UNIVERSITY OF CALIFORNIA, SAN DIEGO

LDPC Codes – Structural Analysis and Decoding Techniques

A dissertation submitted in partial satisfaction of the
requirements for the degree
Doctor of Philosophy

in

Electrical Engineering
(Communication Theory and Systems)

by

Xiaojie Zhang

Committee in charge:

Professor Paul H. Siegel, Chair
Professor Patrick J. Fitzsimmons
Professor Gert Lanckriet
Professor Laurence B. Milstein
Professor Alexander Vardy

2012

The dissertation of Xiaojie Zhang is approved, and it is acceptable in quality and form for publication on microfilm and electronically:

Chair

University of California, San Diego

2012

DEDICATION

To Linglin, Lingbo, and my parents

TABLE OF CONTENTS

Signature Page . . . iii

Dedication . . . iv

Table of Contents . . . v

List of Figures . . . viii

List of Tables . . . x

Acknowledgements . . . xi

Vita . . . xiii

Abstract of the Dissertation . . . xiv

Chapter 1 Introduction . . . 1
1.1 Background . . . 1
1.2 Dissertation Overview . . . 4

Bibliography . . . 5

Chapter 2 Introduction to Low-Density Parity-Check Codes . . . 7
2.1 Representation of LDPC Codes . . . 7
2.2 Channel Models . . . 10
2.2.1 BEC . . . 10
2.2.2 BSC . . . 10
2.2.3 AWGNC . . . 12
2.3 Iterative Message-Passing Decoding . . . 12
2.3.1 Sum-Product Algorithm . . . 14
2.3.2 Min-Sum Algorithm . . . 14
2.4 Linear Programming Decoding . . . 15

Bibliography . . . 18

Chapter 3 Efficient Algorithms to Find All Small Error-Prone Substructures . . . 20
3.1 Error-Prone Substructures . . . 21
3.2 Algorithm Description . . . 22
3.2.1 EPS Search Algorithm . . . 23
3.2.2 FAS Search Algorithm . . . 26
3.2.3 The Exhaustiveness of Proposed Algorithms . . . 28
3.2.4 Efficiency Improvement for QC Codes . . . 28
3.3 Numerical Results . . . 29
3.3.1 Results of EPS Search Algorithm . . . 29
3.3.2 Results of FAS Search Algorithm . . . 32

3.4 Conclusion 32

Bibliography 33

Chapter 4 Quantized Iterative Message Passing Decoders with Low Error Floor for LDPC Codes 35
4.1 Notation and Definitions 37
4.2 Error Floors of LDPC Codes 42
4.2.1 Trapping Sets and Min-Sum Decoding 42
4.2.2 Trapping Sets and Sum-Product Algorithm Decoding 45
4.3 New Quantized Decoders with Low Error Floors 51
4.4 Numerical Results 55
4.5 Conclusion 65

Bibliography 65

Chapter 5 Adaptive Cut Generation Algorithm for Improved Linear Programming Decoding of Binary Linear Codes 68
5.1 Adaptive Linear Programming Decoding 70
5.2 Cut Conditions 71
5.3 LP Decoding with Adaptive Cut-Generating Algorithm 74
5.3.1 Generating Redundant Parity Checks 74
5.3.2 Reducing the Number of Constraints in the LP Problem . . 78
5.4 Numerical Results 80
5.5 Conclusion 91

Bibliography 91

Chapter 6 Efficient Iterative LP Decoding of LDPC Codes with Alternating Direction Method of Multipliers 93
6.1 ADMM Formulation of LP Decoding Problem 94
6.2 Efficient Projection onto Check Polytope 99
6.2.1 Cut Search Algorithm 99
6.2.2 Projection to Check Polytope 101
6.3 Improving the Performance of ADMM-based LP Decoding 106
6.3.1 Early Termination 106
6.3.2 Varying Penalty Parameter and Over-Relaxation 108
6.3.3 Symmetric Implementation of ADMM-based LP Decoding 109
6.3.4 Improving Error-Rate Performance of LP Decoding 110
6.4 Numerical Results 113
6.5 Conclusion 121

Bibliography 121

Chapter 7 Reliable Video Streaming over Distributed Cognitive Radio Networks . . 123
7.1 Introduction . . . 123
7.2 System Description . . . 125
7.2.1 Cognitive Radio Network Description . . . 125
7.2.2 Video Distortion Model . . . 127
7.2.3 Problem Formulation . . . 128
7.3 Proposed Cross-layer Resource Allocation Framework . . . 128
7.3.1 Detection and Estimation of Physical Channel Parameters . 129
7.3.2 Distributed Spectrum Management and Power Control Algorithm . . . 130
7.3.3 Distributed Cross-layer Resource Allocation Algorithm . . 134
7.4 Simulation Results . . . 137
7.5 Conclusion . . . 142

Bibliography . . . 142

LIST OF FIGURES

Figure 1.1: A simplified communication (or storage) system block diagram. 2

Figure 2.1: A Tanner grpah of the Hamming code of length 7. 9
Figure 2.2: A schematic illustration of the BEC, the BSC, and the AWGNC. 11
Figure 2.3: CN-to-VN message update: Each CN receives LLR information from all of its neighboring VNs. For each such VN, it generates an updated “check-to-variable” message using the inputs from all other neighboring VNs. 13
Figure 2.4: VN-to-CN message update: Each VN receives LLR information from all of its neighboring CNs. For each such VN, it generates an updated “variable-to-check” message using the inputs from all other neighboring CNs. . . . 13

Figure 4.1: Examples of (a, b) absolute trapping sets. Check nodes in set C_1 and variable nodes in set V_1 are shown shaded. 39
Figure 4.2: Example of a $(4, 4)$ trapping set and its corresponding computation tree. . . 41
Figure 4.3: A plot of the $\phi(x)$ function. 46
Figure 4.4: A plot of function $\log\left(1 + e^{-|x|}\right)$. 47
Figure 4.5: Probability density function of magnitude of messages. 57
Figure 4.6: FER results of min-sum (MS) decoder on the (640,192) QC-LDPC code on BSC. Uniform quantization step $\Delta = 1$ or 0.5, and $d = 3$ in (q+1)-bit quasi-uniform quantization. 58
Figure 4.7: FER results of min-sum (MS) decoder on the (640,192) QC-LDPC code on AWGNC. The offset factor $\beta = 0.5$, uniform quantization step $\Delta = 0.5$, and $d = 3$ in (q+1)-bit quasi-uniform quantization. 58
Figure 4.8: FER results of offset min-sum (OMS) decoder on the (640,192) QC-LDPC code on AWGNC. The offset factor $\beta = 0.5$, uniform quantization step $\Delta = 0.5$, and $d = 2$ in (q+1)-bit quasi-uniform quantization. 59
Figure 4.9: FER results of SPA decoder on the (640,192) QC-LDPC code on BSC. Uniform quantization step $\Delta = 0.25$, and $d = 1.3$ in (q+1)-bit quasi-uniform quantization. 60
Figure 4.10: FER results of SPA decoder on the (640,192) QC-LDPC code on AWGNC. The uniform quantization step $\Delta = 0.25$, and $d = 1.5$ in (q+1)-bit quasi-uniform quantization. 61
Figure 4.11: FER results of approximate-SPA decoder on the Margulis code of length 2640 on AWGNC. Uniform quantization step $\Delta = 0.25$, and $d = 1.3$ in (q+1)-bit quasi-uniform quantization. 62
Figure 4.12: FER results of approximate-SPA decoder on (4095,3358) LDPC code on AWGNC, where $\Delta = 0.5$ and $d = 1.3$. 63

Figure 5.1: FER versus E_b/N_0 for random (3,4)-regular LDPC code of length 100 on the AWGN channel. 81
Figure 5.2: FER versus E_b/N_0 for MacKay's random (3,6)-regular LDPC code of length 96 on the AWGN channel. 83
Figure 5.3: FER versus E_b/N_0 for (155,64) Tanner LDPC code on the AWGN channel. 83

Figure 5.4: FER versus E_b/N_0 for MacKay's random (3,6)-regular LDPC code of length 408 on the AWGN channel. 85
Figure 5.5: FER versus E_b/N_0 for MacKay's random (3,27)-regular LDPC code of length 999 on the AWGN channel. 85
Figure 5.6: Average number of iterations for decoding one codeword of (155,64) Tanner LDPC code. 86
Figure 5.7: Average number of constraints/cuts during decoding iterations for decoding one frame of (155,64) Tanner LDPC code. 87
Figure 5.8: Average simulation time for decoding one codeword of (155,64) Tanner LDPC code. 89

Figure 6.1: FER comparison of ADMM-base LP decoder with various maximum number of iterations for (1057,833) LDPC code on AWGNC. 114
Figure 6.2: Average simulation time for decoding one codeword. 115
Figure 6.3: ADMM parameter comparison for LDPC code of rate 0.77 and length 1057 on AWGNC, $T_{\max} = 200$ and $E_b/N_0 = 3.75$ dB. 117
Figure 6.4: ADMM parameter comparison for LDPC code of rate 0.33 and length 1920 on AWGNC, $T_{\max} = 200$ and $E_b/N_0 = 2$ dB. 118
Figure 6.5: FER comparison of ADMM-base LP decoders with additional penalty terms in objective function for (576,288) LDPC code on AWGNC. $T_{\max}$ is set to 200 for all decoders. $\alpha = 0.12$ for the $\ell 1$ penalty, $\alpha = 0.1$ for the $\ell 2$ penalty, and $\beta_i = 0.78 d_i$ for the general $\ell 2$ penalty. 119
Figure 6.6: FER comparison of ADMM-base LP decoders with additional penalty terms in objective function for (4095,3358) LDPC code on AWGNC. $T_{\max}$ is set to 200 for all decoders. $\alpha = 0.08$ for the $\ell 1$ penalty, $\alpha = 0.13$ for $\ell 2$ penalty, and $\beta_i = 0.78 d_i$ for the general $\ell 2$ penalty. 119

Figure 7.1: A cognitive radio network with 8 users (4 pairs). 125
Figure 7.2: Transmission time-frame structure. 132
Figure 7.3: An example of water-filling solution for 8 subcarriers. 132
Figure 7.4: Flowchart of distributed cross-layer resource allocation algorithm. 135
Figure 7.5: Worst-case distortion comparison for 16 and 32 available subcarriers with various number of active user-pairs. 139
Figure 7.6: Video quality versus coding bit rate. 140
Figure 7.7: Percentage of turned-off time. Note that the least turned-off user never turns off for the range of active user-pairs shown here. 141

LIST OF TABLES

Table 3.1: (a,b) EPS enumerators for the (155,64) Tanner code 29
Table 3.2: (a,b) EPS enumerators for the M816 code 30
Table 3.3: (a,b) EPS enumerators for the M1008 code 30
Table 3.4: (a,b) EPS enumerators for (256,128) LDPC code 31
Table 3.5: Simulation time comparison for FAS of PEGR504 code ($a_{\max} = b_{\max} = 5$) 31

Table 4.1: (3+1)-bit quasi-uniform quantization with $\Delta = 1$ and $d = 3$. 54
Table 4.2: 4-bit quasi-uniform quantization with $\Delta = 1$ and $d = 3$. 55

Table 5.1: Frame Errors of ACG-ALP decoder on MacKay's random (3,6)-regular LDPC code of length 96 on the AWGN channel. 82
Table 5.2: The average accumulated number of constraints in all iterations of decoding one codeword of (155,64) Tanner code on the AWGN channel 88

ACKNOWLEDGEMENTS

The past fours yeas in UCSD will be a precious memory in my whole life.

First of all, I would like to show my most sincere gratitude to my advisor, Prof. Paul H. Siegel, for his invaluable advice, unreserved support and for spending so much precious time in guiding me through my doctoral research. His kindness and enthusiasm for research is strongly impressed on my memory. From him, I not only learned the knowledge that I wrote in this dissertation, but more importantly also the way of how to solve a new problem. His speed and thoroughness is also truly impressive to me, and from him I realized the importance of paying attention to every detail. I feel really fortunate to be able to have him as my advisor, and joining his group is the best decision I have ever made. I enjoy working with him so much that the three years in his group only seems to be three months to me.

I would also like to thank Prof. Pamela Cosman and Prof. Laurence Milstein for providing me the great opportunity of working with them in the first year of my Ph.D. course and helping me adapt to the Ph.D. course. I thank Prof. Milstein also for serving in my committee and I thank Prof. Patrick J. Fitzsimmons, Prof. Gert Lanckriet, and Prof. Alexander Vardy for their time and effort in serving in my committee and for their valuable comments and suggestions on my research. I am also very grateful to my former M.S. advisor, Prof. Jungwoo Lee for his confidence in me and encouraging me to pursue my Ph.D. degree.

I would like to thank the current and former STAR members, for providing a fascinating research environment and making STAR such a delightful place for both research and relax. I am grateful to my office mate, Brian Butler, as well as Masoud Alipour, Aman Bhatia, Amir Hadi Djahanshahi, Bing Fan, Aravind Iyengar, Seyhan Karakulak, Scott Kayser, Minghai Qin, Mohammad Hossein Taghavi, Ido Tal, Saeed Sharifi Tehrani, Han Wang, and Eitan Yaakobi for many helpful and delightful discussions. I would like to thank Eitan especially for giving me the LaTex formatting template of this dissertation.

I wish to thank the CMRR and ECE staff for their excellent administrative support, particularly Ray Descoteaux, Laura Hurt, Betty Manoulian, Shana Slebioda, Bernadette Villaluz, Iris Villanueva, and Kevin Wong.

Last but most importantly, I would also like to extend my deepest gratitude to my family. I am grateful to my parents for their constant emotional support and for their words of encouragement and concern from such a distance for all these years. I owe my deepest gratitude to my wife, Linglin, who has been very supportive since I have decided to pursue my Ph.D. degree. During my Ph.D. course, she took on most of the housework even when she was busy with her own course work as an MBA student. I am also very grateful to her for taking such a good care of our son Lingbo, who also supported my study by going to bed early and giving me time to work at night. To them I dedicate this dissertation.

This research was supported in part by the Center for Magnetic Recoding Research at University of California, San Diego, the National Science Foundation under Grants CCF-0829865 and CCF-1116739, the University of California Lab Fees Research Program, Award No. 09-LR-06-118620-SIEP, and the Office of Naval Research under Grant No. N000140810081.

Chapter 3 is in part a reprint of the material in the papers: Xiaojie Zhang and Paul H. Siegel, “Efficient algorithms to find all small error-prone substructures in LDPC codes,” in *Proc. IEEE Global Communication Conference*, Houston, TX, Dec. 5–9, 2011, pp. 1–6. Chapter 4 is in part a reprint of the material in the papers: Xiaojie Zhang and Paul H. Siegel, “Quantized min-sum decoders with low error floor for LDPC codes,” in *Proc. IEEE International Symposium on Information Theory*, Cambridge, MA, July 2–5, 2012, pp. 2871–2875., and Xiaojie Zhang and Paul H. Siegel, “Will the real error floor please stand up?” in *Proc. IEEE International Conference Signal Processing and Communication*, Bangalore, India, July 22–25, 2012, pp. 1–5. Chapter 5 is in part a reprint of the material in the paper: Xiaojie Zhang and Paul H. Siegel, “Adaptive cut generation algorithm for improved linear programming decoding of binary linear codes,” *IEEE Transactions on Information Theory*, vol. 58, no. 10, pp. 6581–6594, Oct. 2012. Chapter 6 is in part a reprint of the material in the paper: Xiaojie Zhang and Paul H. Siegel, “Efficient iterative LP decoding with alternating direction method of multipliers,” *Submitted to IEEE Transactions on Information Theory*.

