*{argmax}_{\underline{x}\in\mathcal{Q}(\widetilde{H})} \underline{\gamma}^*\cdot\underline{x}. \tag{4.3}$$

For the same reasons as with the decoder (4.1), the above decoder has the "MAP certificate" property.

Here, the first k components of the new cost function (4.2) correspond only to the source statistics (the systematic bits of $\widetilde{C}$), and the last n components correspond only to the received signal information, $\underline{y}$, and the channel statistics. Essentially, the "extended codeword polytope", $\mathcal{Q}(\widetilde{H})$, allows the systematic bits to be linked to the punctured codewords during decoding without having to transmit them. We will refer to the decoder (4.3) as the non-systematic JSC decoder or the "extended polytope" decoder.

4.3 The AWGN Channel

LP decoders can be designed for decoding BPSK modulated codes over the AWGN channel, as described in [9]. In order to do this, we must re-design the cost function in order to deal with continuous output. Here, we demonstrate this for the case of a non-uniform source and a systematic code, though it can easily be extended to the non-systematic decoder described previously. The optimal decoder (in terms of minimizing the probability of codeword error) in this context is the MAP decoding rule. For a BPSK modulated systematic (n, k) code C transmitted over the AWGN channel, it is possible to derive a linear cost function of the c_i variables whose set of optimizing codewords is the same as that for the (optimal) MAP metric. We assume a non-uniform source with the probability of 1 and 0 being p_1 and p_0, respectively

and that the real n-vector $\underline{y}$ is received. Skipping steps where prior derivations can fill in the gaps, we have

$$\begin{aligned}
\underset{\underline{c}\in C}{\operatorname{argmax}}\ P[\underline{c}]P[\underline{y}|\underline{c}] &= \underset{\underline{c}\in C}{\operatorname{argmax}} \prod_{i=1}^{k} P[c_i] \prod_{i=1}^{n} P[y_i|c_i] \\
&= \underset{\underline{c}\in C}{\operatorname{argmax}} \sum_{i=1}^{k} \log\left(\frac{p_1}{p_0}\right) c_i + \sum_{i=1}^{n} \log\left(\frac{P[y_i|1]}{P[y_i|0]}\right) c_i \\
&= \underline{\gamma}^o \cdot \underline{c},
\end{aligned}$$

where $\underline{\gamma}^o = (\gamma_1^o, \ldots, \gamma_n^o)$ is defined as

$$\gamma_i^o = \begin{cases} \log\left(\frac{P[y_i|1]}{P[y_i|0]}\right) + \log\left(\frac{p_1}{p_0}\right) & \text{for } 1 \le i \le k \\ \log\left(\frac{P[y_i|1]}{P[y_i|0]}\right) & \text{for } k < i \le n. \end{cases}$$

The main difference here is that the values $P[y_i|1]$ and $P[y_i|0]$ represent densities from the pdf of the Gaussian noise, where before they represented probabilities. Specifically, letting Z be the random variable corresponding to the noise, we have

$$P[y_i|1] = p_Z(y_i - 1) \quad \text{and} \quad P[y_i|0] = p_Z[y_i + 1],$$

where $p_Z(\cdot)$ is the pdf of Z, given by (2.1).

Now, if we select $H \subset C^\perp$ corresponding to the rows of a parity-check matrix of C, we can construct the relaxed polytope $Q(H)$, as in the previous sections. Decoding

for the AWGN channel can then be performed by selecting $\underline{\hat{y}}$ to be in the set

$$\underset{\underline{x} \in \mathcal{Q}(\widetilde{H})}{\operatorname{argmax}} \underline{\gamma}^{o} \cdot \underline{x}. \tag{4.4}$$

4.4 Simulation Results

The systematic and non-systematic JSC decoders (4.1) and (4.3) were implemented and compared with the standard LP decoder (3.5) in terms of the probability of codeword error (PCE) for non-uniform sources with $p_1 = 0.9, 0.8$ and 0.7 over the BSC. For the systematic JSC decoder, a systematic $(200, 100)$ $(3, 6)$-regular LDPC code was used. For the non-systematic JSC decoder, a systematic $(300, 100)$ $(3, 9)$-regular LDPC code was used, and the first 100 (systematic) bits were punctured so as to give an equivalent effective rate of $\frac{1}{2}$.

In Figure 4.1, with the highly non-uniform source where $p_1 = 0.9$, substantial gains are obtained by the systematic JSC decoder as compared to the standard LP decoder. This is expected, as the standard LP decoder does not make use of the (high amount of) *a-priori* codeword information. Further, when we use the extended polytope (non-systematic) decoder with a punctured systematic code, we see additional gains for the same effective rate of $R = \frac{1}{2}$.

In Figure 4.2, where $p_1 = 0.8$, moderate gains are obtained by the systematic JSC decoder when compared to the standard LP decoder. Smaller gains are expected here as the standard LP decoder is missing out on a smaller amount of *a-priori* codeword

information when $p_1 = 0.8$, as opposed to when $p_1 = 0.9$. Interestingly, when we use the extended polytope decoder with the punctured systematic code of equivalent rate, we in fact see worse performance when compared to the systematic decoder; however, it still performs slightly better than the standard LP decoder. Here, what we are observing is that the losses due to code puncturing [10] outweigh the gains associated with using a non-systematic code. In the case where $p_1 = 0.9$, the gains associated with using a non-systematic code are greater, and hence in that scenario the non-systematic decoder has better performance.

In Figure 4.3, where $p_1 = 0.7$, we see very modest gains when comparing the systematic JSC decoder with the standard LP decoder. Again, this result is to be expected, as when $p_1 = 0.7$ there are even smaller amounts of a-priori codeword information to be exploited by the JSC decoder. Further, we again see that it is not beneficial to use the extended polytope decoder, which in this case performs worse than even the standard LP decoder. This is a consequence of the same trade-off discussed above, but with even less gain associated with using a non-systematic code.

From these simulations, we can infer the existence of a value of p_1, say p_1^*, for which the systematic and non-systematic decoders perform nearly equally well. Ignoring complexity, we would have that for sources with p_1 greater than p_1^*, it is advantageous to use the non-systematic decoder (4.3), and otherwise it is advantageous to use the systematic decoder. Based on the results presented herein, p_1^* would lie somewhere between 0.8 and 0.9. We also note that the value of p_1^* would likely vary depending

on the range of values of the channel BER p considered, as the relative performance of the systematic and non-systematic decoders appears to vary depending on p.

Figure 4.4 compares the performance of a $(1000, 500)$ $(3, 6)$-regular LDPC code to a $(200, 100)$ $(3, 6)$-regular LDPC code under decoder (4.1) with $p_1 = 0.7$. We note that for high values of p the length-1000 code performs worse than the length-200 code; however, for lower values of p, which are of greater interest, the opposite is true, demonstrating a substantial gain in performance associated with using larger block-lengths.

Finally, we include Figure 4.5 which compares the JSC LP decoder for the AWGN channel (4.4) to that for the BSC (4.1) using the same systematic $(200, 100)$ $(3, 6)$-regular LDPC code when $p_1 = 0.8$. This plot demonstrates an approximately 2dB gain when communicating over the AWGN channel with the corresponding decoder, as compared to the BSC. The horizontal axis, p, represents the BSC crossover probability and the equivalent hard-decision error rate p_{err} for the AWGN channel (see Section 2.2.3).

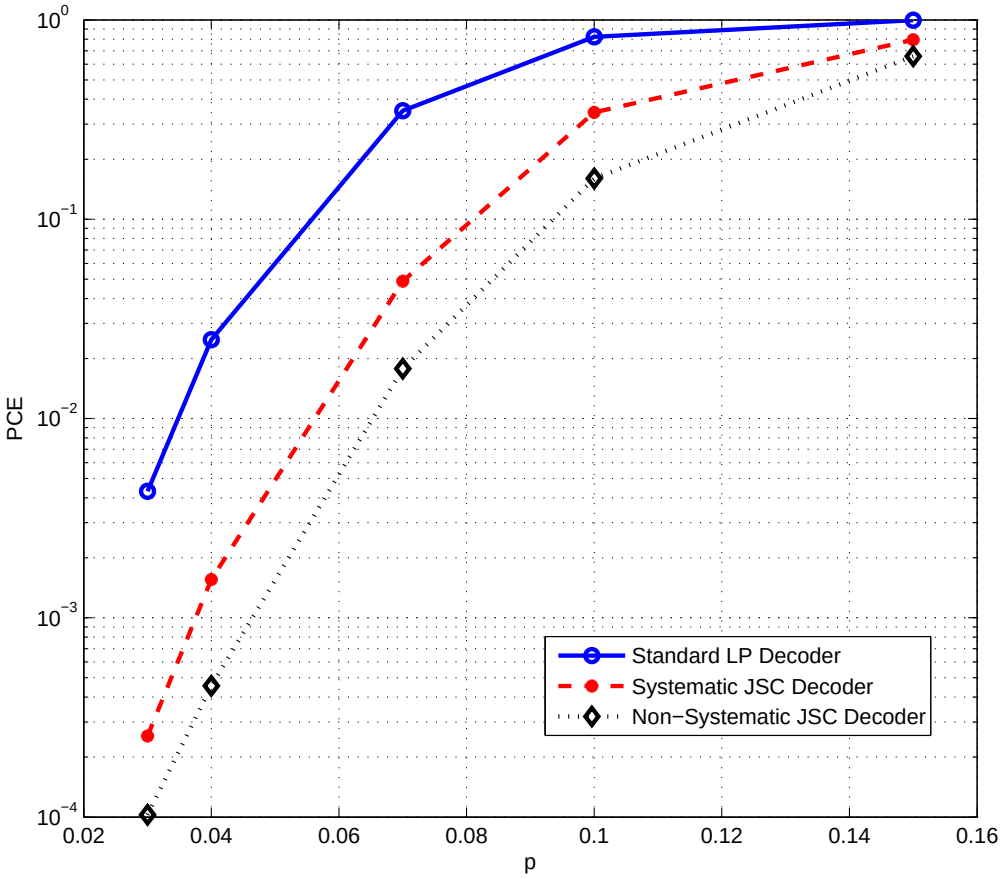

Figure 4.1: Standard LP Decoder vs. Modified LP Decoders for Non-Uniform Source $n = 200$, $p_1 = 0.9$, $R = \frac{1}{2}$ over the BSC.

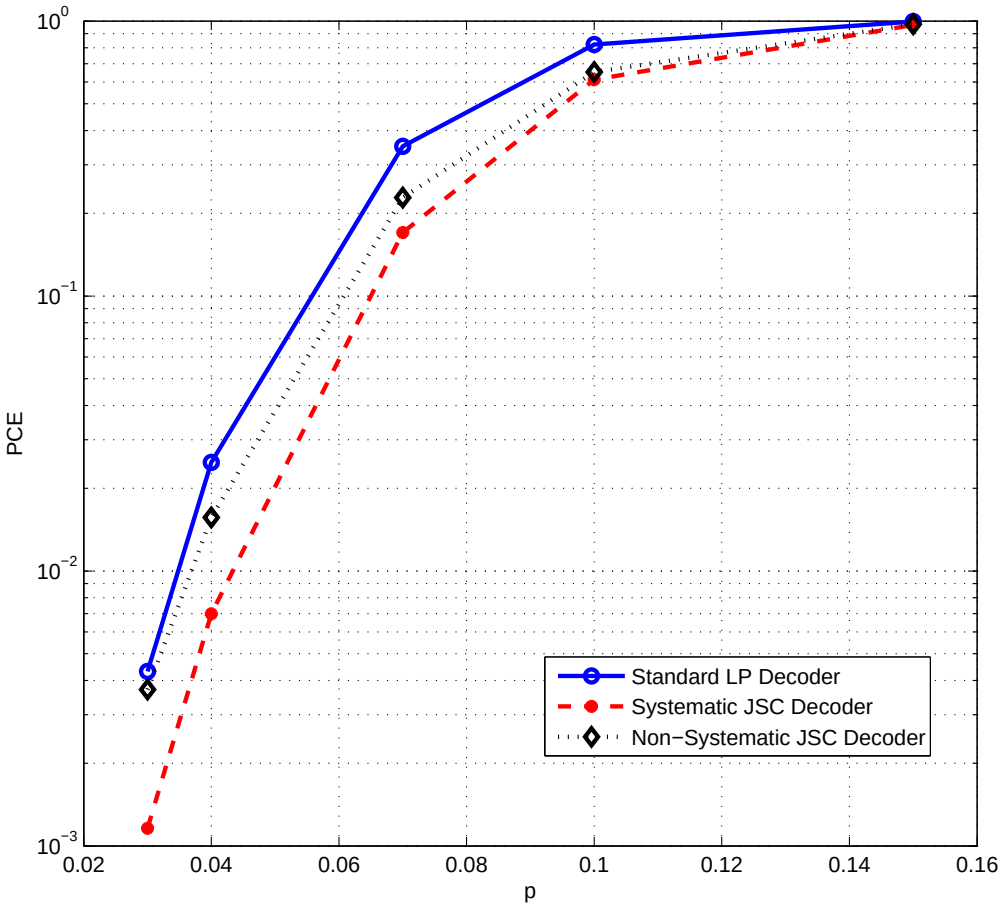

Figure 4.2: Standard LP Decoder vs. Modified LP Decoders for Non-Uniform Source $n = 200$, $p_1 = 0.8$, $R = \frac{1}{2}$ over the BSC.

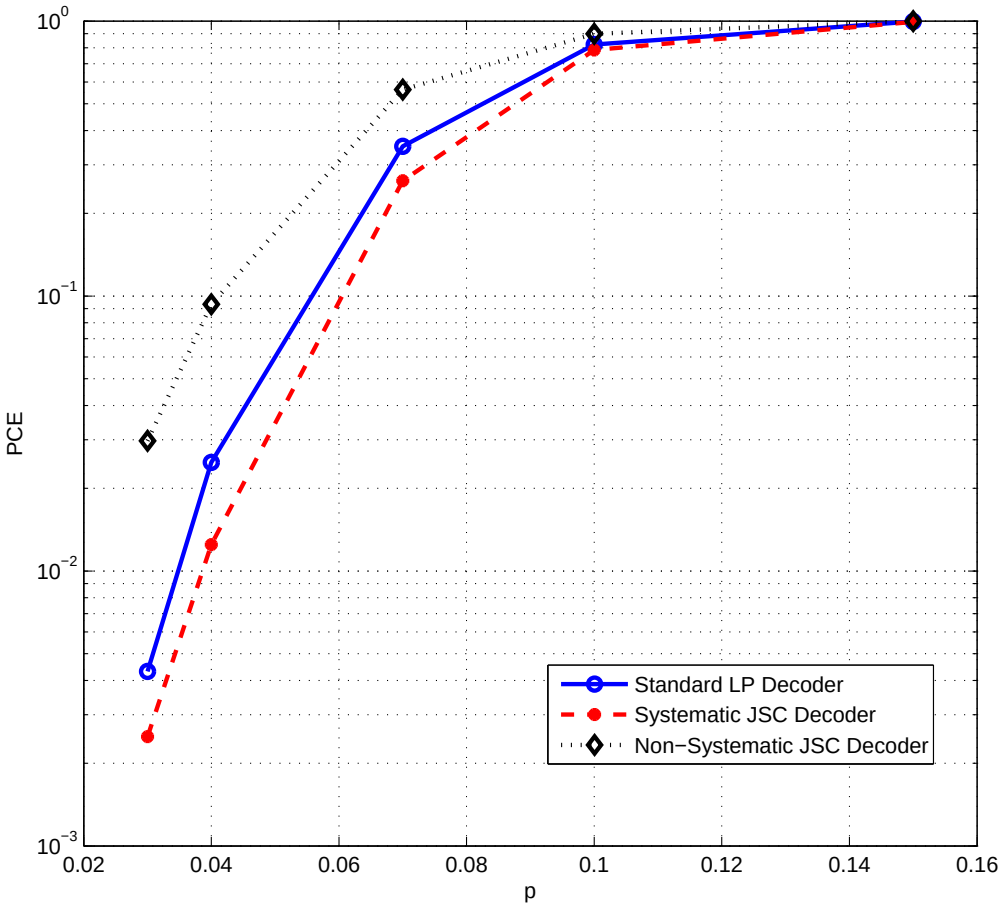

Figure 4.3: Standard LP Decoder vs. Modified LP Decoders for Non-Uniform Source $n = 200$, $p_1 = 0.7$, $R = \frac{1}{2}$ over the BSC.

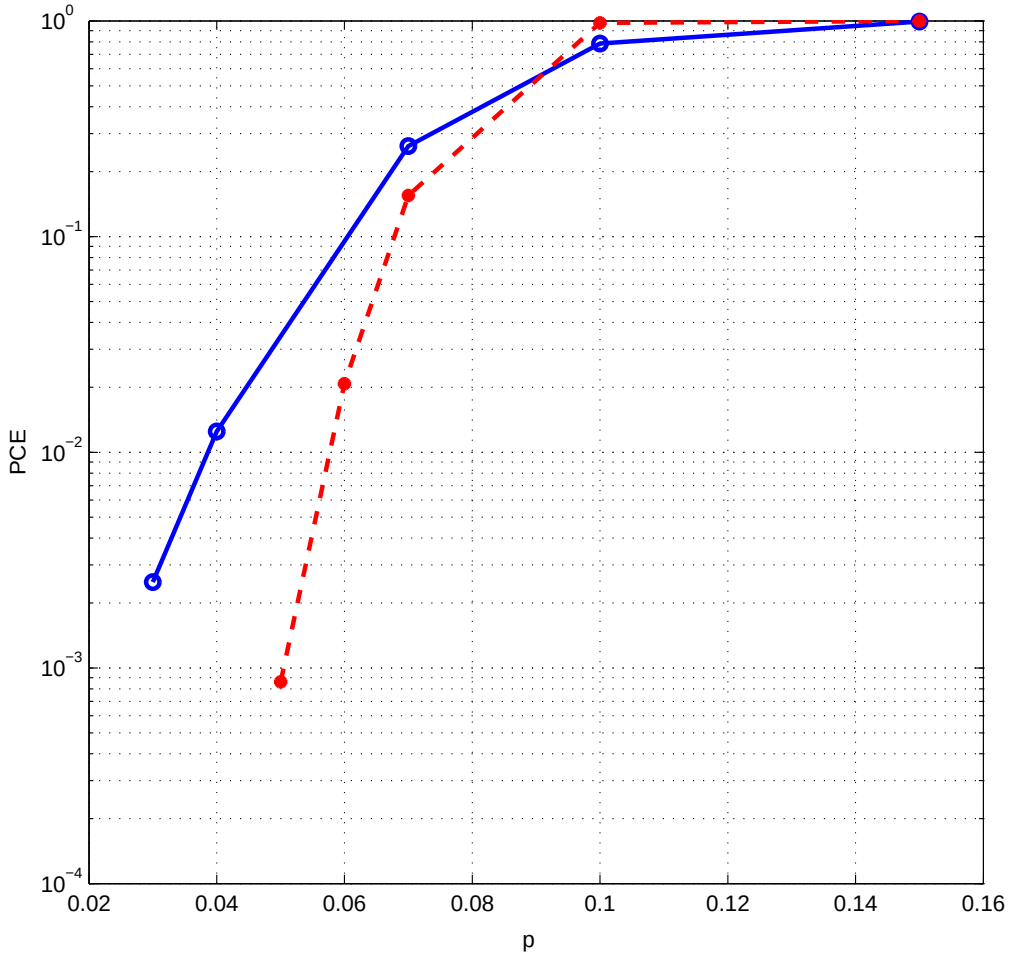

Figure 4.4: Systematic $(200, 100)$ LDPC code vs. Systematic $(1000, 500)$ LDPC code at $R = \frac{1}{2}$, $p_1 = 0.7$ over the BSC.