The dissertation author was the primary investigator and author of all these papers.

VITA

2004	B.S. in Electrical Engineering (Teaching Reform Class), Shanghai Jiao Tong University, Shanghai, China.
2006	M.S. in Electrical Engineering, Seoul National University, Seoul, Korea.
2006-2008	System Engineer, Samsung Electronics, Korea
2012	Ph.D. in Electrical Engineering (Communication Theory & Systems), University of California, San Diego.

PUBLICATIONS

Xiaojie Zhang and Paul H. Siegel, "Efficient iterative LP decoding with alternating direction method of multipliers," *Submitted to IEEE Transactions on Information Theory*, Dec. 2012.

Xiaojie Zhang and Paul H. Siegel, "Quantized iterative message passing decoders with low error floor for LDPC codes," *Submitted to IEEE Transactions on Communication*, Nov. 2012.

Xiaojie Zhang and Paul H. Siegel, "Adaptive cut generation algorithm for improved linear programming decoding of binary linear codes," *IEEE Transactions on Information Theory*, vol. 58, no. 10, pp. 6581–6594, Oct. 2012.

Xiaojie Zhang and Paul H. Siegel, "Will the real error floor please stand up?" in *Proc. IEEE International Conference Signal Processing and Communication (SPCOM 2012)*, Bangalore, India, July 22–25, 2012, pp. 1–5.

Xiaojie Zhang and Paul H. Siegel, "Quantized min-sum decoders with low error floor for LDPC codes," in *Proc. IEEE International Symposium on Information Theory (ISIT 2012)*, Cambridge, MA, July 2–5, 2012, pp. 2871–2875.

Xiaojie Zhang and Paul H. Siegel, "Efficient algorithms to find all small error-prone substructures in LDPC codes," in *Proc. IEEE Global Communication Conference (Globecom 2011)*, Houston, TX, Dec. 5–9, 2011, pp. 1–6.

Xiaojie Zhang and Paul H. Siegel, "Adaptive cut generation for improved linear programming decoding of binary linear codes," in *Proc. IEEE International Symposium on Information Theory (ISIT 2011)*, St. Petersburg, Russia, August 1–5, 2011, pp. 1638–1642.

Xiaojie Zhang and Jungwoo Lee, "Low complexity MIMO scheduling with channel decomposition using capacity upperbound," *IEEE Transactions on Communications*, vol. 56, pp. 871–876, June 2008.

Xiaojie Zhang, Jungwoo Lee, and Huaping Liu, "Low complexity multiuser MIMO scheduling with channel decomposition," in *Proc. of IEEE wireless Communication and Networking Conference (WCNC 2007)*, Hong Kong, March 2007, pp. 2452–2456.

Xiaojie Zhang and Jungwoo Lee, "Low complexity multiuser MIMO scheduling with channel decomposition," in *Proc. of IEEE Asilomar Conference on Signals, Systems, and Computers*, Pacific Grove, CA, Oct. 29– Nov.1, 2006, pp. 641–645.

ABSTRACT OF THE DISSERTATION

LDPC Codes – Structural Analysis and Decoding Techniques

by

Xiaojie Zhang

Doctor of Philosophy in Electrical Engineering
(Communication Theory and Systems)

University of California, San Diego, 2012

Professor Paul H. Siegel, Chair

Low-density parity-check (LDPC) codes have been the focus of much research over the past decade thanks to their near Shannon limit performance and to their efficient message-passing (MP) decoding algorithms. However, the error floor phenomenon observed in MP decoding, which manifests itself as an abrupt change in the slope of the error-rate curve, has hindered the adoption of LDPC codes and MP decoders in some applications requiring very low error rates.

As an alternative to MP decoding, linear programming (LP) decoding is an approximation to maximum-likelihood decoding by relaxing the optimal decoding problem into a linear optimization problem. It has been noticed that, when the symbols error probability in channel output is low, LP decoding has superior error correction performance. However, due to the inefficiency of general-purpose LP solvers which are not optimized for solving LP problems, LP decoding is computationally more complex than MP decoding, especially for codes of large block size.

In this dissertation, we first design an efficient exhaustive search algorithm to find all small error-prone substructures, some of which are commonly blamed for certain decoding failures of MP decoding. Then, we investigate the cause of error floors in LDPC codes from the perspective of the MP decoder implementation, with special attention to limitations that decrease the numerical accuracy of messages passed during decoding, and propose a quantization method for fixed-point implementation of MP decoding which significantly improves the error-floor performance by overcoming the limitations of standard quantization rules.

For LP decoding, we improve the error-correcting capability of LP decoding by using an effective algorithm to generate additional redundant parity-check constraints which eliminate certain undesired solutions to LP decoding problem. We further propose an efficient message-passing algorithm to solve the LP decoding problem. This algorithm is based on the alternating direction method of multipliers (ADMM), a classic technique in convex optimization theory, and our key contribution is a novel, efficient projection algorithm that can improve the decoding speed of the ADMM-based LP decoder.

The last part of this dissertation is a separate piece of work on optimizing video transmission over distributed cognitive radio network.

Chapter 1

Introduction

1.1 Background

The human civilization is developed generation by generation, based on learning old knowledge from previous generations and creating new for current and future generations. In modern society, the amount of knowledge and information grows exponentially, and people are interacting with each other more often. Thanks to the digital revolution, almost all of the information in our daily lives is digitalized. At work, people write emails on their cell phones and send them through wireless networks. At home, people watch high definition movies from DVDs or even in realtime streaming through the internet. At school, many textbooks have their electronic version and students read books on their computers and tablets. Digital media and digital communication systems are required to store and transmit such digital information. Examples of digital media include data-storage devices such as magnetic disk drives, optical disk drives and flash drives, etc. Digital communication examples include but are not limited to cell phones in cellular networks, digital TV via satellite or cable, wireless or wired connection to the internet.

All of these examples of storage systems and digital communication systems, while wildly distinct in implementation method and apparatus, are generally based on a common foundation established by Claude Shannon in 1948 [1]. Fig. 1.1, although much simplified, shows the essentials of a digital system. The information from the source is first converted into *codewords* by an encoder before being sent through the channel. At the user side, in order to recover the actual information sent from the source, the outputs of the channel have to be decoded by a decoder. Any digital media or digital communication system can be considered as a transmission

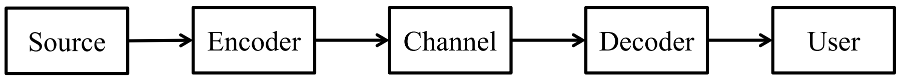


Figure 1.1: A simplified communication (or storage) system block diagram.

of information through a communication channel. For example, a DVD can be viewed as the channel that carriers the information of a movie between the movie maker and audiences; and electromagnetic waves are the channels that transmit cellular phones calls. In almost all of the communication channels, information cannot be perfectly transmitted, and a distorted (or noisy) version is often received. A scratch on a DVD will corrupt the information stored in that area on the DVD, and any natural source such as weather conditions, radiation, and thermal effects can cause the distortion of electromagnetic waves that carry modulated and coded information.

To transmit information reliably over an unreliable channel has been one of the crucial topics in information theory. On the basis of the model described in Fig. 1.1, Shannon showed that every noisy channel has a parameter, C, called the *channel capacity*, which is a measure of how much information can be reliably transmitted through the channel and is determined by the statistics of the channel. This is much like the capacity of an elevator to carry people. It was also recognized that errors that occur in some parts of the data can be recovered from the rest by introducing redundancy into the stream of digital data bits. This method is call *channel coding*, and the data stream with redundant information capable of correcting errors is called an *error-correction code*. The ratio of the number of information bits (without redundancy) to the total number of information and redundant bits is call the *code rate*, commonly denoted by R, with $R \in [0, 1]$. When using the same unit as R to measure channel capacity C, i.e., information bits per channel bit, Shannon's coding theory indicated that there exist error-correction codes of any rate $R < C$ that provide arbitrarily reliable communication. This means that the decoder only fails to recover the information bits with an arbitrarily small probability . Conversely, reliable communication can not be realized with codes of rate greater than the channel capacity.

When Shannon proved in his celebrated 1948 paper *A Mathematical Theory of Communication* that there exist good codes that achieve the channel capacity, he also pointed out that a randomly chosen code is asymptotically good enough on average to achieve the channel capacity with high probability. However, it remains unclear how to find such good codes and how to encode and decode them in practice with reasonable complexity. Since then, finding good codes that achieve channel capacity has been the focus of much research in coding theory. In 1955, Elias [2] showed that the capacity of a discrete memoryless channel can be achieved even with *linear codes*, which is a set of vectors from a linear vector subspace and can be described as the kernel of a *parity-check matrix*. The linear codes can be efficiently encoded by multiplying the source information vector with a *generator matrix*, which is a basis for the linear subspace and also a parity-check matrix for the dual code. Since then, the focus of coding theory has been on capacity-achieving code constructions and corresponding efficient decoding methods. Since then until the 1990s, many classic codes have been designed. Although none of them were able to provide performance close to Shannon's limit, some of them are still of great success and widely implemented even today. Examples are *Bose-Chaudhuri-Hocquenghem (BCH)* codes [3–5], *Reed-Solomon* codes [6], and *convolutional* codes [7, 8].

In the 1990s, there were significant breakthroughs in coding theory and application. The first breakthrough came in 1993 with the discovery of *turbo* codes, the first class of codes with performance near Shannon's theoretical limit [9, 10]. A couple of years after the discovery of turbo codes, another breakthrough came with the rediscovery of *low-density parity-check (LDPC)* codes, which also closely approach Shannon's capacity limit in practice [11–13]. Although LDPC codes were first invented by Gallager in the 1960s [14, 15], due to the limit of computational power at that time, they stayed mostly unnoticed for more than forty years. Turbo codes and LDPC codes both have random-like code construction with well structured *partial descriptions* and large length, and efficient and effective suboptimal decoding algorithms. The randomness provides vast choices of codes that are likely to provide good performance with a reasonable large block size, while partial descriptions allow the decoding being based on a set of "constituent decoders" that iteratively process and exchange their local information obtained from partial descriptions. The decoding of constituent codes in turbo decoding and the updates of messages in parity-checks of LDPC codes are examples of the constituent decoders. Although for most turbo or LDPC codes there is no guarantee that such a decoding process will converge to a codeword, or will converge at all, these iterative decoders in general perform extremely well in practice. However, both of these classes of codes suffer from the error floor phenomenon, which manifests itself as an abrupt change in the slope of the error-rate curve. It has been known that the low-weight codewords in turbo codes determine the error floor. However, for general memoryless binary-input output-symmetric (MBIOS) channels such as the binary symmetric channel (BSC) and the additive white gaussian noise channel (AWGNC), it is still not clear what causes and determines the error floor of LDPC codes.

1.2 Dissertation Overview

In this dissertation, we focus on the structural analysis and decoding techniques of binary LDPC codes.

Chapter 2 provides background about LDPC codes and most frequently encountered memoryless channels. It also introduces two major types of decoding algorithms for LDPC codes – *iterative message-passing (MP)* decoding algorithms and *linear programming (LP)* decoding algorithms.

The error floor phenomenon observed with LDPC codes and their graph-based, iterative MP decoders is commonly attributed to the existence of error-prone substructures in a Tanner graph representation of the code. However, exhaustively enumerating all small error-prone substructures in arbitrary, finite-length LDPC codes has been proven to be NP-complete. In Chapter 3, we present two efficient exhaustive search algorithms that are able to find all small error-prone substructures of an arbitrary LDPC code given its parity-check matrix up to a certain given size.

The study of error floors with MP decoding is continued in Chapter 4. In contrast to many observations, we show that the source of the error floors observed in the literature could be partially due to imprecise implementation of the iterative MP decoding algorithms and the message quantization rules used. We then propose a new quantization method to overcome the limitations of conventional quantization rules. Performance simulation results for two LDPC codes commonly found to have high error floors when used with fixed-point iterative MP decoding algorithms provide an example of the practical application of our idealized theoretical results and the effectiveness of the proposed quantization method.

We continue our study with LP decoders for LDPC codes. In Chapter 5, we first derive a necessary condition and a sufficient condition for a violated parity inequality constraint, or "cut," at a point in the unit hypercube. Then, we propose a new and effective algorithm to generate parity inequalities derived from certain additional redundant parity check constraints that can eliminate pseudocodewords produced by the LP decoder, often significantly improving the decoder error-rate performance. The cut-generating algorithm is based upon a specific transformation of an initial parity-check matrix of the linear block code. We also design two variations of the proposed decoder to make it more efficient when it is combined with the new cut-generating algorithm.

To reduce the complexity of LP decoding, in Chapter 6 we propose an efficient message-passing algorithm to solve the LP decoding problem. It is based on the alternating direction method of multipliers (ADMM), a classic technique in convex optimization theory that is designed for parallel implementation. The computational complexity of ADMM-based LP decoding is largely determined by the method used to project a vector of real values to some specific polytopes. We proposed a novel, efficient projection algorithm that can substantially improve the decoding speed of the ADMM-based LP decoder, and further improvement on decoding efficiency can be achieved by using early termination and some optimization techniques such as over-relaxation which give superior convergence in practice.

Chapter 7 is a self-contained study on a cross-layer distributed power control framework for reliably transmitting real-time videos on a distributed cognitive radio network using multicarrier direct-sequence code division multiple access (DS-CDMA) over frequency-selective fading channels. The framework exploits the relationship between transmission rate and quality of the video when combined with a physical layer consisting of a multicarrier direct sequence waveform. We employ multiuser diversity on each subcarrier of a cognitive radio system to achieve reliable video transmission subject to constraints on delay and limited system resources. We formulate the reliability problem as minimizing the maximal end-to-end video distortion received among all users. Compared to non-cross-layer schemes, where subcarriers and video rates are assigned separately, our cross-layer algorithm provides substantial performance improvement, in terms of the maximum distortion of all transmitted video streams.

Bibliography

[1] C.E. Shannon, “A mathematical theory of communication,” *Bell Systems Tech. J.*, vol. 27, pp. 379–423 and pp. 623–656, 1948.

[2] P. Elias, “Coding over noisy channels,” *IRE Convention Record*, 1955, pp. 37–46.

[3] A. Hocquenghem, “Codes correcteurs d’erreurs,” Chiffres, vol. 2, pp. 147–156, 1959.

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

[5] W. W. Peterson, “Encoding and error-correction procedures for the BoseCChaudhuri codes,” *IRE Trans. Information Theory*, vol. 6, no. 5, pp. 459–470, September 1960.

[6] I. S. Reed and G. Solomon, “Polynomial codes over certain finite fields,” *J. Soc. Indust. Appl. Math.* vol. 8, pp. 300–304, June 1960.

[7] G. D. Forney, “Convolutional codes I: Algebraic structure,” *IEEE Trans. Information Theory*, vol. 16, no. 11, pp. 720–738, November 1970.

[8] R. Johannesson and K. S. Zigangirov, *Fundamentals of Convolutional Coding*, New York, IEEE Press, 1999.