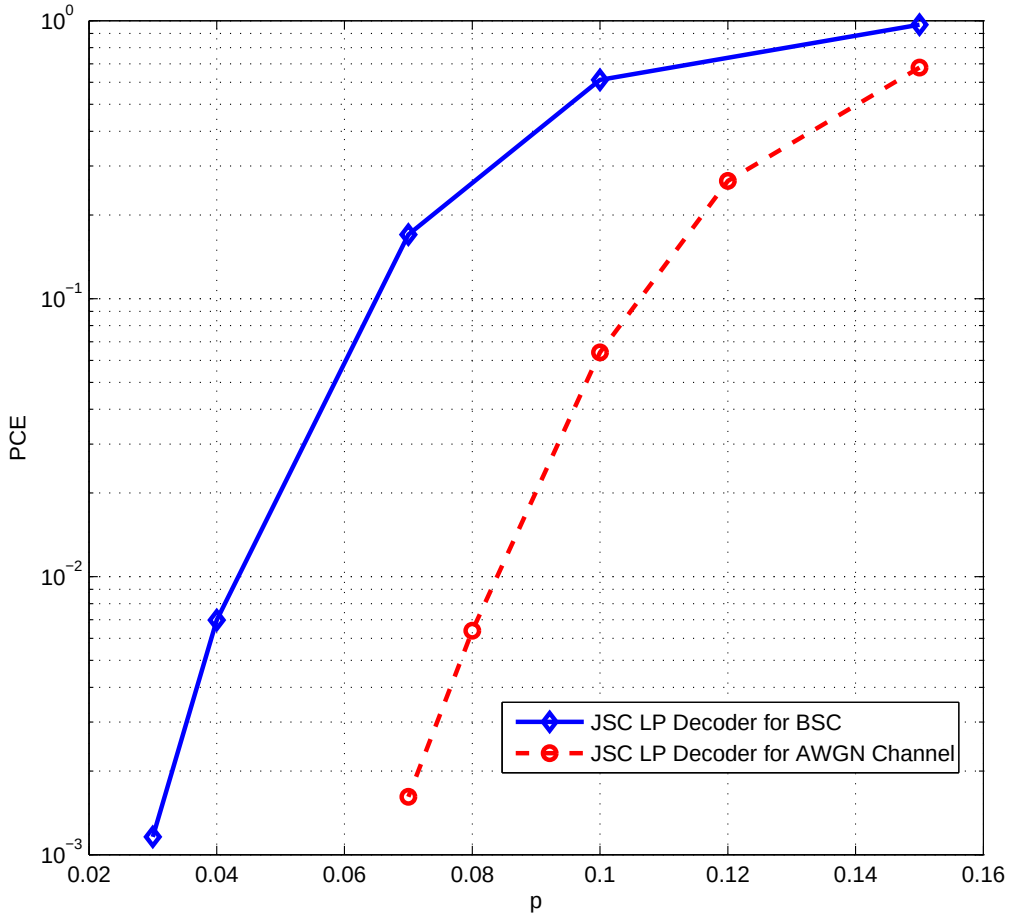

Figure 4.5: LP JSC decoders for the BSC and AWGN channel with systematic $(200, 100)$ LDPC code and $p_1 = 0.8$

Chapter 5

LP Decoding for Channels with Memory

Feldman's LP decoding formulation is designed for channels without memory. Here we attempt to extend this formulation to apply to channels with memory. More specifically, we explore the application of LP decoding techniques to two channels with memory: the first-order Markov noise channel and the infinite memory non-ergodic Polya channel. We succeed in developing a decoder only in the latter case; however, we do attain some insight into the feasibility of an LP decoder in the former case.

5.1 First-Order Markov Noise Channel

Since the decoding polytope is independent of the channel, when developing an LP formulation of decoding over the first-order Markov noise channel we only need to develop a new linear cost function. The most obvious approach is to attempt to linearize the ML decoding metric for the Markov channel, as done in the previous chapter for the case of non-uniform sources. Here we attempt to do just that, using similar manipulations. We let $P[y_i|c_i, c_{i-1}, y_{i-1}]$ be the probability that y_i was received at time i given that c_i and c_{i-1} were transmitted at time i and $i-1$, respectively, and that y_{i-1} was received at time $i-1$. Further, we allow any or all of c_i, c_{i-1} and y_{i-1} to be replaced by constants keeping the same meaning, *e.g.*, $P[y_i|0,1,1]$ is the probability that y_i was received at time i given that 0 and 1 were transmitted at times i and $i-1$, respectively, and that 1 was received at time $i-1$:

$$\begin{aligned}
\underline{\hat{y}} &= \underset{\underline{c}\in C}{\operatorname{argmax}}\ P[\underline{y}|\underline{c}] \\
&= \underset{\underline{c}\in C}{\operatorname{argmax}}\ P[y_1|c_1]\prod_{i=2}^{n} P[y_i|c_i, c_{i-1}, y_{i-1}] \\
&= \underset{\underline{c}\in C}{\operatorname{argmax}}\ \underbrace{\log\left(P[y_1|c_1]\right)}_{T1} + \underbrace{\sum_{i=2}^{n} \log\left(P[y_i|c_i, c_{i-1}, y_{i-1}]\right)}_{T2}.
\end{aligned}$$

Now, we expand the terms $T1$ and $T2$, keeping in mind that these terms take only binary values for the c_i variables. First, we note that $T1$ can be expressed in the

following way (one only needs to verify that this holds when $c_1 = 0$ and when $c_1 = 1$):

$$\begin{aligned} T1 &= \log\left(P[y_1|1]\right) c_1 - \log\left(P[y_1|0]\right) c_1 + \log\left(P[y_1|0]\right)) \\ &= \log\left(\frac{P[y_1|1]}{P[y_1|0]}\right) c_1 + \text{const}(y_1), \end{aligned}$$

where $\text{const}(y_1)$ represents a constant dependent only on y_i, *i.e.*, not dependent on $\underline{c}$.

Next, we examine $T2$:

$$\begin{aligned} T2 &= \sum_{\substack{i:c_i=1 \\ 2\leq i\leq n}} \log\left(P[y_i|1, c_{i-1}, y_{i-1}]\right) c_i + \sum_{\substack{i:c_i=0 \\ 2\leq i\leq n}} \log\left(P[y_i|0, c_{i-1}, y_{i-1}]\right) \\ &= \underbrace{\sum_{i=2}^{n} \log\left(P[y_i|1, c_{i-1}, y_{i-1}]\right) c_i}_{T3} \underbrace{- \sum_{\substack{i:c_i=1 \\ 2\leq i\leq n}} \log\left(P[y_i|0, c_{i-1}, y_{i-1}]\right) c_i}_{T4} \\ &\quad + \underbrace{\sum_{i=2}^{n} \log\left(P[y_i|0, c_{i-1}, y_{i-1}]\right)}_{T5} \end{aligned}$$

We now expand $T3$, $T4$, $T5$, the subterms of $T2$ labelled above, starting with $T3$:

$$\begin{aligned} T3 &= \sum_{\substack{i:c_{i-1}=1 \\ 2\leq i\leq n}} \log\left(P[y_i|1, 1, y_{i-1}]\right) c_i c_{i-1} \\ &\quad + \sum_{\substack{i:c_{i-1}=0 \\ 2\leq i\leq n}} \log\left(P[y_i|1, 0, y_{i-1}]\right) c_i \end{aligned}$$

$$
\begin{aligned}
&= \sum_{i=2}^{n} \log\left(P[y_i|1,1,y_{i-1}]\right) c_i c_{i-1} \\
&\quad + \sum_{i=2}^{n} \log\left(P[y_i|1,0,y_{i-1}]\right) c_i \\
&\quad - \sum_{i=2}^{n} \log\left(P[y_i|1,0,y_{i-1}]\right) c_i c_{i-1} \\
&= \sum_{i=2}^{n} \log\left(\frac{P[y_i|1,1,y_{i-1}]}{P[y_i|1,0,y_{i-1}]}\right) c_i c_{i-1} \\
&\quad + \sum_{i=2}^{n} \log\left(P[y_i|1,0,y_{i-1}]\right) c_i,
\end{aligned}
$$

Next, we consider $T4$:

$$
\begin{aligned}
T4 &= - \sum_{\substack{i:c_{i-1}=1 \\ 2\le i\le n}} \log\left(P[y_i|0,1,y_{i-1}]\right) c_i c_{i-1} \\
&\quad - \sum_{\substack{i:c_{i-1}=0 \\ 2\le i\le n}} \log\left(P[y_i|0,0,y_{i-1}]\right) c_i \\
&= - \sum_{i=2}^{n} \log\left(P[y_i|0,1,y_{i-1}]\right) c_i c_{i-1} \\
&\quad - \sum_{i=2}^{n} \log\left(P[y_i|0,0,y_{i-1}]\right) c_i \\
&\quad + \sum_{i=2}^{n} \log\left(P[y_i|0,0,y_{i-1}]\right) c_i c_{i-1} \\
&= \sum_{i=2}^{n} \log\left(\frac{P[y_i|0,0,y_{i-1}]}{P[y_i|0,1,y_{i-1}]}\right) c_i c_{i-1} \\
&\quad - \sum_{i=2}^{n} \log\left(P[y_i|0,0,y_{i-1}]\right) c_i,
\end{aligned}
$$

and, lastly, $T5$:

$$\begin{aligned}
T5 &= \sum_{\substack{i:c_{i-1}=1 \\ 2\le i\le n}} \log\left(P[y_i|0,1,y_{i-1}]\right) c_{i-1} \\
&\quad + \sum_{\substack{i:c_{i-1}=0 \\ 2\le i\le n}} \log\left(P[y_i|0,0,y_{i-1}]\right) \\
&= \sum_{i=2}^{n} \log\left(P[y_i|0,1,y_{i-1}]\right) c_{i-1} \\
&\quad + \sum_{i=2}^{n} \log\left(P[y_i|0,0,y_{i-1}]\right) \\
&\quad - \sum_{i=2}^{n} \log\left(P[y_i|0,0,y_{i-1}]\right) c_{i-1} \\
&= \sum_{i=2}^{n} \log\left(\frac{P[y_i|0,1,y_{i-1}]}{P[y_i|0,0,y_{i-1}]}\right) c_{i-1} + \text{const}(\underline{y}),
\end{aligned}$$

where, again, const($\underline{y}$) is a term dependent on $\underline{y}$, but not on $\underline{c}$. Finally, gathering the terms $T1$, $T3$, $T4$, $T5$, and dropping the constant terms independent of $\underline{c}$, as they will have no effect on a maximization with respect to $\underline{c}$, we arrive at an expression for $\hat{\underline{y}}$:

$$\begin{aligned}
\hat{\underline{y}} &= \underset{\underline{c}\in C}{\text{argmax}}\; T1 + T3 + T4 + T5 \\
&= \underset{\underline{c}\in C}{\text{argmax}} \log\left(\frac{P[y_1|1]}{P[y_1|0]}\right) c_1 + \sum_{i=2}^{n} \log\left(\frac{P[y_i|1,0,y_{i-1}]}{P[y_i|0,0,y_{i-1}]}\right) c_i \\
&\quad + \sum_{i=2}^{n} \log\left(\frac{P[y_i|0,1,y_{i-1}]}{P[y_i|0,0,y_{i-1}]}\right) c_{i-1} \\
&\quad + \sum_{i=2}^{n} \log\left(\frac{P[y_i|0,0,y_{i-1}]P[y_i|1,1,y_{i-1}]}{P[y_i|0,1,y_{i-1}]P[y_i|1,0,y_{i-1}]}\right) c_i c_{i-1}.
\end{aligned}$$

The above expression for the ML cost function is non-linear in terms of the c_i variables, as a consequence of the last term, which contains quadratic components of the form $c_i c_{i-1}$. So, in order to formulate the decoding problem as an LP, we would need to find some linearization or linear approximation to the cost function. Another alternative would be to use codes with no adjacent 1's, which would zero-out the last term, leaving a linear cost function. Attempts to linearize the cost function were unanimously unsuccessful. In fact, in terms of error performance over the Markov channel, the memoryless cost function (3.2) performed best (in terms of codeword error probability) among all linear cost functions tested.

Since the cost function above is quadratic in form, the use of quadratic programming was considered. A quadratic program (QP) with linear constraints can be defined as follows

$$\begin{aligned} &minimize\ \frac{1}{2}\underline{x}^t Q\underline{x} + \underline{g}^t\underline{x} \\ &\text{constrained by } A\underline{x} \leq \underline{b} \end{aligned} \tag{5.1}$$

where $\underline{g}$, $\underline{x}$, and $\underline{b}$ are (column) vectors in $\mathbb{R}^n$, Q and A are n-by-n real matrices, and Q is a symmetric n-by-n real matrix [43, Section 1.5]. If the matrix Q is positive semi-definite, then the problem is convex, and there exist algorithms to solve the QP efficiently (*i.e.,* in polynomial time). On the other hand, if the matrix Q is indefinite, then in general, the problem is NP-hard. We next examine the nature of the matrix Q in this instance. It should be noted here that we have formulated the problem as a

maximization; however, a QP in the standard form (5.1) is posed as a minimization. This can be overcome by simply considering the problem of minimizing the negative of the function derived above, and, as we will see shortly, this has no effect on the analysis which follows.

Choosing an arbitrary pair of parameters, α and β, for the first order Markov noise channel, it can easily be verified that the quadratic terms of (5.1) take on a value of either positive or negative L, where

$$L = \log\left(\frac{(1-\alpha)(1-\beta)}{\beta\alpha}\right).$$

More precisely,

$$\log\left(\frac{P[y_i|0,0,y_{i-1}]P[y_i|1,1,y_{i-1}]}{P[y_i|0,1,y_{i-1}]P[y_i|1,0,y_{i-1}]}\right) = \begin{cases} L & \text{for } y_i = y_{i-1} \\ -L & \text{for } y_i \neq y_{i-1} \end{cases}.$$

From the above, it follows by a simple expansion that if a QP were implemented according to the expression (5.1), then the symmetric matrix Q would have the following form

$$Q = L\begin{bmatrix} 0 & \pm 1 & 0 & \dots & 0 \\ \pm 1 & 0 & \pm 1 & & 0 \\ 0 & \pm 1 & \ddots & \ddots & 0 \\ \vdots & & \ddots & 0 & \pm 1 \\ 0 & 0 & \dots & \pm 1 & 0 \end{bmatrix},$$

where each off-diagonal pair of ± 1's are equal and fixed to either 1 or -1 depending on the received vector $\underline{y}$.

Now, it is not hard to show that the matrix Q is indefinite (*i.e.* it has at least one negative and one positive eigenvalue). In order to do so, we first must define what is meant by a *principal minor* of a matrix. A *minor* of a matrix A is the determinant of a square matrix obtained from A by removing some number of rows and columns. If the square matrix is obtained from A such that the rows and columns removed are of corresponding indicies, then the determinant of that matrix is a principal minor of A (*e.g.*, rows 2 and 4 and columns 2 and 4 are removed from A) [44, pg. 2]. Now, we can use the Sylvester criterion for positive semi-definiteness in order to show that Q is indefinite. This criterion states that a matrix A is positive semi-definite if and only if all principal minors of the matrix are non-negative [44, pg. 307]. Now, if we consider the the upper-left 2×2 square submatrix of Q, call it B, then B has one of the following two forms, as Q is symmetric:

$$B = L \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \quad \text{or} \quad L \begin{bmatrix} 0 & -1 \\ -1 & 0 \end{bmatrix}.$$

In both cases, the determinant of B is $-L^2$, and hence Q has a negative principal minor and is not positive semi-definite by the Sylvester criterion. The same argument shows that Q is not negative semi-definite, since if that were the case, $-Q$ would be positive semi-definite, which cannot be the case as the possible upper-left 2×2 square

submatrices of $-Q$ are the same as for Q. Therefore Q is indefinite, and we do not have any guarantee of an efficient algorithm to solve the QP of interest. This is not a particularly optimistic result; however, this does not reveal anything about the existence of any such efficient algorithm to solve this optimization. To be precise, this states that the problem of optimizing the quadratic function we are interested in belongs to a class of optimization problems that is NP-hard.

As a future direction, it would be worth investigating whether or not any stronger statements could be made about the existence of an efficient algorithm to solve this problem.

5.2 Polya Channel

As discussed in Section 2.2.5, the ML decoding rule for the infinite memory non-ergodic Polya channel with bit error rate ρ and correlation parameter δ relies on both minimum- and maximum-distance decoding. As shown in [9], assuming the binary n-vector $\underline{y}$ is received, it is possible to formulate MDD using an LP:

$$\underline{\hat{y}} = \underset{\underline{c} \in C}{\operatorname{argmin}} \ \underline{\gamma'} \cdot \underline{c} \ , \text{ where}$$

$$\gamma'_i = \begin{cases} -1 & \text{if } y_i = 1 \\ 1 & \text{if } y_i = 0 \end{cases}$$

Using this assignment of γ', we have that

$$\gamma' \cdot \underline{c} \;\; = \;\; d(\underline{c}, \underline{y}) - \omega(\underline{y})$$

where $\omega(\underline{y})$ represents the Hamming weight of the vector $\underline{y}$. Thus,

$$\underset{\underline{c} \in C}{\operatorname{argmin}} \; \gamma' \cdot \underline{c} \;\; = \;\; \underset{\underline{c} \in C}{\operatorname{argmin}} \; d(\underline{c}, \underline{y})$$

since the term $\omega(\underline{y})$ does not depend on $\underline{c}$, and hence can be left out of the minimization without consequence. Therefore, the LP formulation using $\underline{\gamma}'$ as the cost function and codeword polytope $\mathcal{P}(C)$ as the region of optimization is equivalent to MDD. Similarly, we can attain a formulation for maximum-distance decoding using $\underline{\gamma}'' = -\underline{\gamma}'$ as the cost function.

Having linear cost functions corresponding to minimum and maximum distance decoding, we can formulate a relaxed LP decoder corresponding to ML decoding for the Polya channel. First, we select an (n, k) LDPC code, C, and $H \subset C^{\perp}$ corresponding to the rows of a parity-check matrix of C, and construct its relaxed codeword polytope, $\mathcal{Q}(H)$. Then, relaxed minimum- and maximum-distance decoding can be expressed as LPs by selecting $\underline{y}_{min}$ and $\underline{y}_{max}$ to be in the sets

$$\underset{\underline{x} \in \mathcal{Q}(H)}{\operatorname{argmin}} \; \underline{\gamma}' \cdot \underline{x} \quad \text{and} \quad \underset{\underline{x} \in \mathcal{Q}(H)}{\operatorname{argmin}} \; \underline{\gamma}'' \cdot \underline{x},$$

respectively.

We solve the LP's corresponding to minimum- and maximum-distance decoding and get the minimum-distance solution, $\hat{\underline{y}}_{min}$, and the maximum-distance solution $\hat{\underline{y}}_{max}$. These solutions are rounded in order to obtain the nearest codeword, and are then used to compute d_{min} and d_{max}. The decoded word is then taken as $\hat{\underline{y}}_{min}$ or $\hat{\underline{y}}_{max}$ according to the conditions (2.2) and (2.3). We will refer to this LP decoder as the relaxed minimum/maximum-distance decoder (MMDD). It is important to note that this is not ML decoding, as we do not know if $\hat{\underline{y}}_{min}$ and $\hat{\underline{y}}_{max}$ represent the true minimum and maximum distance codewords, and hence whether the computed d_{min} and d_{max} are correct.

This decoder has the following ML certificate property: if both the minimum and maximum distance LPs yield integral solutions, then the final solution, whether $\hat{\underline{y}}_{min}$ or $\hat{\underline{y}}_{max}$, is guaranteed to be the ML solution. We have this property because if both $\hat{\underline{y}}_{min}$ and $\hat{\underline{y}}_{max}$ are integral, then, by previous arguments, we know that they represent the true minimum and maximum distance codewords, and since the decoder in [2] is the true ML decoder, we know that conditions (2.2) and (2.3) will yield the ML codeword.

5.3 Simplified ML Decoding for the Polya Channel

Here we present a simplification of the ML decoding rule for certain codes transmitted over the Polya channel.

Lemma 1. *For a linear code containing the all-ones codeword, if $\rho \leq 0.5$ then ML decoding over the Polya channel reduces to minimum Hamming distance decoding. Otherwise, if $\rho > 0.5$, ML decoding reduces to maximum Hamming distance decoding.*

Proof. Suppose that we have a binary linear (n, k) code, C, containing the all-ones word, $\underline{1}_n$. Then we have that

1. For a received word $\underline{y} \in \{0,1\}^n$, and $\underline{c} \in C$ s.t $d(\underline{y}, \underline{c}) = t$, there exists $\underline{c}' \in C$ s.t. $d(\underline{y}, \underline{c}') = n - t$, since we can just take $\underline{c}'$ to be $\underline{c} \oplus \underline{1}_n$, which is in the code by closure.

2. The above implies that for $\underline{y} \in \{0,1\}^n$, $d_{min}(\underline{y}) = n - d_{max}(\underline{y})$, where

$$d_{min}(\underline{y}) \doteq \min_{\underline{c} \in C} d(\underline{c}, \underline{y}) \quad \text{and} \quad d_{max}(\underline{y}) \doteq \max_{\underline{c} \in C} d(\underline{c}, \underline{y}).$$

   To see this, consider $\underline{c}_{min}$ achieving $d_{min}(\underline{y})$. From above, we have that $d_{max}(\underline{y})$ is at least $n - d_{min}(\underline{y})$, as $d(\underline{1}_n \oplus \underline{c}_{min}, \underline{y}) = n - d_{min}(\underline{y})$. Conversely, by similar argument, $d_{min}(\underline{y})$ is at least as small as $n - d_{max}(\underline{y})$. So $d_{max}(\underline{y}) \geq n - d_{min}(\underline{y})$ and $d_{min}(\underline{y}) \leq n - d_{max}(\underline{y}) \Rightarrow d_{max}(\underline{y}) \leq n - d_{min}(\underline{y})$, and so $d_{min}(\underline{y}) = n - d_{max}(\underline{y})$.