[9] C. Berrou, A. Glavieux, and P. Thitimajshima, “Near Shannon limit error-correcting coding and decoding: Turbo codes,” in *Proc. International Conference on Communications*, Geneva, Swizerland, May 1993, pp. 1064–1070.

[10] C. Berrou and A. Glavieux, “Near optimum error correcting coding and decoding: turbo-codes,” *IEEE Trans. Communications*, vol. 44, no. 10, pp. 1261–1271, October 1996.

[11] D. J. C. MacKay and R. M. Neal, “Good codes based on very sparse matrices,” in *Cryptography and Coding: Lecture Notes in Computer Science*, vol. 1025. C. Boyd, Ed. Heidelberg: Springer Berlin, 1995, pp. 100–111.

[12] N. Wiberg, “Codes and decoding on general graphs,” Ph. D. dissertation, Linköping University, 1996.

[13] T. J. Richardson, M. A. Shokrollahi, and R. L. Urbanke, “Design of capacity-approaching irregular low-density parity-check codes,” *IEEE Trans. Information Theory*, vol. 47, no. 2, pp. 619–637, Februay 2001.

[14] R. G. Gallager, “Low-density parity-check codes.” *IRE Trans. Inform. Theory*, vol. 8, pp. 21–28, Jan. 1962.

[15] R. G. Gallager, *Low-Density Parity-Check Code*, MIT Press, Cambridge, MA, 1963.

Chapter 2

Introduction to Low-Density Parity-Check Codes

Shannon pointed out in his landmark paper [1] that a random code of large block length with optimal decoding can achieve channel capacity. However, the optimal decoder alone is prohibitively complex and impossible for practical implementation. In the 1960s, Gallager proposed a class of linear block codes whose parity-check matrices have few non-zero entries, and hence such codes are named *low-density parity-check (LDPC)* codes [2, 3]. He also provided a decoding method for LDPC codes which consists of a set of constituent decoders of low complexity that update and exchange information iteratively. Although this suboptimal *iterative message-passing (MP)* decoding method is algorithmically simple, performance simulation was still beyond the computational power of that age and its near-capacity performance in practice on LDPC codes of large block length was not realized. As a result, LDPC codes were largely forgotten until they were rediscovered by MacKay *et al.* in the 1990s [4, 5].

In this chapter, we will give an overview of binary LDPC codes, and introduce two major classes of decoding algorithms.

2.1 Representation of LDPC Codes

A *linear code* of length n and dimension k is a linear subspace $\mathcal{C}$ with dimension k of the vector space $\mathbb{F}_q^n$ where $\mathbb{F}_q$ is the finite field with q elements. The vectors in $\mathcal{C}$ are called *codewords*. When $q = 2$, the code is called *binary* code.

A binary linear block code generates a block of n coded bits from k information bits.

We call this an (n, k) binary block code, and the rate of the code is $R = k/n$. The *generator matrix* $\mathbf{G}$, a $k \times n$ binary matrix, maps an information vector $\mathbf{u} \in \mathbb{F}_2^k$ into a codeword $\mathbf{c} \in \mathbb{F}_2^n$ such that

$$\mathbf{c} = \mathbf{u}\mathbf{G}.$$

The rows of generator matrix form a basis for the codeword space $\mathcal{C}$, and any $k \times n$ matrix whose rows form a basis for $\mathcal{C}$ can also be used as the generator matrix for the code.

An alternative way to define a linear block code of length n is to use an $m \times n$ *parity-check matrix* $\mathbf{H}$ such that

$$\mathcal{C} = \{\mathbf{x} \in \mathbb{F}_2^n : \mathbf{H}\mathbf{x}^T = \mathbf{0}\}.$$

The code $\mathcal{C}$ is the kernel of $\mathbf{H}$, and any set of vectors that span the row-space generated by $\mathbf{H}$ can serve as the rows of a parity check matrix. The row rank of any parity-check matrices describing an (n, k) coed should be $n - k$. For any generator matrix $\mathbf{G}$ and parity-check matrix $\mathbf{H}$ of the same code, we have

$$\mathbf{H}\mathbf{G}^T = \mathbf{0}.$$

Example 2.1 The following generator matrix $\mathbf{G}$ and parity-check matrix $\mathbf{H}$ describe the same linear block code, which is known as the *Hamming code* of length 7 [6].

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

It can be verified in this example that $\mathbf{H}\mathbf{G}^T = \mathbf{0}$.

A binary *low-density parity-check (LDPC)* code is a binary linear block code with a parity-check matrix that has a low density of 1s. This property make it convenient to represent an LDPC code described by parity-check matrix $\mathbf{H}$ with a bipartite graph, called a *Tanner Graph* [7]. The two types of nodes in a Tanner graph are the *variable nodes (VNs)* and the *check nodes (CNs)*. The variable nodes are also known as *bit* nodes or *left* nodes, and the check nodes are also known as *constraint* nodes or *right* nodes. Every variable node (or check node) corresponds to a column (or row) of the parity-check matrix $\mathbf{H}$. We denote by $V = \{v_1, \ldots, v_n\}$ the set of variable nodes, and by $C = \{c_1, \ldots, c_m\}$ the set of check nodes. We also index each row of $\mathbf{H}$ by $\mathcal{J} = \{1, \ldots, m\}$ and each column of $\mathbf{H}$ by $\mathcal{I} = \{1, \ldots, n\}$. In this dissertation, we shall use both the notation CN i and VN j and the notation CN c_i and VN v_j, depending on the context.

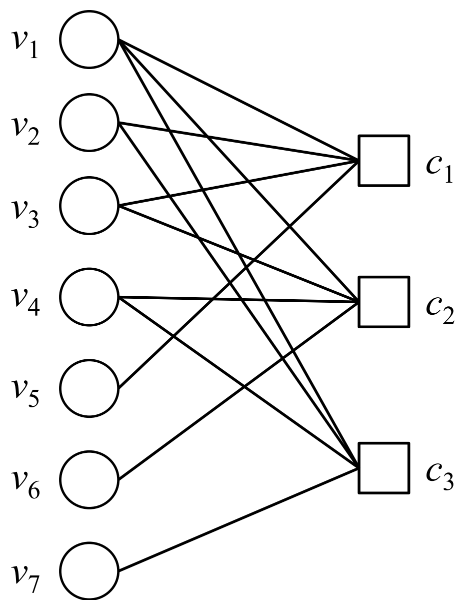


Figure 2.1: A Tanner grpah of the Hamming code of length 7.

In the Tanner graph, VN v_i is connected to CN c_j via an edge if $H_{j,i} = 1$, and the set of edges on the Tanner graph is denoted by set E. The Tanner graph, $G = (V \cup C, E)$, of the parity-check matrix in Example 2.1 is shown in Fig. 2.1.

Two nodes connected by an edge in the Tanner graph are called *neighboring nodes*. The *degree* of a node is the number of edges connected to it, i.e., the number of its neighboring nodes. If the degrees of all VNs in the Tanner graph are the same, then the corresponding LDPC code is called a *variable-regular* code, and the degree of VNs is called the *variable* degree or *left* degree. If the degrees of all CNs are the same, then the code is called *check-regular*, and the degree of CNs is called the *check* degree or *right* degree. If the degree of all VN is d_v and the degree of all CNs is d_c, then the LDPC code is called a (d_v, d_c)-regular code; otherwise, the code is called an *irregular* LDPC code.

The following notation will be frequently used throughout this dissertation. We denote by $\mathcal{N}_j = \{i \in \mathcal{I} : H_{ij} = 1\}$ the index set of neighboring VNs of CN $j \in \mathcal{J}$, and analogously denote by $\mathcal{N}_i = \{j \in \mathcal{J} : H_{ij} = 1\}$ the index set of neighboring CNs of VN $i \in \mathcal{I}$. The degree of CN j is denoted by $d_j = |\mathcal{N}_j|$, and the degree of VN i is $d_i = |\mathcal{N}_i|$, where $|\cdot|$ denotes the cardinality of a set. Denote by $\mathcal{N}_j(k) \in \mathcal{N}_j$ the kth element in $\mathcal{N}_j$, $1 \leqslant k \leqslant |\mathcal{N}_j|$.

2.2 Channel Models

In this dissertation, we focus our study on memoryless binary-input output-symmetric (MBIOS) channels, such as the binary erasure channel (BEC), the binary symmetric channel (BSC), and the additive white gaussian noise channel (AWGNC). For simplicity, it is assumed that each encoded bit $c_i \in \{0, 1\}$ is transmitted over the MBIOS channel as a binary antipodal symbol $x_i \in \{-1, 1\}$ such that

$$x_i = 1 - 2c_i.$$

For the MBIOS channel, if the transmitted vector is $\mathbf{x} = [x_1, \ldots, x_n]$, the probability of receiving vector $\mathbf{y} = [y_1, \ldots, y_n]$ can be computed as

$$\Pr(\mathbf{y}|\mathbf{x}) = \prod_{i=1}^{n} \Pr(Y_i = y_i | X_i = x_i) \triangleq \prod_{i=1}^{n} P_{Y_i|X_i}(y_i|x_i).$$

Hence, without loss of generality, we focus on computing the *log-likelihood ratio (LLR)* of the ith channel output symbol, L_i^{ch}, which is defined as

$$\begin{aligned} L_i^{ch} &= \log\left(\frac{P_{Y_i|X_i}(y_i|x_i = 1)}{P_{Y_i|X_i}(y_i|x_i = -1)}\right) \\ &= \log\left(\frac{P_{Y_i|C_i}(y_i|c_i = 0)}{P_{Y_i|C_i}(y_i|c_i = 1)}\right). \end{aligned}$$

We consider the following three widely studied cases of MBIOS channels.

2.2.1 BEC

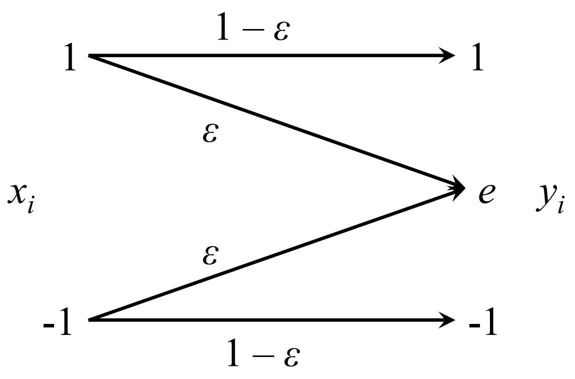
The received symbol of BEC is shown in Fig. 2.2(a), where $y_i \in \{-1, 1, e\}$. We define $\epsilon = \Pr(y_i = e | x_i = b)$ to be the erasure probability, where $b \in \{-1, 1\}$. The LLR of BEC output y_i can be computed as

$$L_i^{ch} = \begin{cases} +\infty & \text{if } y_i = 1, \\ -\infty & \text{if } y_i = -1, \\ 0 & \text{if } y_i = e. \end{cases}$$

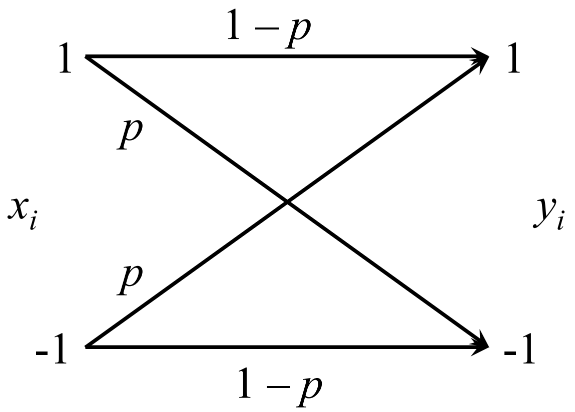
2.2.2 BSC

The received symbol of BSC is shown in Fig. 2.2(b), where $y_i \in \{-1, 1\}$. We define $p = \Pr(y_i = -b | x_i = b)$ to be the channel error probability, where $b \in \{-1, 1\}$. The LLR of BSC output y_i can be computed as

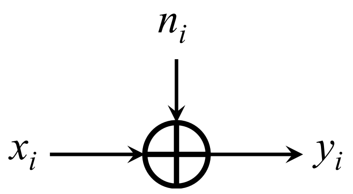
$$L_i^{ch} = y_i \log\left(\frac{1-p}{p}\right).$$



(a) BEC(ϵ)



(b) BSC(p)



(c) AWGNC(σ^2)

Figure 2.2: A schematic illustration of the BEC, the BSC, and the AWGNC.

We can see that, for a fixed p, the LLRs of BSC outputs have the same magnitude.

2.2.3 AWGNC

The AWGNC is shown in Fig. 2.2(c). The received symbol $y_i = x_i + n_i$ is a real value, where n_i is independent and identically-distributed (i.i.d.) Gaussian noise of zero-mean and variance σ^2. Then, the conditional probability between the transmitted and received symbol is

$$P_{Y_i|X_i}(y_i|x_i) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y_i - x_i)^2}{2\sigma^2}\right).$$

Hence, the LLR of AWGNC output y_i can be computed as

$$L_i^{ch} = \log\left(\frac{\frac{1}{\sqrt{2\pi}\sigma}\exp\left(-\frac{(y_i-1)^2}{2\sigma^2}\right)}{\frac{1}{\sqrt{2\pi}\sigma}\exp\left(-\frac{(y_i+1)^2}{2\sigma^2}\right)}\right) = \frac{2y_i}{\sigma^2}.$$

2.3 Iterative Message-Passing Decoding

The idea of iterative *message-passing (MP)* is that, once the decoder gets the LLRs of every received codeword symbol, every node updates its messages only based on the messages received from its neighboring node in the Tanner graph. If the parity-check matrix is sparse, each node only has a small number of neighbors and the complexity of computing the updated messages is hence low. Although there is no guarantee that such iterative updating and exchanging of messages with neighbors will eventually make messages in all VNs satisfy every parity-check constraint so that they correspond to a valid codeword, the iterative MP decoding performs extremely well in practice with well-designed LDPC codes and their properly chosen Tanner graph representations.

The iterative MP decoder alternates between two phases, a “VN-to-CN” phase during which VNs send messages to CNs along their adjacent edges, and a “CN-to-VN” phase during which CNs send messages to their adjacent VNs. The message update rules, whose details will be given later in this section, are depicted schematically in Figs. 2.3 and 2.4, respectively. In the initialization step of the decoding process, VN v_i forwards the same message to all of its neighboring CNs in $\mathcal{N}_i$, namely the LLR L_i^{ch} derived from the corresponding channel output. In the CN-to-VN message update phase, CN c_j uses the incoming messages and CN update rule to compute and forward, to VN v_i in $\mathcal{N}_j$, a new “CN-to-VN” message, $L_{j\to i}$. VN v_i then processes its incoming messages according to VN update rule and forwards to each adjacent CN an updated “VN-to-CN” message, $L_{i\to j}$. After a prespecified number of iterations, VN v_i sums

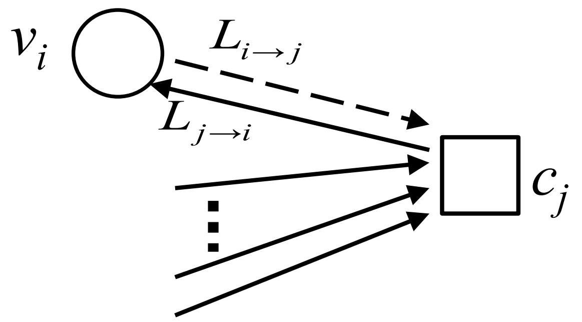


Figure 2.3: CN-to-VN message update: Each CN receives LLR information from all of its neighboring VNs. For each such VN, it generates an updated "check-to-variable" message using the inputs from all other neighboring VNs.