Note that since $d_{min}(\underline{y}) = n - d_{max}(\underline{y})$, we have that $d_{min}(\underline{y}) \leq \frac{n}{2}$, since $d_{min}(\underline{y}) \leq d_{max}(\underline{y})$. Now, suppose that $\rho < 0.5$. It follows that

$$d_0 \;=\; \frac{n}{2} + \frac{1 - 2\rho}{2\delta} > \frac{n}{2};$$

in other words,

$$d_0 = \frac{n}{2} + \epsilon, \quad \epsilon > 0.$$

Now, we consider condition (2.2),

$$\begin{aligned} \left| d_{max}(\underline{y}) - d_0 \right| &= \left| n - d_{min}(\underline{y}) - \frac{n}{2} - \epsilon \right| \\ &= \left| (\frac{n}{2} - d_{min}(\underline{y})) - \epsilon \right|. \end{aligned}$$

We also have that

$$\left| d_{min}(\underline{y}) - d_0 \right| = \left| -(\frac{n}{2} - d_{min}(\underline{y})) - \epsilon \right|;$$

but $\frac{n}{2} - d_{min}(\underline{y}) \geq 0$ as $d_{min}(\underline{y}) \leq \frac{n}{2}$. Thus, it follows that condition (2.2) is satisfied, and hence MDD is optimal.

So, if a binary linear code has the all-ones word, and if $\rho < 0.5$, ML decoding is equivalent to MDD. By a symmetric argument, we can show that if $\rho > 0.5$, then ML decoding is equivalent to maximum distance decoding. □

The above lemma allows us to formulate ML decoding in a more straightforward manner as long as we are dealing with a linear code containing the all-ones codeword. Fortunately, this class of codes is of interest, including the class of regular LDPC codes with even row-degree.

Therefore, we can implement relaxed ML decoding for the non-ergodic Polya channel with a single LP when using a code with the all-ones codeword and $\rho < 0.5$. First, we select an (n, k) LDPC code C containing the all-ones codeword and $H \subset C^{\perp}$ corresponding to the rows of a parity-check matrix of C, and construct its relaxed codeword polytope, $\mathcal{Q}(H)$. Then, relaxed ML decoding for the Polya channel can be implemented as relaxed MDD by selecting the decoded vector $\underline{\hat{y}}$

$$\underset{\underline{x} \in \mathcal{Q}(H)}{\operatorname{argmin}} \underline{\gamma}' \cdot \underline{x}. \tag{5.2}$$

In this simplified scenario, we have a straight-forward ML certificate property, that is, if the LP yields an integral solution, then it is guaranteed to be the ML codeword.

5.4 Simulation Results

Since there are no known simulation results for decoding over the infinite-memory Polya channel, it is natural for us to compare simulation results to the theoretical performance limit, which is the channel ϵ-capacity (2.4) defined in Section 2.2.5. We note that since the infinite-memory Polya channel is a (non-ergodic) averaged channel with BSC components governed by the Beta distribution, its channel capacity is zero; however, its ϵ-capacity is strictly positive for $\epsilon > 0$ and strictly decreasing to zero as $\epsilon \searrow 0$ [2].

A closed-form expression for the ϵ-capacity, C_{ϵ}, of the infinite-memory Polya chan-

nel is given in [2, Equation (8)] as a function of ϵ, ρ and δ, as has been discussed in Section 2.2.5. For the purpose of comparison, given ρ, δ and rate R', we can determine the ϵ for which R' is the maximum achievable rate (i.e., the ϵ-capacity) by solving $C_\epsilon = R'$ over ϵ. This value of ϵ is thus a theoretical lower bound on the PCE of a rate-R' code over the infinite-memory Polya channel with parameters ρ and δ.

One issue that needs to be considered is how one should simulate communications over a non-ergodic channel. Simulating over a single instance[1] of the channel is certainly not correct, as this would demonstrate only one of many possible behaviours of the channel. Instead, results were obtained by repeatedly initializing the channel (setting the urn to the initial conditions) and transmitting a fixed number of codewords in each instance. This allowed for the overall (Beta) distribution of possible channel outcomes to be explored. For example, for the MMDD curve in Figure 5.3, 100 blocks (with each block consisting of 200 bits) were transmitted for each instance of the channel, and this was repeated between 8000 and 90000 times, depending on ρ.

In Figures 5.1, 5.2 and 5.3, simulation results are shown for relaxed MDD and relaxed MMDD using LDPC codes with and without the all-ones codeword, respectively. For the MDD decoder in these figures, a $(200, 100)$ $(3, 6)$-regular LDPC code was used. This code clearly contains the all-ones codeword, as the row-degree of every row is even (it is equal to 6). For the MMDD decoder in these figures, an irregular

[1] "Single instance" here means a single realization of the infinite-memory non-ergodic Polya noise process.

$(200, 100)$ LDPC code with a constant column-degree of 3 was used. This code was verified not to contain the all-ones codeword. Simulations were performed over a range of values of the channel BER ρ. In each figure, we include the appropriate curve for the channel ϵ-capacity, as discussed above. We also include in each figure the regular LDPC code's performance over the BSC (*i.e.*, when $\delta = 0$), corresponding to the situation where an ideal (infinite-depth) interleaver is applied to the channel.

In studying Figures 5.1, 5.2 and 5.3, the following trends can be observed. Firstly, for all codes and decoders, we see an absolute improvement in error performance with increasing δ. Secondly, as δ increases, we see a relative improvement in error performance of the relaxed MMDD with the irregular LDPC code as compared to the relaxed MDD with the regular LDPC code. Tables 5.1 and 5.2 summarize the performance of the relaxed MDD and MMDD decoders relative to the ϵ-capacity in terms of the percentage difference of ρ at which a certain fixed PCE is obtained. Results are computed using a PCE of 10^{-2} and 10^{-3} (if data is available at this error level).

Table 5.1: Relaxed MDD performance for Polya channel

δ	10^{-2}
2	33%
4	51%
10	74%

In Figure 5.4, we demonstrate that the gains associated with the relaxed MMDD decoder are not simply a result of the irregular code. Instead, this figure shows that

Table 5.2: Relaxed MMDD performance for Polya channel

δ	10^{-2}	10^{-3}
2	20%	NA
4	35%	33%
10	16%	13%

the gains are a result of the combination of using the more complex decoding rule in conjunction with an irregular code that can exploit it, since it does not contain the all-ones codeword. This is demonstrated by comparing results of the irregular LDPC code being decoded using relaxed MDD and relaxed MMDD. We also include the appropriate ϵ-capacity curve and the performance of the regular LDPC code under relaxed MDD for reference. In this figure, $\delta = 10$ is used as it demonstrates the most extreme difference between the MDD and MMDD decoders. Another interesting point to note is that the regular LDPC code and the irregular LDPC code appear to perform nearly identically under MDD.

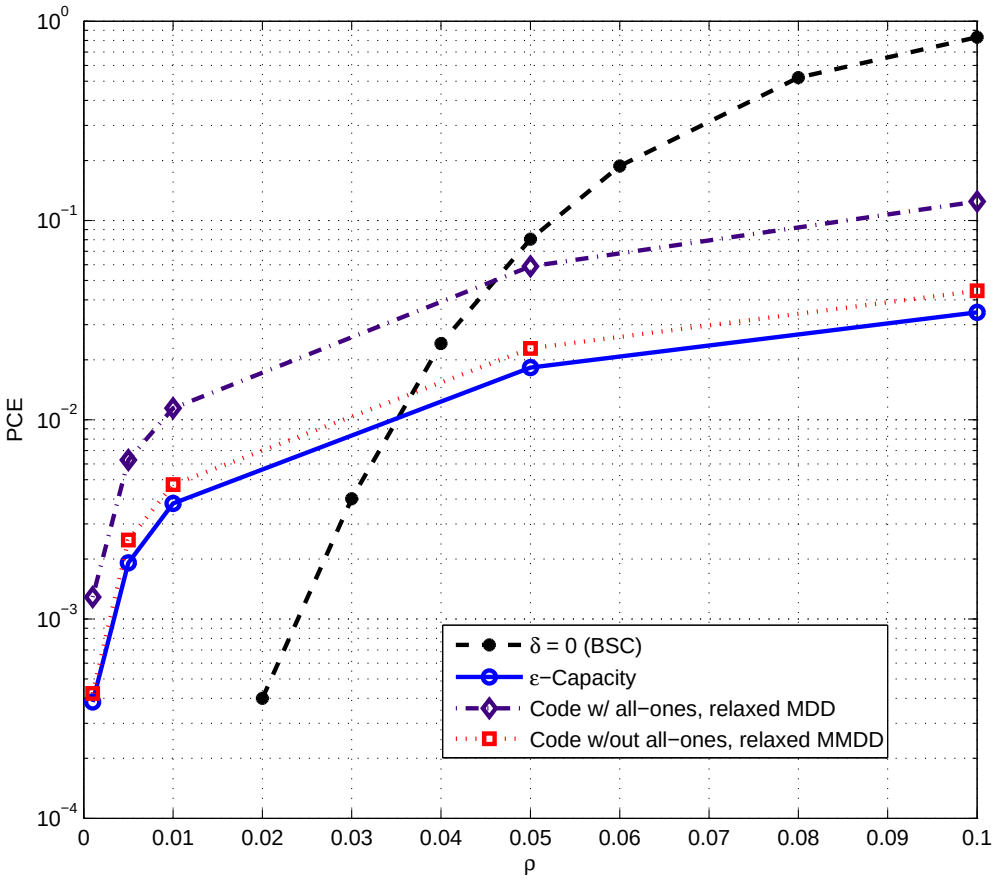

Figure 5.1: $\delta = 10$: $(200, 100)$ $(3, 6)$-regular LDPC code under relaxed MDD decoding and irregular $(200, 100)$ LDPC code under relaxed MMDD. Curves representing the ϵ-capacity and the case of ideal interleaving (BSC) are also included for comparison.

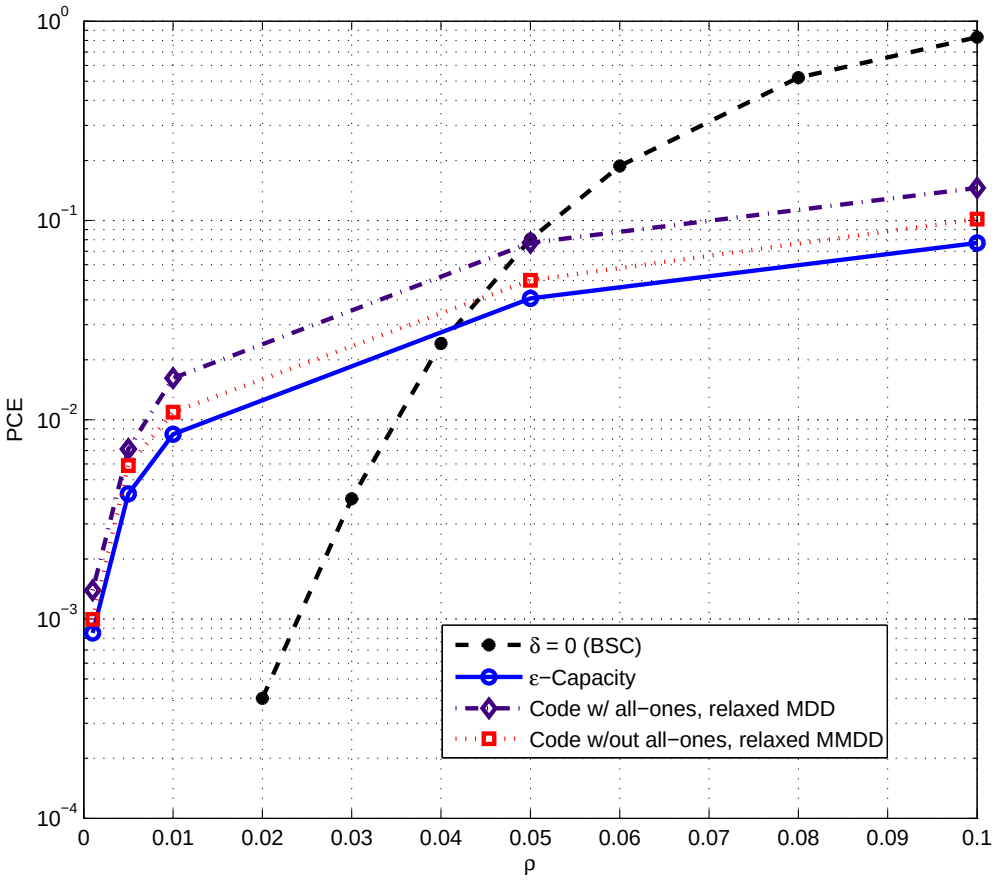

Figure 5.2: $\delta = 4$: $(200, 100)$ $(3, 6)$-regular LDPC code under relaxed MDD decoding and irregular $(200, 100)$ LDPC code under relaxed MMDD. Curves representing the ϵ-capacity and the case of ideal interleaving (BSC) are also included for comparison.

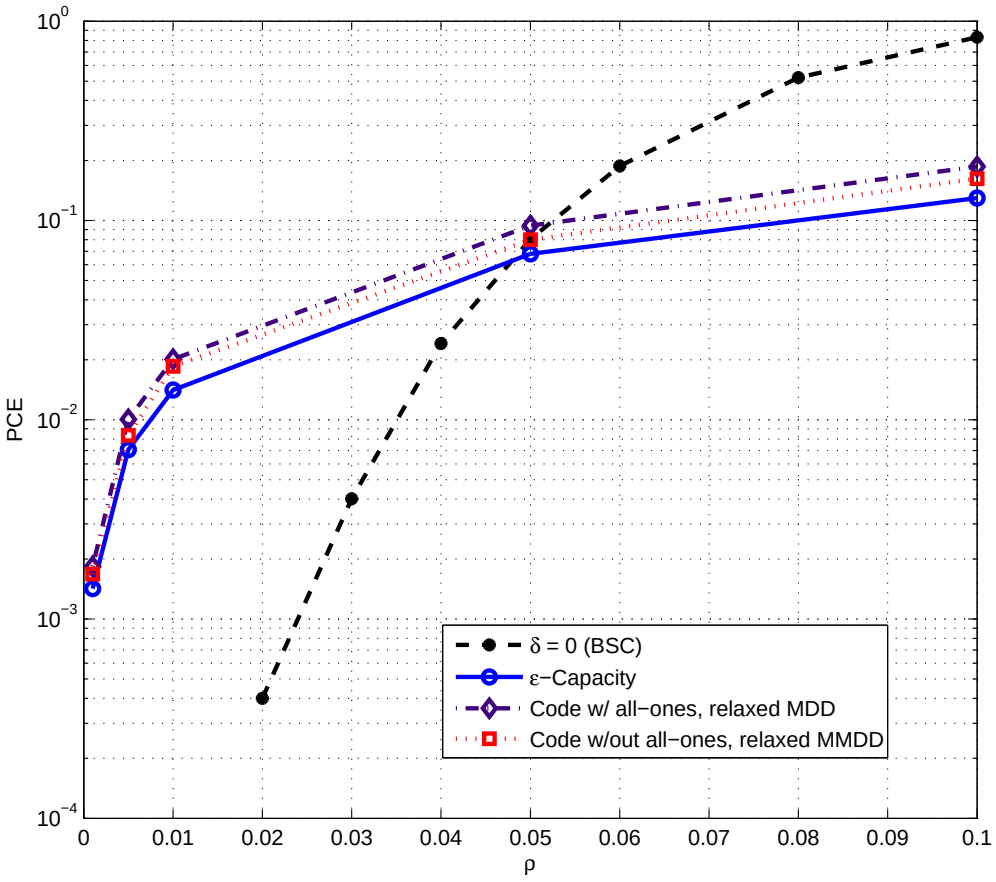

Figure 5.3: $\delta = 2$: $(200, 100)$ $(3, 6)$-regular LDPC code under relaxed MDD decoding and irregular $(200, 100)$ LDPC code under relaxed MMDD. Curves representing the ϵ-capacity and the case of ideal interleaving (BSC) are also included for comparison.

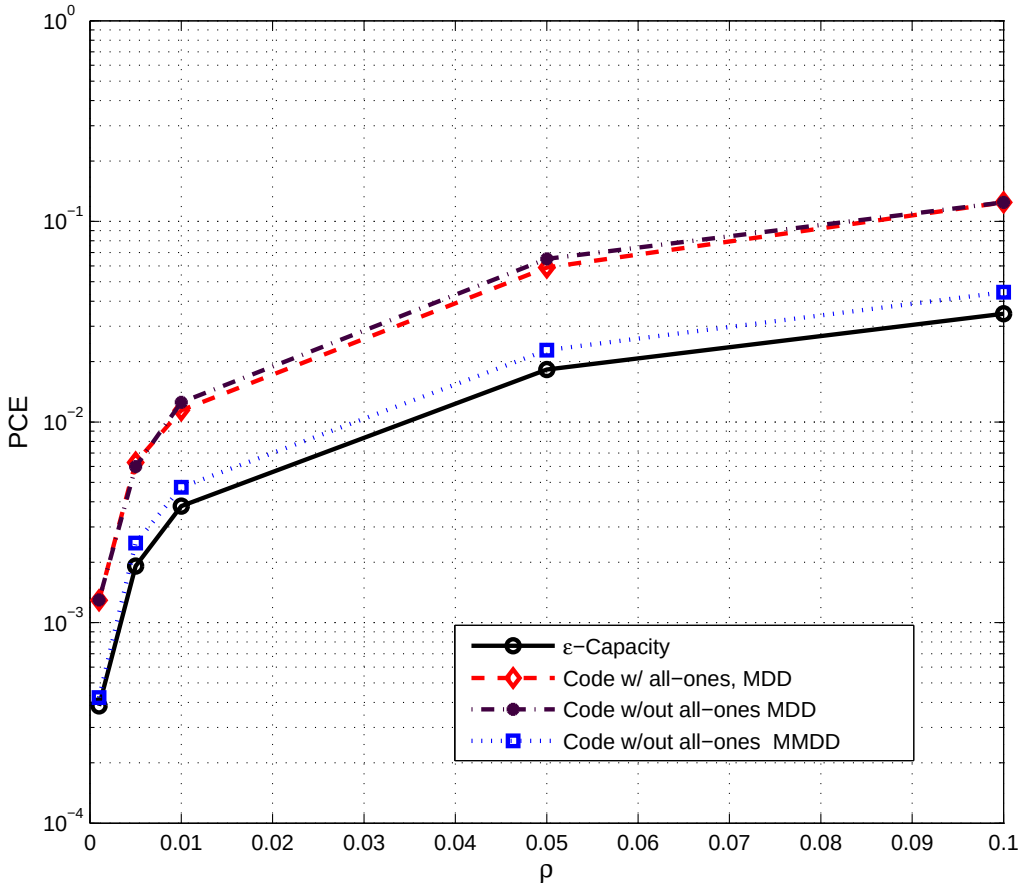

Figure 5.4: $\delta = 10$: $(200, 100)$ $(3, 6)$-regular LDPC code and irregular $(200, 100)$ LDPC code under relaxed MDD and irregular $(200, 100)$ LDPC code under relaxed MMDD. Curve representing ϵ-capacity is also included.

Chapter 6

Conclusions and Future Work

We have developed two LP decoders for the scenario of non-uniformity at the source. One decoder uses systematic codes, and the other uses punctured systematic codes and an extended polytope decoder. The former performs better for low values of the source skew parameter p_1 and the latter for high values of p_1 ($>\sim 0.9$). These decoders were developed primarily by modifying the linear cost function to incorporate the *a-priori* codeword probabilities. Performance gains were observed for both decoders as compared to the standard LP decoder. The gains observed increase with increasing skew at the source, that is, with increasing p_1, $p_1 > 0.5$.

An interesting direction for future work in this area would be the development of an LP decoder for non-systematic LDPC codes which does not rely on puncturing and an extended polytope. Because puncturing is detrimental to performance, the proposed extended polytope decoder only proves useful for the highest values of p_1, since at

this level of skew, the advantage of transmitting a non-systematic code is significant enough to overcome the losses due to puncturing. Additionally, the extended polytope increases the decoding complexity. Thus, finding an alternate approach to using LP decoding with non-systematic codes is well-motivated. It should be pointed out that this was attempted unsuccessfully, as it is not clear how to incorporate the *a-priori* codeword information as a linear function of the codeword bits when the code is non-systematic.