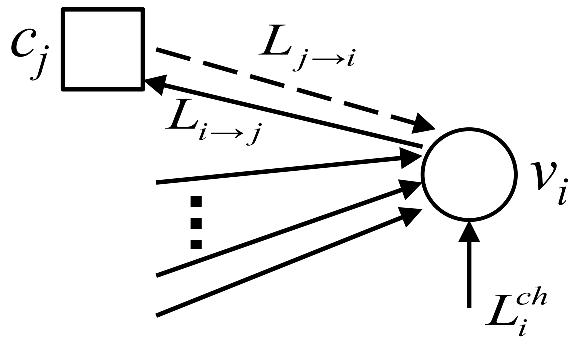


Figure 2.4: VN-to-CN message update: Each VN receives LLR information from all of its neighboring CNs. For each such VN, it generates an updated "variable-to-check" message using the inputs from all other neighboring CNs.

all of the incoming LLR messages to produce an estimate of the corresponding code bit i. Note that all of the “CN-to-VN” message updates can be done in parallel, as can all of the “VN-to-CN” message updates. This enables efficient, high-speed software and hardware implementations of the iterative MP decoding algorithms. In the remaining part of this section, we will introduce two classes of widely adopted iterative MP decoding algorithms, the sum-product algorithm and the min-sum algorithm.

2.3.1 Sum-Product Algorithm

When Gallager introduced LDPC codes in 1960s, he also proposed a suboptimal decoding algorithm which provides near-optimal decoding performance. This algorithm is now called the *sum-product algorithm (SPA)* [8], or *belief-propagation (BP)* algorithm [9].

The VN update rule of SPA is as follow

$$L_{i\to j} = L_i^{ch} + \sum_{j'\in\mathcal{N}_i\backslash j} L_{j'\to i}\ , \tag{2.1}$$

and the CN update rule is

$$L_{j\to i} = \left[\prod_{i'\in\mathcal{N}_j\backslash i} \text{sign}(L_{i'\to j})\right] \cdot \min_{i'\in\mathcal{N}_j\backslash i} |L_{i'\to j}|. \tag{2.2}$$

In practical implementations of the SPA, the following equivalent CN update rule is often used

$$L_{j\to i} = \left[\prod_{i'\in\mathcal{N}_j\backslash i} \text{sign}(L_{i'\to j})\right] \cdot \phi^{-1}\left(\sum_{i'\in\mathcal{N}_j\backslash i} \phi(|L_{i'\to j}|)\right) \tag{2.3}$$

where $\phi(x) = -\log[\tanh(x/2)]$ and $\phi^{-1}(x) = \phi(x)$. In Chapter 4, we will discuss more about the CN update rule of SPA.

2.3.2 Min-Sum Algorithm

The *min-sum algorithm (MSA)* can be viewed as a simple approximation of the sum-product algorithm [5]. Its VN update rule is the same as that of SPA, given by

$$L_{i\to j} = L_i^{ch} + \sum_{j'\in\mathcal{N}_i\backslash j} L_{j'\to i}\ , \tag{2.4}$$

and its CN update rule is

$$L_{j\to i} = \left[\prod_{i'\in\mathcal{N}_j\backslash i} \text{sign}(L_{i'\to j})\right] \cdot \min_{i'\in\mathcal{N}_j\backslash i} |L_{i'\to j}|. \tag{2.5}$$

It can been seen from (2.4) and (2.5) that the min-sum decoding algorithm is insensitive to linear scaling, meaning that linearly scaling all input messages from the channel would not affect the decoding performance.

In general, the error-rate performance of min-sum decoding is not as good as the more complicated SPA decoding. However, there are several quite simple but effective ways to adjust the CN update rule of min-sum decoding to get comparable performance to SPA decoding. One method is *attenuated-min-sum* (AMS) decoding [10], where the magnitudes of messages are attenuated at CNs. The corresponding CN update rule of AMS is as follows

$$L_{j\to i} = \left[\prod_{i'\in\mathcal{N}_j\backslash i} \mathrm{sign}(L_{i'\to j})\right] \cdot \alpha \cdot \min_{i'\in\mathcal{N}_j\backslash i} |L_{i'\to j}|, \tag{2.6}$$

where $0 < \alpha < 1$ is the attenuation factor, which can be a fixed constant or adaptively adjusted.

Another way to improve the error-rate performance of min-sum decoding is *offset-min-sum* (OMS) decoding, which applies an offset to reduce the magnitudes of CN output messages. The resulting CN update equation is

$$L_{j\to i} = \left[\prod_{i'\in\mathcal{N}_j\backslash i} \mathrm{sign}(L_{i'\to j})\right] \cdot \max\{\min_{i'\in\mathcal{N}_j\backslash i} |L_{i'\to j}| - \beta,\ 0\}, \tag{2.7}$$

where $\beta > 0$ is the offset which, like the attenuation factor, can be a fixed constant or adaptively adjusted. In some implementations, for additional simplicity, the attenuation factor or offset is set to be the same fixed constant for all CNs and all iterations [10].

2.4 Linear Programming Decoding

Despite the unparalleled success of iterative MP decoding in practice, it is quite difficult to analyze the performance of such iterative MP decoders due to the heuristic nature of their message update rules and their local nature. An alternative approach, linear programming (LP) decoding, was introduced by Feldman *et al.* [11] as an approximation to maximum-likelihood (ML) decoding.

Many theoretical and empirical observations suggest similarities between the performance of LP and MP decoding methods. For example, graph-cover decoding can be used as a theoretical tool to show the connection between LP decoding and iterative MP decoding [12].

However, there are some key differences that distinguish LP decoding from iterative MP decoding. One of these differences is that the LP decoder has the *ML certificate property*, i.e., it is detectable if the decoding algorithm fails to find an ML codeword. When it fails to find an ML codeword, the LP decoder finds a non-integer solution, commonly called a *pseudocodeword*. Another difference is that while adding redundant parity checks satisfied by all the codewords can only improve LP decoding, adding redundant parity checks may have a negative effect on MP decoding, especially in the waterfall region, due to the creation of short cycles in the Tanner graph. This property of LP decoding allows improvements by tightening the LP relaxation, i.e., reducing the feasible space of the LP problem by adding more linear constraints from redundant parity checks, as will be discussed in details in Chapter 5. In this section, we will briefly introduce the LP decoding technique.

It is well-known that the ML decoder finds the solution to the following optimization problem

$$\begin{aligned} \text{minimize} \quad & \boldsymbol{\gamma}^T \mathbf{u} \\ \text{subject to} \quad & \mathbf{u} \in \mathcal{C} \end{aligned} \tag{2.8}$$

where $u_i \in \{0, 1\}$, and $\boldsymbol{\gamma}$ is the vector of LLR of each received codeword symbol. Since the ML decoding problem (2.8) is an integer programming problem, it is desirable to replace its integrality constraints with a set of linear constraints, transforming the IP problem into a more readily solved LP problem. The desired feasible space of the corresponding LP problem should be the *codeword polytope*, i.e., the convex hull of all the codewords in $\mathcal{C}$. With this, unless the cost vector of the LP decoding problem is orthogonal to a face of the constraint polytope, the optimal solution is one integral vertex of its codeword polytope, in which case it is the same as the output of the ML decoder. When the LP solution is not unique, there is at least one integral vertex corresponding to an ML codeword. However, the number of linear constraints typically needed to represent the codeword polytope increases exponentially with the code length, which makes such a relaxation impractical.

As an approximation to ML decoding, Feldman *et al.* [11] relaxed the codeword polytope to a polytope now known as the *fundamental polytope* [12], denoted as $\mathcal{P}(\mathbf{H})$, which depends on the parity-check matrix $\mathbf{H}$.

Definition 2.1 *Let us define*

$$\mathcal{C}_j \triangleq \{\mathbf{x} \in \mathbb{F}_2^n | \langle \mathbf{x}, \mathbf{h}_j \rangle = 0 \; (\textit{in}\, \mathbb{F}_2)\} \tag{2.9}$$

where $\mathbf{h}_j$ is the jth row of the parity-check matrix $\mathbf{H}$ and $1 \leqslant j \leqslant m$. Thus, $\mathcal{C}_j$ is the set of all binary vectors satisfying the jth parity-check constraint. We denote by conv$(\mathcal{C}_j)$ the convex hull of $\mathcal{C}_j$ in $\mathbb{R}^n$, which consists of all possible real convex combinations of the points in $\mathcal{C}_j$, *now regarded as points in $\mathbb{R}^n$. The fundamental polytope $\mathcal{P}(\mathbf{H})$ of the parity-check matrix $\mathbf{H}$ is defined to be the set*

$$\mathcal{P}(\mathbf{H}) = \bigcap_{j=1}^{m} \mathit{conv}(\mathcal{C}_j). \tag{2.10}$$

Therefore, LP decoding can be written as the following optimization problem:

$$\begin{aligned} &\text{minimize} \quad \boldsymbol{\gamma}^T \mathbf{u} \\ &\text{subject to} \quad \mathbf{u} \in \mathcal{P}(\mathbf{H}). \end{aligned} \tag{2.11}$$

The solution of the above LP problem corresponds to a vertex of the fundamental polytope that minimizes the cost function. Since the fundamental polytope has both integral and nonintegral vertices, with the integral vertices corresponding exactly to the codewords of $\mathcal{C}$ [11, 12], if the LP solver outputs an integral solution, it must be a valid codeword and is guaranteed to be an ML solution. Since the fundamental polytope is a function of the parity-check matrix $\mathbf{H}$ used to represent the code $\mathcal{C}$, different parity-check matrices for $\mathcal{C}$ may have different fundamental polytopes. Therefore, a given code has many possible LP-based relaxations, and some may be better than others when used for LP decoding.

The fundamental polytope can also be described by a set of linear inequalities, obtained as follows. First of all, for a point $\mathbf{u}$ within the fundamental polytope, it should satisfy the box constraints such that $0 \leqslant u_i \leqslant 1$, for $i = 1, \ldots, n$. For each row $j = 1, \ldots, m$ of the parity-check matrix, corresponding to a check node in the associated Tanner graph, the linear inequalities used to form the fundamental polytope $\mathcal{P}(\mathbf{H})$ are given by

$$\sum_{i \in \mathcal{V}} (1 - u_i) + \sum_{i \in \mathcal{N}_j \setminus \mathcal{V}} u_i \geqslant 1, \ \forall \mathcal{V} \subseteq \mathcal{N}_j, \text{with } |\mathcal{V}| \text{ odd} \tag{2.12}$$

where for a set $\mathcal{X}$, $|\mathcal{X}|$ denotes its cardinality. It is easy to see that (2.12) is equivalent to

$$\sum_{i \in \mathcal{V}} u_i - \sum_{i \in \mathcal{N}_j \setminus \mathcal{V}} u_i \leqslant |\mathcal{V}| - 1, \ \forall \mathcal{V} \subseteq \mathcal{N}_j, \text{with } |\mathcal{V}| \text{ odd.} \tag{2.13}$$

Note that, for each check node j, the corresponding inequalities in (2.12) or (2.13) and the linear box constraints exactly describe the convex hull of the set $\mathcal{C}_j$.

The linear constraints in (2.12) (and therefore also (2.13)) are referred to as *parity inequalities*, which are also known as *forbidden set inequalities* [1]. It can be easily verified that these linear constraints are equivalent to the original parity-check constraints when each u_i takes on binary values only.

Proposition 2.2 (Theorem 4 in [11]) *The parity inequalities of the form* (2.12) *derived from all rows of the parity-check matrix* $\mathbf{H}$ *and the box constraints completely describe the fundamental polytope* $\mathcal{P}(\mathbf{H})$.

With this, LP decoding can also be formulated as follows

$$\begin{aligned} \text{minimize} \quad & \boldsymbol{\gamma}^T \mathbf{u} \\ \text{subject to} \quad & 0 \leqslant u_i \leqslant 1, \ \text{for all} \ i; \\ & \sum_{i \in \mathcal{V}} (1 - u_i) + \sum_{i \in \mathcal{N}_j \setminus \mathcal{V}} u_i \geqslant 1 \\ & \text{for all} \ j, \mathcal{V} \subseteq \mathcal{N}_j, \text{with} \ |\mathcal{V}| \ \text{odd.} \end{aligned} \tag{2.14}$$

In this dissertation, we refer to the above formulation of LP decoding problem based on the fundamental polytope of the original parity-check matrix as the *original* LP decoding.

Bibliography

[1] C.E. Shannon, “A mathematical theory of communication,” *Bell Systems Tech. J.*, vol. 27, pp. 379–423 and pp. 623–656, 1948.

[2] R. G. Gallager, “Low-density parity-check codes.” *IRE Trans. Information Theory*, vol. 8, pp. 21–28, Jan. 1962.

[3] R. G. Gallager, *Low-Density Parity-Check Code*, MIT Press, Cambridge, MA, 1963.

[4] D. J. C. MacKay and R. M. Neal, “Good codes based on very sparse matrices,” in *Cryptography and Coding: Lecture Notes in Computer Science*, vol. 1025. C. Boyd, Ed. Heidelberg: Springer Berlin, 1995, pp. 100–111.

[5] N. Wiberg, “Codes and decoding on general graphs,”Ph. D. dissertation, Linköping University, 1996.

[6] R. Hamming, “Error detecting and correcting codes,” *Bell System Tech. J.*, pp. 147–160, April 1950.

[7] R. M. Tanner, “A recursive approach to low complexity codes,” *IEEE Trans. Information Theory*, vol. 27, no. 9, pp. 533–547, September 1981.

[8] F. Kschischang, B. Frey, and H. Loeliger, “Factor graphs and the sum-product algorithm,” *IEEE Trans. Information Theory*, vol. 47, no. 2, pp. 498–519, February 2001.

[9] J. Pearl, *Probabilistic Reasoning in Intelligent Systems*, San Mateo, CA: Morgan Kaufmann, 1988.

[10] J. Chen, A. Dholakia, E. Eleftheriou, M. Fossorier, and X. Hu, “Reduced-complexity decoding of LDPC codes,” *IEEE Trans. Communications*, vol. 53, no. 8, pp. 1288–1299, August 2005.

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

[12] P. O. Vontobel and R. Koetter, “Graph-cover decoding and finite-length analysis of message-passing iterative decoding of LDPC codes,” CoRR, arxiv.org/abs/cs.IT/0512078.

Chapter 3

Efficient Algorithms to Find All Small Error-Prone Substructures

Low-density parity-check (LDPC) codes have been the focus of much research over the past decade thanks to their near Shannon limit performance and to their efficient message-passing (MP) decoders. However, some properties of LDPC codes are still not fully understood. One of the most important open problems is the phenomenon of the error floor. Roughly speaking, the error floor is an abrupt change in the error-rate performance of a MP decoder in the high SNR region. It is known that the error floor performance of a finite-length LDPC code is dominated by certain error-prone substructures (EPS) within the Tanner graph used to represent the code. For the binary erasure channel (BEC), the class of EPS known as *stopping sets* determine the error floor performance as well as the error-rate performance [1]. For general memoryless binary-input output-symmetric (MBIOS) channels, the EPS that dominate the error floor performance have not yet been fully characterized, although some classes of EPS have been identified and, depending upon the context and performance analysis techniques being used, given names such as *near-codewords* [2], *trapping sets* [3], *absorbing sets*, *fully absorbing sets* [4], and *pseudocodewords* [5].

Generally, the error floor of LDPC codes occurs at a bit error rate (BER) of around 10^{-5} to 10^{-7}, but many important applications, such as data storage, require a very low error floor, in the range of 10^{-12} to 10^{-15}. This makes estimating error floors by Monte Carlo simulation virtually impossible. With importance sampling, the error floor of LDPC codes can be semi-analytically estimated by characterizing and enumerating the trapping sets [3]. Moreover, with even a partial list of trapping sets of an LDPC code, new LDPC codes having a low error floor can be designed [6]; alternatively post-processing decoding can be used to reduce the error floor [7].

In [8], it was proven that exhaustively enumerating all small *k-out trapping sets* in an arbitrary, finite-length LDPC code is NP-complete. This result was extended to different kinds of EPS [9]. In spite of the hardness of finding all EPS, some practical search algorithms have been proposed recently for limited-length LDPC codes. In [10], a non-exhaustive search algorithm based upon the impulse approach was proposed. It can find a portion of the trapping sets efficiently, but can not guarantee a complete enumeration of all the trapping sets. An exhaustive search algorithm based on a *branch-and-bound* approach was proposed in [11], which can find all small-sized fully absorbing sets for LDPC codes with moderate length (< 1000).

In this paper, we introduce two practical, exhaustive search algorithms based upon the branch-and-bound principle which was previously used in [11] and [12]. During the bounding step, we formulate two new and different linear programming (LP) problems for our EPS enumeration algorithm and FAS enumeration algorithm, respectively. By solving the LP problem, we can decide the minimum size of EPS and FAS under given constraints, and thereby bound the search. In comparison to the EPS search algorithm in [10], our EPS enumeration algorithm can find all small EPS rather than only a subset of them. Moreover, as we will show later in this paper, our FAS enumeration algorithm is more efficient than a recently proposed exhaustive FAS search algorithm [11]. Finally, for quasi-cyclic (QC) LDPC codes, we can further improve the efficiency for both algorithms by exploiting the cyclicity properties of the code.

The remainder of the paper is organized as follows. In Section 3.1, we give some basic notation and definitions. Section 3.2 describes our proposed exhaustive search algorithms. In Section 3.3, we apply our algorithms to some well-documented LDPC codes, and compare the efficiency and accuracy with that of other algorithms. Section 3.4 concludes the paper.

3.1 Error-Prone Substructures

Several classes of EPS have been identified in the literature. On the BEC, it is stopping sets, clearly defined substructure within the Tanner graph of an LDPC code, that dominate the error-rate performance of the message-passing (MP) decoder: the decoder fails if and only if the set of erased bits contains a stopping set [1]. However, for general MBIOS channels such as the binary symmetric channel (BSC) and the additive white Gaussian noise channel (AWGNC), the exact substructures that cause the error-floor are still not fully understood. The most prevalent term for such a substructure is *trapping set* (TS), which is operationally defined as a subset of variable nodes that is susceptible to errors under iterative decoding. However, this concept depends on both the channel and the decoding algorithm.

To facilitate our discussion, we define the term *error-prone substructure* from a graph-theoretic perspective, independent of the channel and the decoder.

Definition 3.1 *A subset of V is an (a, b)* error-prone substructure (EPS) *if in the induced subgraph, there are a variable nodes and b odd degree check nodes as neighbors. Correspondingly, a binary indicator vector $\mathbf{t} \in \{0,1\}^n$ is also called an EPS if it has weight a and its syndrome $\mathbf{s}$ has weight b, where $\mathbf{s} = \mathbf{Ht}$.*

Remark 3.1 Unlike the conventional definition of trapping set, the subgraph corresponding to an EPS is not necessarily a connected graph.

For a set of variable nodes $A \subseteq V$, denote by $E_A \subseteq C$ and $O_A \subseteq C$ the sets of check nodes connected to A an even number or an odd number of times, respectively. Let $E_A(v)$ and $O_A(v)$ be the sets of neighboring check nodes of v in E_A and O_A, respectively.

Definition 3.2 *A set of variable nodes $A \subseteq V$ is an* absorbing set *(AS) if $|E_A(v)| > |O_A(v)|$ for all $v \in A$; and a set of variable nodes $A \subseteq V$ is a* fully absorbing set *(FAS) if $|E_A(v)| > |O_A(v)|$ for all $v \in V$. An AS (or FAS) is also called an (a, b) AS (or FAS) where $a = |A|$ and $b = |O_A|$.*

From Definitions 4.1 and 3.2, it is easy to see that all trapping sets are EPS, and all the absorbing sets and fully absorbing sets defined in [4] are also EPS. Actually, any subset of variable nodes can be considered as an (a, b) EPS, but the EPS of most interest are those with small a and b values, and/or small b/a ratio.

3.2 Algorithm Description

In the following description of the proposed algorithms, we use an extension of the notion of indicator vector that we call a *candidate vector*. Specifically, a candidate vector $\mathbf{f} = [f_1, \ldots, f_n]^T \in \{0, 1, *\}^n$ is a vector of length n, where 1 and 0 indicate that the corresponding VN is included or excluded from a subset of V, respectively, and $*$ means the position is *unconstrained*, i.e., not yet set to a ”0” or ”1” value. Let $\mathsf{supp}(\mathbf{f}) = \{i : f_i = 1\}$ be the support set of the candidate vector $\mathbf{f}$, and $\mathrm{wt}(\mathbf{f}) = |\mathsf{supp}(\mathbf{f})|$ be the weight of $\mathbf{f}$. Let $\mathbf{f}|_{p,0}$ and $\mathbf{f}|_{p,1}$ be the candidate vectors obtained by setting the p-th position of $\mathbf{f}$ to 0 and 1 while keeping other positions unchanged, respectively. Let $\mathcal{S}$ be the set of row indexes whose corresponding check nodes have at least one unconstrained neighbor VN and an odd number of 1-valued neighbor VNs, according to $\mathbf{f}$. Finally, let $\mathcal{T}$ be the set of column indexes corresponding to the unconstrained positions of $\mathbf{f}$. Define the operation $\otimes$ between a binary vector $\mathbf{h} = [h_1, \ldots, h_n] \in \{0,1\}^n$ and $\mathbf{f}$ as

$$\mathbf{h} \otimes \mathbf{f} = \begin{cases} 0, & \text{if } \exists i : f_i = * \text{ and } h_i = 1; \\ \sum_{i:h_i=1} h_i f_i \text{ (over } \mathbb{F}_2), & \text{otherwise.} \end{cases}$$

Note that $\mathbf{h} \otimes \mathbf{f} \in \{0,1\}$. We can extend this operation to a matrix and a vector, i.e., applying $\otimes$ on each row of the matrix, and get a binary vector, such that $\mathbf{H} \otimes \mathbf{f} = \{\mathbf{h}_1 \otimes \mathbf{f}, \ldots, \mathbf{h}_m \otimes \mathbf{f}\}^T \in \{0,1\}^m$ where $\mathbf{h}_i$ is the i-th row of $\mathbf{H}$. Hence, the weight of $\mathbf{H} \otimes \mathbf{f}$ is the number of unsatisfied CNs which have no unconstrained neighbor VNs.

3.2.1 EPS Search Algorithm

The proposed exhaustive search procedure for EPS is given in Algorithm 5.1 , which finds all (a, b) EPS such that $a \leqslant a_{\max}$ and $b \leqslant b_{\max}$. The object denoted STACK is a last-in first-out (LIFO) stack that keeps candidate vectors. The tightness of the computed lower bound in line 9 and the selection of an unconstrained position in line 11 determine the number of candidate sets (or the number of nodes in the binary search tree in the branch-and-bound approach) considered in the algorithm.

3.2.1.1 Computing Lower Bound in line 9 (Bound Step)

The quantity $\omega(\mathbf{f})$ is a lower bound on the minimum increase in the value of $\alpha + \beta$, given all fixed positions in $\mathbf{f}$, where α and β are defined in line 4. Define $\hat{\mathbf{H}} \triangleq [\mathbf{H}]_{\mathcal{S},\mathcal{T}}$, the submatrix of $\mathbf{H}$ consisting of elements in the rows and columns determined by the sets $\mathcal{S}$ and $\mathcal{T}$, respectively. If set $\mathcal{S}$ is empty, we set lower bound $\omega(\mathbf{f})$ to be zero. Let $\mathbf{x} = \{x_1, \ldots, x_{|\mathcal{T}|}\}$ be the optimization variables. The lower bound on the increase of $\alpha + \beta$ can be found by solving the following integer programming (IP) problem:

$$\begin{aligned} \min \quad & \sum_{i \in \mathcal{T}} x_i \\ \text{s. t.} \quad & \hat{\mathbf{H}}\mathbf{x} \geqslant (1, 1, \ldots, 1)^T, \\ & x_i \in \{0, 1\}. \end{aligned} \tag{3.1}$$

The constraint $\hat{\mathbf{H}}\mathbf{x} \geqslant (1, 1, \ldots, 1)^T$ arises from the fact that each row of $\hat{\mathbf{H}}$ corresponds to an unsatisfied CN which has at least one unconstrained neighbor VN. Consider an unsatisfied CN

Algorithm 3.1 Exhaustive Search Algorithm for EPS

Input: parity-check matrix $\mathbf{H}$, integer $a_{\max}$ and $b_{\max}$.

Output: the exhaustive list $\mathcal{L}$ of all (a, b) EPS with $a \leqslant a_{\max}$ and $b \leqslant b_{\max}$.

```
1:  $\mathcal{L} \leftarrow \varnothing$, $\mathsf{STACK} \leftarrow \varnothing$, and push $(*, \ldots, *)$ into $\mathsf{STACK}$.
2:  **while** $\mathsf{STACK} \neq \varnothing$ **do**
3:      $\mathbf{f} \leftarrow$ pop $\mathsf{STACK}$.
4:      $\alpha \leftarrow |\mathsf{supp}(\mathbf{f})|$ and $\beta \leftarrow$wt$(\mathbf{H} \otimes \mathbf{f})$.
5:      **if** $\alpha \leqslant a_{\max}$ and $\beta \leqslant b_{\max}$ **then**
6:          **if** there is no unconstrained position in **f** **then**
7:              $\mathcal{L} \leftarrow \mathcal{L} \cup \{\mathsf{supp}(\mathbf{f})\}$
8:          **else**
9:              Compute lower bound $\omega(\mathbf{f})$.
10:             **if** $\alpha + \beta + \omega(\mathbf{f}) \leqslant a_{\max} + b_{\max}$ **then**
11:                 Choose position $p \in \{i : f_i = *\}$.
12:                 Push $\mathbf{f}|_{p,0}$ and $\mathbf{f}|_{p,1}$ into $\mathsf{STACK}$.
13:             **end if**
14:         **end if**
15:     **end if**
16: **end while**
```

which has only one unconstrained neighbor VN. Such a CN will increase the value of $\alpha + \beta$ by one, because if we include this VN into the set the value of α will increase by one, and if we do not include it, the value of β will increase by one. Therefore, the purpose of the IP problem above is to find a lower bound of the minimum number of unconstrained VNs that could eliminate all the unsatisfied CNs in $\mathcal{S}$ if they are set to the value 1.

The IP problem (3.1) can be relaxed by allowing x_i to be a nonnegative real value, transforming the optimization problem into an LP problem:

$$\begin{aligned} \min \quad & \sum_{i \in \mathcal{T}} x_i \\ \text{s. t.} \quad & \hat{\mathbf{H}}\mathbf{x} \geqslant (1, 1, \ldots, 1)^T, \\ & \mathbf{x} \geqslant 0. \end{aligned} \tag{3.2}$$

Therefore, the lower bound $\omega(\mathbf{f})$ is the optimal value of the objective function of LP problem (3.2), i.e., $\omega(\mathbf{f}) = \sum\limits_{i \in \mathcal{T}} x_i^* = \sum\limits_{i \in \mathcal{T}^+} x_i^*$, where $\mathcal{T}^+ = \{i : i \in \mathcal{T}, x_i^* > 0\}$.

3.2.1.2 Position Selection in line 11 (Branch Step)

Following [12], the principle of the position selection in line 11 is to find a VN that has the largest number of neighbor CNs connected to the smallest number of unconstrained VNs. Therefore, we choose position $p \in \mathcal{T}^+$ such that

$$(N_p(1), N_p(2), \ldots, N_p(d_p)) \succeq (N_q(1), N_q(2), \ldots, N_q(d_q)) \tag{3.3}$$

for all $q \in \mathcal{T}^+$ and $q \neq p$, where $N_q(k)$ is the number of neighboring CNs of the q-th VN that are connected to k unconstrained VNs, and $\succeq$ denotes *lexicographical order* under which $(x_1, x_2, \ldots, x_l) \succeq (y_1, y_2, \ldots, y_l)$ if $x_j > y_j$ and $x_i = y_i$ for $1 \leqslant i \leqslant j-1$, or $x_i = y_i$ for $1 \leqslant i \leqslant l$. If $d_p \neq d_q$, the vector in (3.3) corresponding to the smaller of the two is padded with zeros. If there is more than one position p satisfying (3.3), then we choose the one with greater column weight in $\hat{\mathbf{H}}$. If $\mathcal{S} = \emptyset$, we choose an unconstrained position with maximum column weight in $\mathbf{H}$.

As we noted previously, our definition of EPS may include unconnected subgraphs especially when b_{max} increases. However, since most EPS of interested are of small b where the number of such ”degenerated” substructures is small if they exist, it is easy to add an additional step after each EPS being found to filter out these undesired substructures, and the computational complexity of such filtering is negligible.

3.2.2 FAS Search Algorithm

The exhaustive search procedure for FAS is given in Algorithm 5.2 , which is based on the EPS search algorithm. The key differences are in line 9 and line 10, where we compute and apply a lower bound, $\mu(\mathbf{f})$, on the minimum number of unconstrained positions in $\mathbf{f}$ that should be set to 1 in order to get a valid FAS.

In line 9, before computing the lower bound, we first check the unsatisfied CNs with all known neighboring VNs to see whether the subgraph is a FAS, i.e., no VN has more unsatisfied neighboring CNs than satisfied neighboring CNs. If the definition of FAS has already been violated by $\mathbf{f}$, the lower bound $\mu(\mathbf{f})$ is set to be $a_{\max}$, implying that the condition in line 10 will not hold; otherwise, the following approach is applied to compute the lower bound.

Let $C_1(\mathbf{f})$ be the set of all unsatisfied CNs represented by $\mathbf{f}$, i.e., the set of CNs that connect to VNs in $\mathsf{supp}(\mathbf{f})$ an odd number of times. Let $U_k \triangleq N(k) \cap C_1(\mathbf{f})$ be the set of unsatisfied neighboring CNs of the k-th VN. The lower bound $\mu(\mathbf{f})$ can be found by solving the following LP problem:

$$\begin{aligned} \min \quad & \sum_{i\in\mathcal{T}} x_i \\ \text{s. t.} \quad & \theta(x_k) + \sum_{j\in U_k} \sum_{i\in\mathcal{N}_j\cap\mathcal{T}/\{k\}} x_i \\ & \geqslant 1 + |U_k| - \lceil\frac{d_k}{2}\rceil \quad \text{for all } k \in V, \\ & x_i \geqslant 0 \quad \text{for all } i \in \mathcal{T} \end{aligned} \tag{3.4}$$

where $\theta(x_k)$ is defined as

$$\theta(x_k) = \begin{cases} (|U_k| - |S_k|)\, x_k, & \text{if } k \in \mathcal{T}; \\ 0, & \text{otherwise,} \end{cases}$$

and S_k is the set of satisfied check nodes whose only unknown neighboring variable node is v_k.