LP decoding for the additive Markov noise channel was considered; however, a seemingly inherent non-linearity in the ML cost function was discovered. This non-linearity acted as a major barrier to the application of LP decoding for the Markov channel. Quadratic programming was also considered, again without success. The quadratic cost function derived herein could potentially serve as a starting point for further investigation into an optimization approach of decoding for the Markov noise channel.

Finally, LP decoding was considered for the infinite-memory non-ergodic Polya contagion channel. Because the ML decoding rule for the Polya channel is based on minimum- and maximum-distance decoding, it was possible to represent a relaxed version of this rule into an LP-based decoder. Further, it was shown in this work that when using a code with the all-ones codeword, and when the channel bit error rate satisfies $\rho < 0.5$, ML decoding for the Polya channel reduces to MDD, and hence can be implemented as a single LP. Simulation results for the Polya channel demonstrate

relatively good performance with respect to the channel ϵ-capacity, especially when using relaxed MMDD with irregular codes.

As mentioned in Section 2.2.5, the infinite-memory Polya channel provides an interesting tool for modeling non-ergodic fading channels. As such, the Lemma regarding the equivalence of MDD to ML decoding for the Polya channel is of interest beyond the scope of LP decoding. It was shown (see Appendix B) that message-passing decoders can be applied to the Polya channel for LDPC codes with the all-ones codeword. These decoders perform as well as the relaxed minimum distance LP decoder presented in this work for codes with the all-ones codeword. Thus, the use of the Polya channel as a model is further motivated by the existence of efficient iterative decoders which perform well on codes with the all-ones codeword. Further investigation into the application of iterative decoders to the channel is warranted.

Bibliography

[1] J. Feldman, M. J. Wainwright, and D. R. Karger, "Using linear programming to decode binary linear codes," *IEEE Trans. Inform. Theory*, vol. 51, no. 3, pp. 954–972, 2005.

[2] F. Alajaji and T. Fuja, "A communication channel modeled on contagion," *IEEE Trans. Inform. Theory*, vol. 60, no. 6, pp. 2035–2041, 1994.

[3] C. E. Shannon, "A mathematical theory of communication," *Bell System Technical Journal*, vol. 27, pp. 379–423, July, October 1948.

[4] T. M. Cover and J. A. Thomas, *Elements of Information Theory*. Wiley, 1991.

[5] R. G. Gallager, *Information Theory and Reliable Communication*. Wiley, 1968.

[6] ——, "Low density parity check codes," Ph.D. dissertation, Massachusetts Institute of Technology, 1960.

[7] D. J. C. MacKay, "Good error-correcting codes based on very sparse matrices," *IEEE Trans. Inform. Theory*, vol. 45, no. 2, pp. 399–431, 1999.

[8] D. J. C. MacKay and R. M. Neal, "Near Shannon limit performance of low density parity check codes," *Electronics Letters*, vol. 32, pp. 1645–1646, 1996.

[9] J. Feldman, "Decoding error-correcting codes via linear programming," Ph.D. dissertation, Massachusetts Institute of Technology, 2003.

[10] G. I. Shamir and J. Boutros, "Non-systematic low-density parity-check codes for nonuniform sources," in *Proc. IEEE Int. Symp. Inform. Theory*, Adelaide, Australia, 2005, pp. 1898–1902.

[11] A. Alloum, J. Boutros, G. I. Shamir, and L. Wang, "Non-systematic LDPC codes for redundant data," in *Proc. 43rd Allerton Conf. Commun., Control, Computing*, Monticello, USA, 2006, pp. 1879–1888.

[12] C. Nicola, F. Alajaji, and T. Linder, "Decoding LDPC codes over binary channels with additive Markov noise," in *Proc. Canadian Workshop Inform. Theory*, Montreal, Canada, June 2005, pp. 187–190.

[13] A. Eckford, F.Kschischang, and S.Pasupathy, "Analysis of low-density parity-check codes for the Gilbert-Elliot channel," *IEEE Trans. Inform. Theory*, vol. 51, pp. 3872–3889, November 2005.

[14] J. Garcia-Frias, "Decoding of low-density parity-check codes over finite-state binary Markov channels," *IEEE Trans. Commun.*, vol. 52, pp. 1840–1843, November 2004.

[15] P. O. Vontobel and R. Koetter, "On the relationship between linear programming decoding and min-sum algorithm decoding," in *Proc. ISITA*, Parma, Italy, October 2004.

[16] J. Feldman and D. R. Karger, "Decoding turbo-like codes via linear programming," *Journal of Computer and System Sciences*, vol. 68, pp. 733–752, June 2002.

[17] N. Kashyap, "A decomposition theory for binary linear codes," *IEEE Trans. Inform. Theory*, vol. 54, no. 7, pp. 3035–3058, July 2008.

[18] M. F. Flanagan, V. Skachek, E. Byrne, and M. Greferath, "Linear-programming decoding of non-binary linear codes," *arXiv*, vol. abs/0707.4360, 2007.

[19] J. Feldman, T. Malkin, R. Servedio, C. Stein, and M. Wainwright, "LP decoding corrects a constant fraction of errors," in *Proc. IEEE Int. Symp. Inform. Theory*, Chicago, USA, 2004, p. 69.

[20] C. Daskalakis, A. G. Dimakis, R. M. Karp, and M. J. Wainwright, "Probabilistic analysis of linear programming decoding," in *SODA '07: Proceedings of the eighteenth annual ACM-SIAM symp. on Discrete algorithms*, New Orleans, USA, 2007, pp. 385–394.

[21] R. Koetter and P. O. Vontobel, "On the block error probability of LP decoding of LDPC codes," in *Proc. Inaugural Workshop of the Center for Information Theory and its Applications*, San Diego, USA, February 2006.

[22] J. Feldman and C. Stein, "LP decoding achieves capacity," in *SODA '05: Proc. of the sixteenth annual ACM-SIAM symp. on Discrete algorithms*, Vancouver, Canada, 2005, pp. 460–469.

[23] M. H. Taghavi and P. Siegel, "Adaptive linear programming decoding," in *Proc. IEEE Int. Symp. Inform. Theory*, Seattle, USA, 2006, pp. 1374–1378.

[24] P. O. Vontobel and R. Koetter, "On low-complexity linear-programming decoding of LDPC codes," *Europ. Trans. on Telecomm.*, vol. 5, pp. 509–517, April 2007.

[25] D. Burshtein, "Iterative approximate linear programming decoding of LDPC codes with linear complexity," in *Proc. IEEE Int. Symp. Inform. Theory*, Toronto, Canada, July 2008, pp. 1498–1502.

[26] S. C. Draper, J. S. Yedidia, and Y. Wang, "ML decoding via mixed-integer adaptive linear programming," in *Proc. IEEE Int. Symp. Inform. Theory*, Nice, France, 2007, pp. 1656–1660.

[27] A. G. Dimakis and M. J. Wainwright, "Guessing facets: Polytope structure and improved LP decoder," in *Proc. IEEE Int. Symp. Inform. Theory*, Seattle, USA, July 2006, pp. 1369–1373.

[28] T. Wadayama, "Interior point decoding for linear vector channels based on convex optimization," in *Proc. IEEE Int. Symp. Inform. Theory*, Toronto, Canada, July 2008.

[29] P. Vontobel, "Pseudo-codewords," 2008, http://www.hpl.hp.com/personal/Pascal_Vontobel/pseudocodewords/papers/.

[30] A. Cohen, F. Alajaji, N. Kashyap, and G. Takahara, "LP decoding for joint source-channel codes and for the non-ergodic Polya channel," *IEEE Commun. Letters*, vol. 12, no. 9, pp. 678–680, September 2008.

[31] R. M. Roth, *Introduction to Coding Theory*. Cambridge University Press, 2006.

[32] F. MacWilliams and N. Sloane, *The Theory of Error-Correcting Codes*. North Holland, 1983.

[33] R. M. Neal, "Software for low density parity check codes," 2006, http://www.cs.toronto.edu/ radford/ftp/LDPC-2006-02-08/index.html.

[34] G. Polya, "Sur quelques points de la théorie de probabilités," *Ann. Inst. H. Poincaré*, vol. 1, pp. 117–161, 1931.

[35] M. Effros, A. Goldsmith, and Y. Liang, "Capacity definitions of general channels with receiver side information," in *Proc. IEEE Int. Symp. Inform. Theory*, Nice, France, June 2007, pp. 921–925.

[36] L. Zhong, F. Alajaji, and G. Takahara, "A model for correlated Rician fading channels based on a finite queue," *IEEE Trans. Veh. Technol.*, vol. 57, no. 1, pp. 79–89, January 2008.

[37] D. Gale, "Linear programming and the simplex method," *Notices of the American Mathematical Society*, vol. 54, no. 3, pp. 364–369, March 2007.

[38] A. Schrijver, *Theory of Linear and Integer Programming*. John Wiley & Sons, 1998.

[39] G. B. Dantzig, "Maximization of a linear function of variables subject to linear inequalities," *Activity Analysis of Production and Allocation*, pp. 339–347, 1951.

[40] L. G. Khachiyan, "A polynomial algorithm in linear programming," *Doklady Akademii Nauk SSSR*, vol. 244, pp. 1093–1096, 1979, (English translation: Soviet Math. Dokl. 20, 191–194).

[41] E. R. Berlekamp, R. J. McEliece, and H. C. A. van Tilborg, "On the inherent intractability of certain coding problems," *IEEE Trans. Inform. Theory*, vol. IT-24, pp. 348–386, 1978.

[42] G. C. Zhu, F. Alajaji, J. Bajcsy, and P. Mitran, "Transmission of nonuniform memoryless sources via nonsystematic turbo codes," *IEEE Trans. Commun.*, vol. 52, no. 8, pp. 1344–1354, 2004.

[43] G. M. Lee, N. N. Tam, and N. D. Yen, *Quadratic Programming and Affine Variational Inequalities*. New York, NY: Springer Science+Business Media, Inc., 2005.

[44] F. R. Gantmacher, *The Theory of Matrices*. New York, N.Y.: Chelsea Publishing Company, 1960, vol. 1.

[45] M. Berkelaar, K. Eikland, and P. Notebaert, "lp_solve 5.5," May 2006, `http://lpsolve.sourceforge.net/5.5/`.