Note that we can remove the constraints in (3.4) if the right-hand side is less than or equal to zero. The optimal value of the LP problem is the lower bound, i.e., $\mu(\mathbf{f}) = \sum_{i\in\mathcal{T}^+} x_i^*$, and the subsequent position selection step in line 11 is the same as in Algorithm 5.1 .

From the definition of FAS, each VN has to connect to more satisfied neighboring CNs than the unsatisfied. Given candidate vector $\mathbf{f}$ and a VN position k, $1 \leqslant k \leqslant n$, the VN has $|U_k|$ unsatisfied neighbor CNs, and we have to have

$$d_k - |U_k| > |U_k| \tag{3.5}$$

Algorithm 3.2 Exhaustive Search Algorithm for FAS

Input: parity-check matrix $\mathbf{H}$, integer $a_{\max}$ and $b_{\max}$.

Output: the exhaustive list $\mathcal{L}$ of all (a, b) FAS with $a \leqslant a_{\max}$ and $b \leqslant b_{\max}$.

```
1: $\mathcal{L} \leftarrow \varnothing$, $\mathsf{STACK} \leftarrow \varnothing$, and push $(*, \dots, *)$ into $\mathsf{STACK}$.
2: while $\mathsf{STACK} \neq \varnothing$ do
3:   $\mathbf{f} \leftarrow$ pop $\mathsf{STACK}$.
4:   $\alpha \leftarrow |\mathsf{supp}(\mathbf{f})|$ and $\beta \leftarrow$wt$(\mathbf{H} \otimes \mathbf{f})$.
5:   if $\alpha \leqslant a_{\max}$ and $\beta \leqslant b_{\max}$ then
6:     if there is no unconstrained position in f then
7:       $\mathcal{L} \leftarrow \mathcal{L} \cup \{\mathsf{supp}(\mathbf{f})\}$
8:     else
9:       Compute lower bound $\mu(\mathbf{f})$.
10:      if $\alpha + \mu(\mathbf{f}) \leqslant a_{\max}$ then
11:        Choose position $p \in \{i : f_i = *\}$.
12:        Push $\mathbf{f}|_{p,0}$ and $\mathbf{f}|_{p,1}$ into $\mathsf{STACK}$.
13:      end if
14:    end if
15:  end if
16: end while
```

in order to get a valid FAS. If this inequality does not hold for a VN, we have to set itself and/or some of its neighboring VNs (2-step neighbors on the Tanner graph) to one to let the inequality hold. The LP problem in (3.4) gives a lower bound on the minimum number of unconstrained positions in $\mathbf{f}$ needed to further be set to one in order to have (3.5) hold for every VN.

By comparing Algorithm 5.1 and Algorithm 5.2 , we can see that the value of $a_{\max} + b_{\max}$ dominates the computational complexity of Algorithm 5.1 in the search for general EPS, whereas in Algorithm 5.2 , where we restrict the search to FAS, it is $a_{\max}$ alone that determines the running time.

3.2.3 The Exhaustiveness of Proposed Algorithms

The searches for EPS and FAS preformed in Algorithm 5.1 and Algorithm 5.2 can be seen as a search on a binary tree, whose root is the first VN position that the algorithms pick to start with. If no branch-and-bound is performed, the leaves of the binary tree are all 2^n binary vectors of length n. The proposed exhaustive search algorithms traverse the binary tree in an efficient way such that they compute the lower bound on $a + b$ for EPS or a for FAS on all children of the current node. If the lower bound on such parameters exceeds thresholds on a certain node, the algorithms cut the branch from this node off the binary tree. With branch-and-bound, the search algorithms avoid going through all 2^n leaves of the binary tree, but still traverse the whole binary tree; therefore, our proposed algorithms is an exhaustive search. Note that, using different ways to compute the lower bounds and to select next position p does not affect the exhaustiveness of the search algorithm, it only affects the efficiency of the search.

3.2.4 Efficiency Improvement for QC Codes

Consider a vector of length βt where both β and t are positive integers,

$$\mathbf{v} = (\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_t) = (v_{1,1}, \dots, v_{1,\beta}, \dots, v_{t,1}, \dots, v_{t,\beta}).$$

Let $\mathbf{v}_i^{(l)} = (v_{i,\beta-l+1}, \dots, v_{i,\beta-l})$ be the (right) cyclic-shift of $\mathbf{v}_i$ by l positions. The vector $\mathbf{v}^{[l]} = (\mathbf{v}_1^{(l)}, \dots, \mathbf{v}_t^{(l)})$ is called the *t-section* cyclic-shift of $\mathbf{v}$.

Definition 3.3 *Let β and t be positive integers. A linear block code $\mathcal{C}_{qc}$ of length βt is called a* quasi-cyclic (QC) *code if the following conditions hold: (1) each codeword consists of t sections of β bits each; and (2) every t-section cyclic-shift of a codeword in $\mathcal{C}_{qc}$ is also a codeword in $\mathcal{C}_{qc}$. Such a QC code is also called a t-*section *QC code.*

Table 3.1: (a, b) EPS enumerators for the (155,64) Tanner code

(a,b)	count	(a,b)	count	(a,b)	count
(5,3)	155	(6,4)	2790	(7,3)	930
(8,2)	465	(8,4)	14415	(9,3)	5580
(10,2)	1395	(10,4)	83235	(11,3)	17360
(12,2)	930	(12,4)	36280	(13,3)	43245

From the definition above, we can see that if an indicator vector s of length βt corresponds to an (a, b) EPS (or FAS) of a t-section QC code, the t-section cyclic-shift of s is also an (a, b) EPS (or FAS). In this case, we can initialize the STACK with t candidate vectors, $\mathbf{f}^i = (f^i_1, \ldots, f^i_{\beta t})$, $0 \leqslant i \leqslant t-1$, where

$$f^i_j = \begin{cases} 0, & \text{if } j = k\beta + l + 1,\ 0 \leqslant k \leqslant i-1,\ 0 \leqslant l \leqslant \lambda \\ 1, & \text{if } j = i\beta + \lambda + 1 \\ *, & \text{otherwise} \end{cases}$$

and $\lambda \triangleq \max\left(\lceil \frac{\beta}{a_{\max}} \rceil - 1, 0\right)$.

Moreover, we can further reduce the search space by initialize the STACK with more properly designed candidate vectors. The EPS (or FAS) found using such an initialization are then quasi-cyclically shifted and the number of distinct ones are counted. In comparison to the initialization proposed in [12], our method generates initialization vectors with more fixed positions, and therefore, it more efficiently reduces the search space.

3.3 Numerical Results

In this section, we provide some numerical results for several well-documented LDPC codes. By comparing our results with existing results in the literature, we demonstrate the completeness and efficiency of our proposed search algorithms.

3.3.1 Results of EPS Search Algorithm

3.3.1.1 The (3,5)-Regular (155,64) QC Tanner Code [13]

In Table 3.1, we list the number all (a, b) EPS where $a \leqslant 13$ and $b \leqslant 4$. To demonstrate the efficiency of the proposed algorithm, consider the case of finding all EPS for $a_{\max} = 9$ and

Table 3.2: (a,b) EPS enumerators for the M816 code

(a,b)	EPS	FAS	analytical search [15]
(3,3)	132	126	132
(4,2)	3	3	0
(4,4)	3459	1350	1372
(5,3)	120	86	41

Table 3.3: (a,b) EPS enumerators for the M1008 code

(a,b)	EPS	FAS	analytical search [15]
(3,3)	165	153	165
(4,2)	6	6	0
(4,4)	3701	1130	1215
(5,3)	160	92	14

$b_{\max} = 3$. With a brute force exhaustive search, we would be required to check $\sum_{i=1}^{9} \binom{155}{i} \approx 1.2 \times 10^{14}$ sets in order to find all EPS of size up to 9. In contrast, the proposed algorithm only searchers through 6.8×10^6 candidate vectors if initialized with an empty STACK. If we further take advantage of the QC property of the Tanner (155,64) code and initialize the STACK with 5 candidate vectors, the number of candidate vectors the algorithm has to consider drops to 6.0×10^5.

3.3.1.2 The (3,6)-Regular (816,408) M816 and (1008,504) M1008 LDPC Codes [14]

M816 and M1008 are two random (3,6)-regular LDPC codes generated by MacKay. In Table 3.2 and Table 3.3, we compare the number of EPS and FAS ($a_{\max} \leqslant 5$) found by our proposed algorithms with the number of trapping sets found by the analytical search method in [15] for the M816 and M1008 code, respectively. According to the definition of trapping set in [15] (i.e., if an error vector stays the same after one iteration of Gallager B decoding, it corresponds to a trapping set), although all EPS in the tables above can not be correctly decoded by the Gallager B decoder, some of them are not trapping sets since they decode to error vectors corresponding to trapping sets of other sizes. Interestingly, for the parameters listed in Table 3.2 and Table 3.3, the FAS correspond precisely to the set of all trapping sets. The analytical method

Table 3.4: (a, b) EPS enumerators for (256,128) LDPC code

(a, b)	exhaustive	non-exhaustive [10]
(5,3)	32	32
(7,3)	32	32
(8,2)	32	32
(8,4)	544	540
(9,3)	192	191
(10,4)	2304	1600
(11,3)	128	117
(12,2)	32	32
(12,4)	6304	1645
(13,3)	864	457
(14,2)	96	91
(14,4)	28896	2466
(19,1)	64	38

Table 3.5: Simulation time comparison for FAS of PEGR504 code ($a_{\max} = b_{\max} = 5$)

Algorithm 5.1	Algorithm 5.2	EFSA [11]
59 min	3 min	6 hr 55 min

in [15], which counts cycles and cycle interactions on the Tanner graph, is intended to enumerate all the trapping sets, but as can be seen, the results obtained are not always accurate.

3.3.1.3 The (256, 128) CCSDS QC-LDPC Code [10]

Table 3.4 compares the number of EPS found by our algorithm with that found by a non-exhaustive approach proposed in [10], which tries to find as many EPS as possible. This code has irregular variable degree, such that some VNs are of degree 3 and others have degree 5. We can see that the non-exhaustive approach works well when searching for EPS with small a and b, but the accuracy decreases dramatically as the size of EPS increases.

3.3.2 Results of FAS Search Algorithm

Due to space limitations and the fact that the FAS form a subset of EPS, we do not include here the number of FAS found using our exhaustive search algorithm. We provide instead only some numerical results that demonstrate the efficiency of our proposed algorithms. The algorithms were implemented with VC++ and GLPK as the LP solver [16]. Note that we *do not* use any coding optimization techniques to accelerate the execution of our C++ code. The running times were obtained on a standard desktop PC with a 2.67GHz processor.

For the comparison, we examined the PEGR504 code, which is a (3,6)-regular (504,252) LDPC code taken from [14]. We found the same number of FAS for this code as was reported in [11]. Table 3.5 compares the running times of our proposed algorithms with that of the exhaustive FAS search algorithm (EFSA) in [11] for the case where $a_{\max} = b_{\max} = 5$. We can see that both of our proposed algorithms are significantly faster than EFSA. Algorithm **5.2** , in particular, reduced the execution time by more than two orders of magnitude. The improvement on efficiency comes from two parts. One is that our algorithms have a more efficient ways to estimate the lower bound and to select next constraint position. The other is that, when computing the lower bound, our algorithms only need to solve an LP problem which can be efficiently solved, but EFSA has to solve an IP problem which is generally more computational complicated and requires more CPU running time. Note that the PEGR504 code is not quasi-cyclic, so we could not take the advantage of QC codes as described in Section 3.2.4. However, we can expect even more efficiency improvement with our proposed algorithms for QC-LDPC codes.

3.4 Conclusion

In this chapter, we proposed efficient, exhaustive search algorithms for EPS and FAS. By properly initializing the algorithms, we further improved the execution-time efficiency for QC-LDPC codes. A comparison of our results to those obtained with previously proposed full and partial search algorithms confirms the relative efficiency of our algorithms, while identifying weaknesses in some of the others. The dissertation author was the primary investigator and author of this paper.

Acknowledgment

This chapter is in part a reprint of the material in the papers: Xiaojie Zhang and Paul H. Siegel, “Efficient algorithms to find all small error-prone substructures in LDPC codes,” in *Proc. IEEE Global Communication Conference*, Houston, TX, Dec. 5–9, 2011, pp. 1–6.

Bibliography

[1] C. Di, D. Proietti, E. Telatar, T. Richardson, and R. Urbanke, “Finite length analysis of low-density parity-check codes on the binary erasure channel,” *IEEE Trans. Inf. Theory*, vol. 48, no. 6, pp. 1570–1579, Jun. 2002.

[2] D. MacKay and M. Postol, “Weakness of Margulis and Ramanujan-Margulis low-density parity check codes,” *Electron. Notes Theor. Comp. Sci.*, vol. 74, 2003.

[3] T. Richardson, “Error-floors of LDPC codes,” *Proc. of the 41st Annual Allerton Conference on Communication, Control, and Computing, Monticello*, IL, Oct. 1–3, 2003, pp. 1426–1435.

[4] L. Dolecek, Z. Zhang, V. Anantharam, M. Wainwright, and B. Nikolic, “Analysis of absorbing sets and fully absorbing sets of array-based LDPC codes,” *IEEE Trans. Inform. Theory*, vol. 56, no. 1, pp. 181–201, Jan. 2010.

[5] P. O. Vontobel and R. Koetter, “Graph-cover decoding and finite-length analysis of message-passing iterative decoding of LDPC codes,” CoRR, arxiv.org/abs/cs.IT/0512078.

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

[7] E. Cavus and B. Daneshrad, “A performance improvement and error floor avoidance technique for belief propagation decoding of LDPC codes,” in *Proc. IEEE Intl. Symp. on Pers., Indoor and Mobile Radio Comm.*, Berlin, Germany, Sep. 2005, pp. 2386–2390.

[8] C. Wang, S. R. Kulkarni, and H. V. Poor, “Finding all small error-prone substructures in LDPC codes,” *IEEE Trans. Inform. Theory*, vol. 55, no. 5, pp. 1979–1999, May 2009.

[9] A. McGregor and O. Milenkovic, “On the hardness of approximating stopping and trapping sets,” *IEEE Trans. Inform. Theory*, vol. 56, no. 4, pp. 1640–1650, Apr. 2010.

[10] S. Abu-Surra, D. DeClerq, D. Divsalar, and W. Ryan, “Trapping set enumerators for specific LDPC codes,” *Information Theory and Applications Workshop (ITA)*, La Jolla, CA, Jan. 31–Feb. 5, 2010.

[11] G. Kyung and C. Wang, “Exhaustive search for small fully absorbing sets and the corresponding low error-floor decoder,” in *Proc. IEEE Int. Symp. Inform. Theory (ISIT)*, Austin, TX, Jul. 2010, pp. 739–743.

[12] E. Rosnes and Ø. Ytrehus, "An efficient algorithm to find all small-size stopping sets of low-density parity-check matrices," *IEEE Trans. Inform. Theory*, vol. 55, no. 9, pp. 4167–178, Sep. 2009.

[13] R. M. Tanner, D. Sridhara, and T. Fuja, "A class of group-structured LDPC codes," in *Proc. Int. Symp. Communication Theory and Applications (ISCTA)*, Ambleside, U.K., Jul. 2001, pp. 365–369.

[14] D. J. C. MacKay, *Encyclopedia of Sparse Graph Codes*. [Online]. Available: http://www.inference.phy.cam.ac.uk/mackay/codes/data.html

[15] S. K. Chilappagari, S. Sankaranarayanan, and B. Vasic, "Error floors of LDPC codes on the binary symmetric channel," in *Proc. IEEE Int. Conf. on Commun. (ICC)*, Istanbul, Turkey, Jun. 2006, pp. 1089–1094.

[16] GNU Linear Programming Kit, http://www.gnu.org/software/glpk

Chapter 4

Quantized Iterative Message Passing Decoders with Low Error Floor for LDPC Codes

One common way to improve the error floor performance of LDPC codes has been to redesign the codes to have Tanner graphs with large girth and without small EPSs [1–3]. However, for LDPC codes that have been standardized, approaches are needed that do not modify the codes. In the literature, many modifications to the iterative MP decoding algorithms have been proposed in order to improve high SNR performance, such as averaged decoders [4], reordered decoders [5, 6], and decoders with post processing [7–11]. In [4], the authors noticed that the emergence of errors in EPSs is heuristically related to a sudden magnitude change in the values of certain variable nodes (VNs). Hence, it was proposed to average the messages in a belief-propagation (BP) decoder over several iterations to avoid such sudden changes and therefore slow down the convergence rate for variable nodes in a trapping set and decrease the frequency of trapping set errors. Another heuristic approach is to process messages based on the order of node reliabilities computed at each iteration [5], and it was suggested that the scheduled decoders are able to resolve some standard trapping set errors [6]. Although these general approaches are capable of improving the average error rate performance to some extent, the resulting decoders still fail on small EPSs and their effect on the error floor is not significant.

To further improve the error floor behavior, decoders that make use of the prior knowledge of some small size EPSs are designed to reduce the decoding failures due to such EPSs. In [7] and [8], the authors proposed a post-processing decoder that matches the configuration of unsatisfied check nodes (CNs) to trapping sets in a precomputed list after conventional MP decoding has failed. The size and completeness of the trapping set list directly affect the performance gain of such decoders, but to obtain a complete list of small trapping sets of a given LDPC code is generally quite computationally complex. A symbol-selecting post-processing technique was also developed in [9]. It saturates the channel messages on a set of selected variable nodes at each stage after the conventional MP algorithms fails. In [10], Han and Ryan proposed a bi-mode erasure decoder that combines several problematic check nodes into a generalized constraint processor, to which a corresponding maximum a posteriori (MAP) algorithm, such as the BCJR algorithm, is then applied. Another post-processing approach that utilizes the graph-theoretic structure of absorbing sets, proposed in [11], adjusts the appropriate messages in the iterative MP decoding once the decoder enters and remains in the absorbing set of interest.

All the above approaches either change the message update rules of MP decoders or require extra processing steps after conventional MP decoding fails, both of which increase the decoding complexity relative to the original iterative MP algorithms. Moreover, the post-processing approaches that require prior knowledge of the set of EPSs causing the error floor are only effective when applied to LDPC codes whose EPSs have been carefully studied.

In fixed-point implementation of iterative MP decoding, efforts have also been made to improve the error-rate performance in the waterfall region and/or error-floor region by optimizing parameters of uniform quantization [12–15]. In [12], Zhao *et al.* studied the effect of message clipping and uniform quantization on the performance of the min-sum decoder in the waterfall region and heuristically optimized the number of quantization bits and the quantization step size for selected LDPC codes. In [13], a dual-mode adaptive uniform quantization scheme was proposed to better approximate the log-tanh function used in sum-product algorithm (SPA) decoding. Specifically, for magnitudes less than 1, all quantization bits were used to represent the fractional part; for magnitudes greater than or equal to 1, all bits were dedicated to the representation of the integer part. In [14, 15], Zhang *et al.* proposed a conceptually similar idea to increase precision in the quantization of the log-tanh function. Uniform quantization was applied to messages generated by both variable nodes and check nodes, but the quantization step sizes used in the two cases were separately optimized. We note, however, that none of these modified quantization schemes were primarily intended to significantly increase the saturation level, or range, of quantized messages, and in their reported simulation results, error floors can still be clearly observed.

It was first noticed in [16] that the high error floors associated with certain EPSs of some LDPC codes are closely related to the saturation level imposed on messages passed in the SPA decoder. In this work, we investigate the cause of error floors in binary LDPC codes from the perspective of the MP decoder implementation, with special attention to limitations that decrease the numerical accuracy of messages passed during decoding. We show that, under certain assumptions, the EPSs which are commonly associated with high error floors of some LDPC codes will not trap iterative MP decoders and cause high error floors if messages are accurately represented and there is no limitation on the number of iterations. Based upon an analysis of the growth rate of messages outsides an EPS in an idealized scenario, we propose a novel quasi-uniform quantization method that captures the essence of messages in different ranges of reliability. The proposed quantization method has an extremely large saturation level which prevents iterative MP decoders from being trapped by an EPS. This property, to the best of our knowledge, distinguishes if from other quantization techniques for iterative MP decoding that have appeared in the literature. With the new quantization method, it is possible to have a fixed point implementation of iterative MP decoders that achieves low error floors without an additional post-processing stage or a modification of either the decoding update rules or the graphical code representation upon which the iterative MP decoder operates. We present simulation results for min-sum decoding, SPA decoding, and some of their variants, that demonstrate a significant reduction in the error floors of two representative LDPC codes, with no increase in decoding complexity.

The remainder of the chapter is organized as follows. Section 4.1 gives some notation and definitions used throughout this chapter. In Section 4.2, we analytically investigate the impact that message quantization can have on MP decoder performance and the error floor phenomenon. In Section 4.3, we propose an enhanced quantization method intended to overcome the limitations of traditional quantization rules. In Section 4.4, we incorporate the new quantizer into SPA and min-sum decoding and, through computer simulation of several LDPC codes known for their high error floors, demonstrate the significant improvement in error-rate performance that this new quantization approach can afford. Section 4.5 concludes the chapter.

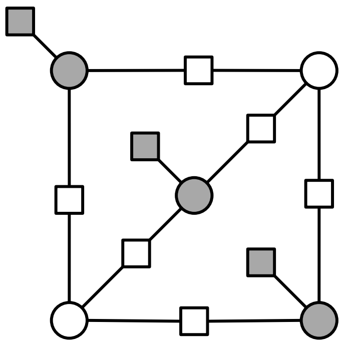
4.1 Notation and Definitions

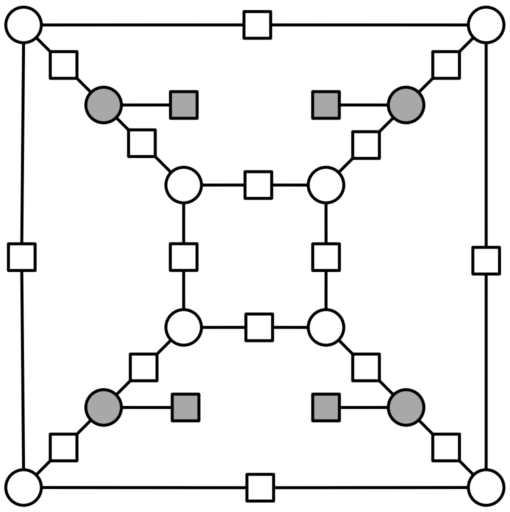
The study of the phenomenon of error floors began shortly after LDPC codes were rediscovered about a decade ago. It has been shown that the EPSs known as *stopping sets* cause the error floor in the binary erasure channel (BEC), and such EPSs have a clear combinatorial description. Enumeration of these structures makes it possible to accurately estimate the error floor [17]. However, for other MBIOS channels such as the binary symmetric channel (BSC) and the additive white gaussian noise channel (AWGNC), it is more difficult to establish the relationship between EPSs and error floors. In [18], it was first pointed out that the *near-codewords* caused error floors in simulations of Margulis and Ramanujan-Margulis LDPC codes on the AWGNC. The term *trapping set* proposed by Richardson [19] is operationally defined as a subset of variable nodes (VNs) that is susceptible to errors under a certain iterative MP decoder over an MBIOS channel. Hence, this concept depends on both the channel and the decoding algorithm. In [20], the error floor is associated with some combinatorial substructures within the Tanner graph, named *absorbing sets*, which are defined independently of the channel. The absorbing sets correspond to a particular type of near codewords or trapping sets that are stable under bit-flipping operations. All these EPSs have been believed to be the cause of error floors, and for some LDPC codes, techniques such as importance sampling used to estimate the error floor are based on the probability of decoding failures on such EPSs [19, 22]. In this section, we will show that under certain assumptions about the correctness of variable nodes outside a given EPS in the Tanner graph, conventional iterative decoders that accurately represent messages will eventually correct errors supported by the EPS.

To facilitate our discussion, we define a substructure called an *absolute trapping set* from a purely graph-theoretic perspective, independent of the channel and the decoder.

Definition 4.1 *A* stopping set *of size a is a configuration of a variable nodes such that the induced subgraph has no check nodes of degree-one. An (a, b)* trapping set *is a configuration of a variable nodes, for which the induced subgraph is connected and has b odd-degree check nodes. If the induced subgraph of an (a, b) trapping set does not contain a stopping set and has at least one check node of degree one, it is called an* absolute trapping set*.*

In the literature, all trapping sets of interest that cause the error floor of an LDPC code are of size smaller than the minimum stopping set size of the code, since otherwise the stopping sets would be the dominate contributor to the error floor [17]. By requiring at least one check node of degree one, we exclude stopping sets from our definition of absolute trapping set. As we will discuss later in this section, these degree-one check nodes are essential because they are able to pass correct extrinsic messages into the trapping set. To the best of our knowledge, almost all trapping sets of interest in the literature are absolute trapping sets. For example, both of the well-known (5,3) trapping sets in the Tanner code of length 155, the notorious (12,4) trapping sets in the (2640,1320) Margulis code, and the (5,5) trapping set in some codes of variable-degree five are all elementary trapping sets. This can be seen in Fig. 4.1, where check nodes of


(a) $a = 5, b = 3$


(b) $a = 12, b = 4$

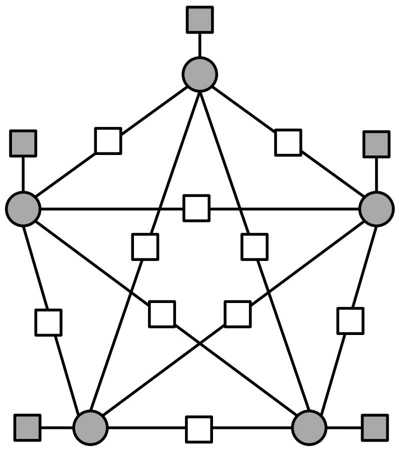

(c) $a = 5, b = 5$

Figure 4.1: Examples of (a, b) absolute trapping sets. Check nodes in set C_1 and variable nodes in set V_1 are shown shaded.

degree-one are shaded. Unless otherwise indicated, all trapping sets referred to in this chapter are elementary trapping sets, as well.

In analogy to the definition of *computation tree* in [23], we define a *k-iteration computation tree* as follows.

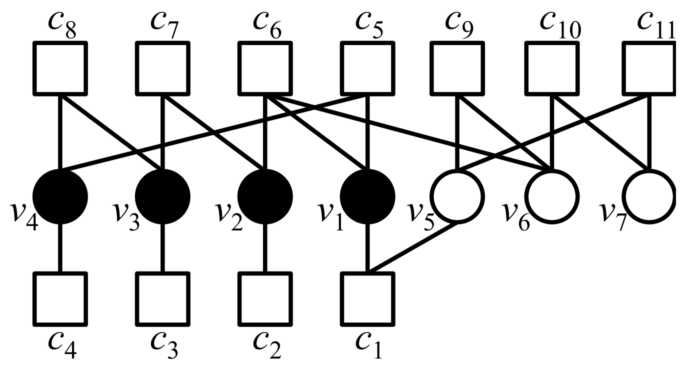
Definition 4.2 *A* k-iteration computation tree $T_k(v)$ *for an iterative decoder in the Tanner graph G is a tree graph constructed by choosing variable node $v \in V$ as its root and then recursively adding edges and leaf nodes to the tree that participate in the iterative message-passing decoding during k iterations. To each vertex that is created in $T_k(v)$, we associate the corresponding node update function in G.*

Let S be the induced subgraph of an (a, b) trapping set contained in G, with VN set $V_S \subseteq V$ and CN set $C_S \subseteq C$. Let set $C_1 \subseteq C_S$ be the set of degree-one CNs in the subgraph S, and let set $V_1 \subseteq V_S$ be the set of neighboring VNs of CNs in C_1. In the (5,3), (12,4), and (5,5) trapping sets shown in Fig. 4.1, the check nodes in set C_1 and the variable nodes in set V_1 are shaded. We refer to a message on an edge adjacent to VN v as a *correct* message if its sign reflects the correct value of v, and as an *incorrect* message, otherwise. Let $D(u)$ be the set of all descendants of the vertex u in a given computation tree.

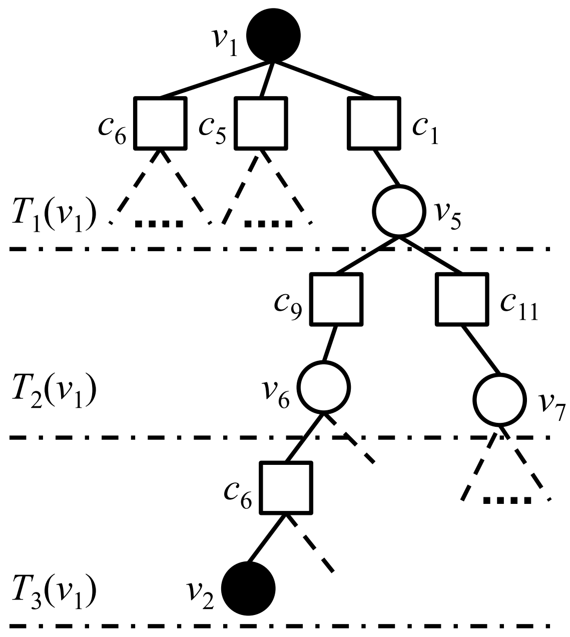
Definition 4.3 *Given a Tanner graph G and an induced subgraph S of a trapping set, a variable node $v \in V_1$ is said to be k-*separated *if, for at least one of its neighboring degree-one check node $c \in C_1$ in S, no variable node $v' \in V_S$ belongs to $D(c) \subset T_k(v)$. If every $v \in V_1$ is k-separated, the induced subgraph S is said to satisfy the k-*separation assumption.