Appendix A

Implementation Details

Simulations were completed by integrating three software packages: Matlab, Radford Neal's Software for Low Density Parity Check Codes [33], and the open source (Mixed-Integer) Linear Programming system "lp_solve" [45]. The following applies to all simulation results presented in the previous chapters.

Matlab was the primary development environment and was selected primarily for its relative ease when programming with mathematics. Although generally much slower than compiled code from a lower-level language such as C, the tasks performed within the Matlab scripts account for a very small percentage ($< 1\%$) of the actual computation time when running the simulations. Hence, using Matlab leaves little lost in terms of performance, and yields substantial gains in terms of ease of development. Specifically, the Matlab scripts are responsible for generating source and noise blocks according to the source and noise distributions, respectively, encoding

the source blocks, determining the error rate as well as serving as the main platform for execution. The primary task in terms of computational complexity associated with the simulations is solving the decoding LP. Consequently, it is necessary to make this component as efficient as possible.

It is possible to interface Matlab with "lp_solve" using a pair of dynamic linking libraries (DLLs). The main DLL, "lpsolve55.dll", contains the core functionality of the "lp_solve" software package. The driver "mxlpsolve.dll" serves as an interface between Matlab and "lpsolve55.dll." Since DLLs by nature represent compiled code, porting the LP tasks to them results in substantially more efficient (on the order of 100 times faster) LP solving when compared to using a built-in Matlab solver (Matlab scripts are not compiled, but interpreted on the fly).

In order to generate LDPC codes, the software of Neal was used [33]. Since a Windows system was used, the Linux emulator "Cygwin" was employed as an environment in which to compile and execute the LDPC code software. The software provides the ability to create LDPC codes flexibly according to the desired length, rate and row and column degree properties. For example, the following command creates a parity-check matrix corresponding to a pseudo-random, $(200, 100)$ $(3, 6)$-regular LDPC code:

make-ldpc [output file] 100 200 [random seed] evenboth 3

Here, [random seed] is used to initialize the pseudo-random number generator, '100' indicates the dimension of the dual code, '200' indicates the code-length, "evenboth"

indicates that the code (if possible) should have a constant row- and column-degrees, and '3' indicates the desired column-degree. Note that the row-degree is a function of the column-degree and the rate of the code, and hence only the column-degree needs to be specified. The next example demonstrates how to generate a pseudo-random $(200, 100)$ irregular LDPC code

make-ldpc [output file] 100 200 [random seed] evencol 3

Differing in this command is only "evencol 3", which indicates that the code should have a constant column degree of '3'[1]. With this command, the row degree is allowed to vary from row-to-row, and hence an irregular code is produced.

Finally, the Adaptive LP method, discussed in Chapter 3, was implemented to help increase the speed of LP solving. Algorithm 1 was implemented in C and compiled as a mex-file (a type of compiled code which is easily interfaced with Matlab) allowing it to be integrated with the existing code.

Figures 1 shows baseline simulations which were performed in order to test the correctness of the implementation. The simulation parameters were set up to correspond with Fig. 7 from [1]. The simulations using the implementation described herein correspond very well with the results in [1], thus verifying the implementation's correctness.

[1]A few columns may be assigned extra 1's so as to avoid rows with degree less than 2.

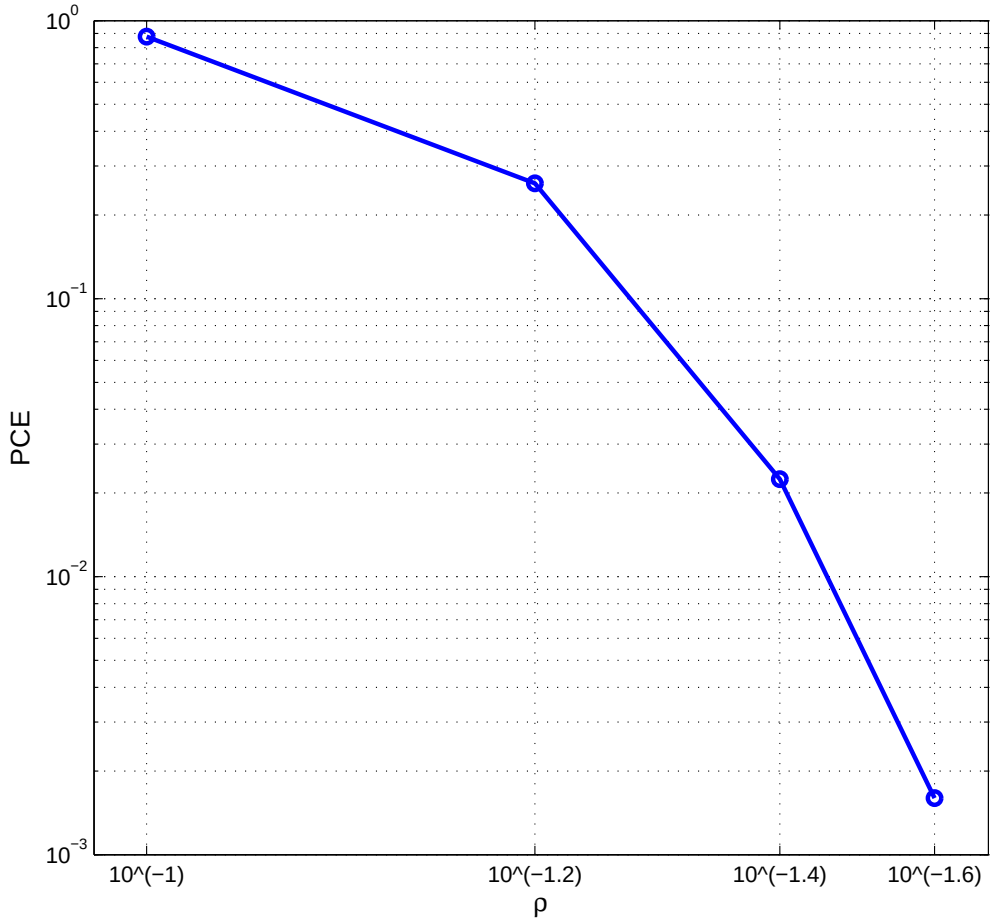

Figure 1: Word error rate of a Rate-$\frac{1}{2}$ $(3, 6)$ LDPC code of length 200 over a range of BSC crossover probabilities, corresponding to [1, Fig. 7].

Appendix B

Iterative Decoders for the Polya Channel

Figure 2 demonstrates the performance of an iterative message-passing decoder applied to the non-ergodic Polya channel using a code with the all-ones codeword. The performance is compared to the performance of the relaxed MDD introduced earlier. We note almost identical performance in comparing these two decoders.

The message-passing decoder was implemented using Matlab's "ldpcdec" object. This decoder uses a variety of the message-passing algorithm and requires as input a vector corresponding to the log-likelihood ratios (LLRs) $\underline{l} = (l_1, \ldots, l_n)$. Given a binary received vector $\underline{y} = (y_1, \ldots, y_n)$, l_i is defined as

$$l_i = \log\left(\frac{P[0|y_i]}{P[1|y_i]}\right). \tag{1}$$

For the purpose of the simulation, the probabilities above were calculated assuming a BSC with parameter p corresponding to the BER of the non-ergodic Polya channel, ρ. It turns out that the use of the parameter ρ in computing the LLRs is important in obtaining good performance over the Polya channel, as other parameters resulted in inferior error performance.

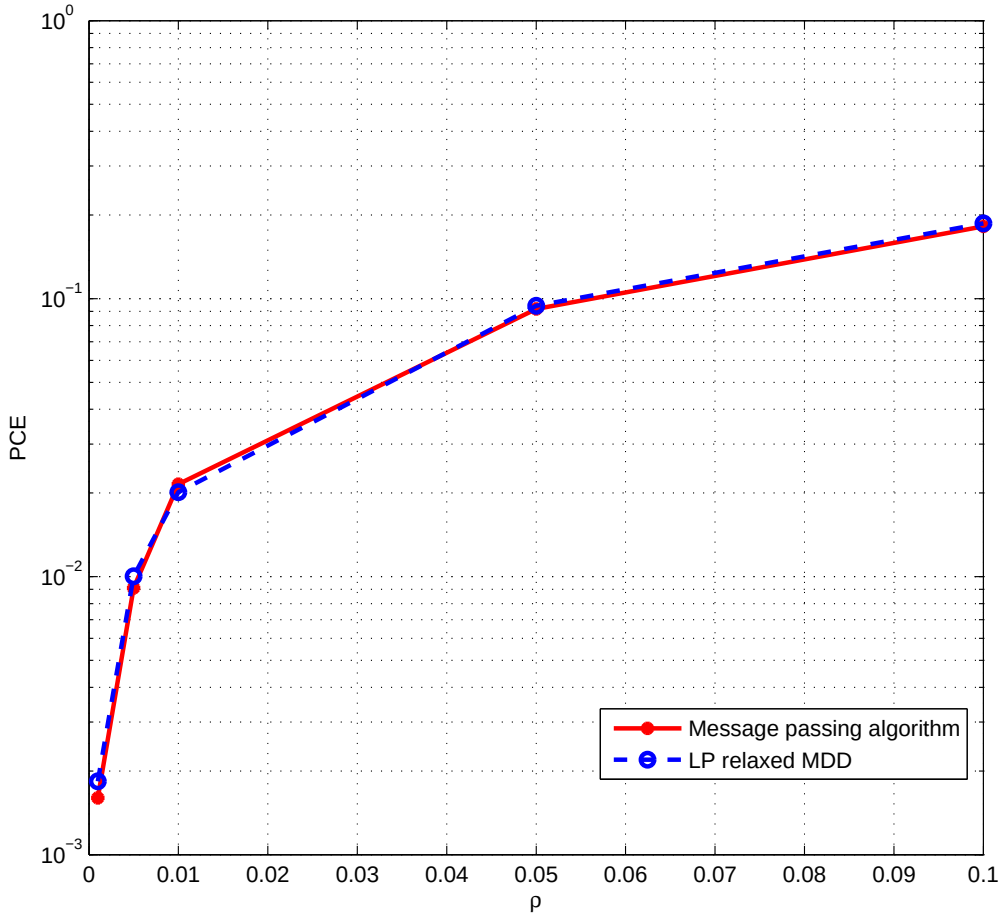

Figure 2: Word error rate of a Rate-$\frac{1}{2}$ $(3, 6)$ LDPC code of length 200 over the non-ergodic Polya channel with $\delta = 2$ using both a message-passing decoder and relaxed MDD over a range of values of the Polya channel BER ρ.