In Fig. 4.2(a), we show the graph of a $(4, 4)$ trapping set and some of its neighboring nodes. The set of VNs in the trapping set is $V_S = \{v_1, v_2, v_3, v_4\}$, represented as solid black cycles. The set of CNs in the trapping set is $C_S = \{c_i\}$, $1 \leqslant i \leqslant 8$. In this trapping set, every VN has a neighboring degree-one CN, i.e., $V_1 = V_S$, and $C_1 = \{c_1, c_2, c_3, c_4\}$. For example, the 3-iteration computation tree of VN v_1 is shown in Fig. 4.2(b). It can be verified from this computation tree that v_1 is 2-separated but not 3-separated, because $v_2 \in V_S$ is a descendant of c_1 in $T_3(v_1)$, but not in $T_2(v_1)$. It is worth noting that whether or not a trapping set satisfies the k-separation assumption depends on the Tanner graph outside the trapping set, not the trapping set itself.

We want to point out that the k-separation assumption is much weaker than the isolation assumption in [24]. The separation assumption here only applies to the VNs that have neighboring degree-one CNs in the induced subgraph S, and these neighboring degree-one CNs do



(a) $(4, 4)$ trapping set and part of its neighboring nodes



(b) Computation tree with root v_1

Figure 4.2: Example of a $(4, 4)$ trapping set and its corresponding computation tree.

not have any VNs from the trapping set as their descendants in the corresponding k-iteration computation tree. With the separation assumption, the descendants of $c \in C_1$ are separated from all the nodes in the trapping set, meaning that the incorrect messages passed in the trapping set do not affect the extrinsic messages sent towards c in the computation tree.

4.2 Error Floors of LDPC Codes

4.2.1 Trapping Sets and Min-Sum Decoding

To get further insight into the connection between trapping sets and decoding failures of iterative MP decoders, we first consider a simple iterative MP decoder, the min-sum (MS) decoder.

Theorem 4.4 *Let G be the Tanner graph of a variable-regular LDPC code that contains a subgraph S induced by a trapping set. When S satisfies the k-separation assumption and when the messages from the BSC to all VNs outside S are correct, the min-sum decoder can successfully correct all erroneous VNs in S, provided k is large enough.*

Proof: Assume VN $v_r \in V_1 \subseteq S$ is k-separated and the corresponding k-iteration computation tree is $T_k(v_r)$. Let $c_r \in C_1$ be the neighboring degree-one CN of v_r in S. From the separation assumption and the assumed correctness of channel messages for VNs outside S, all descendants of c_r in $T_k(v_r)$ receive correct initial messages from the BSC. Like the LLRs of the BSC outputs, all the initial messages in the decoder, L_i^{ch}, $1 \leqslant i \leqslant n$, have the same magnitude. Denote the subtree starting with CN c_r as $T(c_r)$. With the VN/CN update rules of the min-sum decoder, we analyze the messages sent from the descendants of c_r in $T(c_r)$. First, according to the CN update rule described in (2.5), all messages received by a VN from its children CNs in $T(c_r)$ must have the same sign as the message received from the channel by this VN, because all the messages passed in $T(c_r)$ are correct. Therefore, the outgoing message from any VN v_i to its parent CN c_j in $T(c_r)$ satisfies the following equality

$$\begin{aligned} |L_{i\to j}| &= \left| L_i^{ch} + \sum_{j' \in \mathcal{N}_i \backslash j} L_{j' \to i} \right| \\ &= \left| L_i^{ch} \right| + \sum_{j' \in \mathcal{N}_i \backslash j} \left| L_{j' \to i} \right| . \end{aligned} \tag{4.1}$$

Moreover, since the LDPC code considered is variable-regular and all the channel messages from the BSC have the same magnitude, it can be shown that, for the min-sum decoder, all incoming messages received by a VN from its children CNs in $T(c_r)$ must have the same magnitude as well. Therefore, the messages sent from VNs of the same level in the computation tree $T(c_r)$ have the same magnitude. Let $|L_l|$ be the magnitude of the messages sent by the VNs whose shortest path to a leaf VN contains l CNs in $T(c_r)$. Hence, $|L_0|$ is the magnitude of messages sent by leaf VNs, as well as the magnitude of channel inputs. Then, we have

$$\begin{aligned} |L_l| &= |L_0| + (d_v - 1)|L_{l-1}| \\ &> (d_v - 1)|L_{l-1}| \\ &> (d_v - 1)^l |L_0| \end{aligned} \tag{4.2}$$

where d_v is the variable node degree. Hence, it can be seen that the magnitudes of messages sent towards the root CN c_r of the computation tree $T(c_r)$ grow exponentially, with $d_v - 1$ as the base, in every upper VN level. Therefore, for $l \leqslant k$, the magnitude of the message sent from c_r to its parent node v_r, the k-separated root VN of $T_k(v_r)$, in the l-th iteration is greater than $(d_v - 1)^l |L_0|$.

Now, let us consider a branch of the computation tree that starts from another child CN $c' \in C_S \setminus C_1$ of the root v_r in $T_k(v_r)$, denoted by by $T(c')$. Suppose the message received by v_r from c' after l iterations, denoted by L'_l, has a different sign than the message received from $c_r \in C_1$; otherwise, v_r would already be corrected. Since the induced subgraph of the trapping set is connected, there exists an integer t such that any t-level subtree starting from a VN $v \in S$ in $T(c')$, i.e., a subtree with t levels of VNs, must have at least one CN from the set C_1 as its descendant. Since the number of VNs in the trapping set is a, any two VNs in the induced subgraph are connected by a path of length less than $2a$. Hence, it is obvious that $t \leqslant 2a$, and a tighter upper bound is $t \leqslant \frac{2\log a}{\log(d_v - 1)}$. Of course, the exact value of t depends on the structure of the trapping set. For min-sum decoding, every CN in a computation tree takes the minimum magnitude of messages received from its children VNs. Therefore, each t-level subtree can be considered as a "super-node" with $(d_v - 1)^t$ children VNs, and at least one of these children VNs has descendants that all receive correct messages from the channel; this means that at least one of the incorrect messages going into the super-node would be canceled out by one or more correct messages. So if the output message, L_{out}, of such a super-node is incorrect, its magnitude satisfies

$$|L_{out}| < ((d_v - 1)^t - 1)|L_{in}| + |\bar{L}_{ch}|, \tag{4.3}$$

where $|L_{in}|$ is the largest magnitude of all incoming incorrect messages, and the second term $|\bar{L}_{ch}| \triangleq |L_0| \sum_{i=0}^{t-1} (d_v - 1)^i$ is an upper bound on the sum of all channel input LLRs to the VNs in the t-level subtree. Note that the leaf VNs of such t-level subtrees are not necessarily the leaf

VNs of $T_k(v_r)$. Thus, we can upper bound the magnitude of the incorrect message sent from c' to v_r after l iterations by

$$\begin{aligned} |L_l'| &< |L_0| \cdot \left[(d_v-1)^t - 1\right]^{\lceil l/t \rceil} + |\bar{L}_{ch}| \sum_{i=0}^{\lceil l/t \rceil - 1} \left[(d_v-1)^t - 1\right]^i \\ &< |\bar{L}_{ch}| \cdot |L_0| \cdot \left[(d_v-1)^t - 1\right]^{\lceil l/t \rceil} \end{aligned} \tag{4.4}$$

where $\lceil x \rceil$ is the smallest integer greater than x.

Therefore, by taking the logarithm of $|L_l|$ in (4.2) and $|L_l'|$ in (4.4), respectively, we have

$$\begin{aligned} \log|L_l| &> \log|L_0| + l\log(d_v-1) \\ &= \log|L_0| + l \cdot \tfrac{1}{t} \cdot \log(d_v-1)^t, \end{aligned} \tag{4.5}$$

and

$$\begin{aligned} \log|L_l'| < &\ \log|\bar{L}_{ch}| + \log|L_0| + \lceil l/t \rceil \log\left[(d_v-1)^t - 1\right] \\ < &\ \log|\bar{L}_{ch}| + \log|L_0| + \log\left[(d_v-1)^t - 1\right] \\ &+ l \cdot \tfrac{1}{t} \cdot \log\left[(d_v-1)^t - 1\right]. \end{aligned} \tag{4.6}$$

Note that the first term in (4.5) and the first three terms in (4.6) are constants and independent of the number of iterations l.

Since $\log(d_v-1)^t > \log\left[(d_v-1)^t - 1\right]$, if l is large enough and there is no limitation imposed on the magnitude of messages, it is easy to see from (4.5) and (4.6) that $|L_l|$ would be greater than $|L_l'|$ multiplied by any constant. This means that the correct messages coming from outside of the trapping set to VNs in V_1 through their neighboring CNs in C_1 will eventually have greater magnitude than the sum of incorrect messages from other neighboring CNs, i.e., $|L_l| > (d_v-1)|L_l'|$. By letting the number of iterations further grow, the correct messages would eventually be large enough to correct all erroneous VNs in the trapping set. ■

Remark 4.1 Note that the upper bound in (4.4) is extremely loose, and for most small-size trapping sets, the upper bound is generally less than $|L_0|(d_v-2)^l$.

Corollary 4.5 *Let G be the Tanner graph of a variable-regular LDPC code that contains a subgraph S induced by a trapping set. When S satisfies the k-separation assumption and the channel messages from the AWGNC to all VNs outside S are correct, the min-sum decoder can successfully correct all erroneous VNs in S, provided k is large enough.*

Proof: The input LLRs from the AWGNC to the decoder could have different magnitudes, so we define $|L_{\min}|$ and $|L_{\max}|$ to be the minimum and maximum magnitude of all LLRs, respectively.

Then, the bounds on $\log |L_l|$ in (4.5) and on $\log |L'_l|$ in (4.6) can be extended to the AWGNC setting as

$$\log |L_l| > \log |L_{\min}| + l \cdot \frac{1}{t} \cdot \log(d_v - 1)^t,$$

and

$$\begin{aligned}\log |L'_l| < \ & \log |\bar{L}_{ch}| + \log |L_{\max}| + \log \left[(d_v - 1)^t - 1\right] \\ & + l \cdot \tfrac{1}{t} \cdot \log \left[(d_v - 1)^t - 1\right].\end{aligned}$$

Since $\log |L_{\min}|$ and $\log |L_{\max}|$ are both constant terms and do not change as l increases, similar to the BSC case, we can also conclude that the correct messages from outside the trapping set will eventually have greater magnitude than the incorrect messages within the trapping set. ■

Theorem 4.4 and its corollary can be extended to both attenuated min-sum (AMS) and offset min-sun decoding. For AMS, if the attenuation factor is a constant, the proof of Theorem 4.4 can be directly applied. As for OMS, the proof follows the proof of Theorem 4.8 in the next subsection.

4.2.2 Trapping Sets and Sum-Product Algorithm Decoding

In this subsection, we further extend Theorem 4.4 to sum-product algorithm decoding. The optimality criterion in the design of the SPA decoder is symbol-wise maximum a *posteriori* probability (MAP), and it is an optimal symbol-wise decoder on Tanner graphs without cycles.

Recall that, in the CN update rule of SPA decoding, the message sent from CN j to VN i is computed as

$$L_{j \to i} = 2\tanh^{-1}\left(\prod_{i' \in \mathcal{N}_j \backslash i} \tanh \frac{L_{i' \to j}}{2} \right). \tag{4.7}$$

In practical implementations of the SPA, the following equivalent CN update rule is often used

$$L_{j \to i} = \left[\prod_{i' \in \mathcal{N}_j \backslash i} \text{sign}(L_{i' \to j}) \right] \cdot \phi^{-1} \left(\sum_{i' \in \mathcal{N}_j \backslash i} \phi(|L_{i' \to j}|) \right) \tag{4.8}$$

where $\phi(x) = -\log[\tanh(x/2)]$ and $\phi^{-1}(x) = \phi(x)$, as shown in Fig. 4.3. In some fixed-point implementations, in order to have better approximation, different look-up tables could be used to compute $\phi(x)$ and $\phi^{-1}(x)$ [15].

We want to point out that the hyperbolic tangent function, $\tanh(x)$, has numerical saturation problems when computed with finite precision. For example, in double-precision floating-point computer implementation (64-bit IEEE 754) [26], it can be shown that $\tanh(x/2)$ would

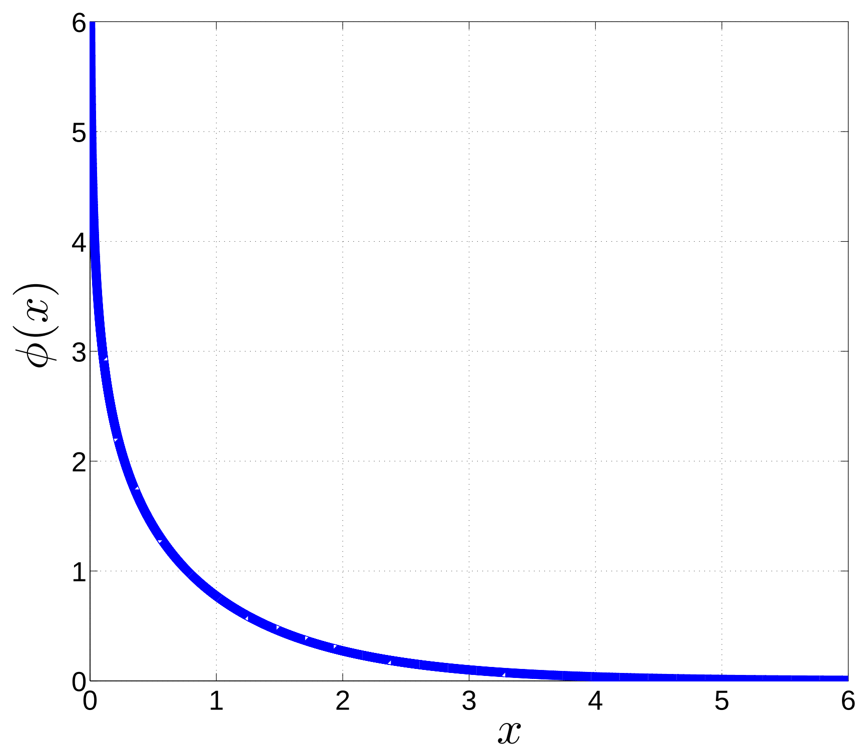


Figure 4.3: A plot of the $\phi(x)$ function.

be rounded to 1 when $x > 38$, meaning that $\phi^{-1}(\phi(x)) = \infty$ for $x > 38$ [27]. In order to avoid such problems that can arise from limited precision, thresholds on the magnitudes of messages must be applied in simulation studies [15].

In order to maintain the performance advantage of SPA decoding over min-sum decoding, the quantization method has to preserve the self-inverse property of the $\phi(x)$ function and to accurately compute the CN update function in (4.8). However, from Fig. 4.3, we can see that it is difficult to have a good approximation of the $\phi(x)$ function with limited resolution, because this requires both fine precision and large range. Efforts have been made to design quantization methods that work effectively with the $\phi(x)$ function. For example, a variable-precision quantization scheme proposed in [13] uses larger quantization step size for magnitudes greater than 1, and smaller step size for magnitudes less than 1. An adaptive uniform quantization method proposed in [14] uses different quantization step sizes for the outputs of the $\phi(x)$ and the $\phi^{-1}(x)$ function in (4.8). Fig. 4.3 clearly shows that, if the output of the $\phi(x)$ function is quantized with finite precision ϵ, inputs greater than $\phi^{-1}(\epsilon)$ can not be distinguished, and $\phi^{-1}(\epsilon)$ is quite small even for extremely fine precision, e.g., $\phi^{-1}(10^{-6}) \approx 14.5$. Hence, the largest supported magni-

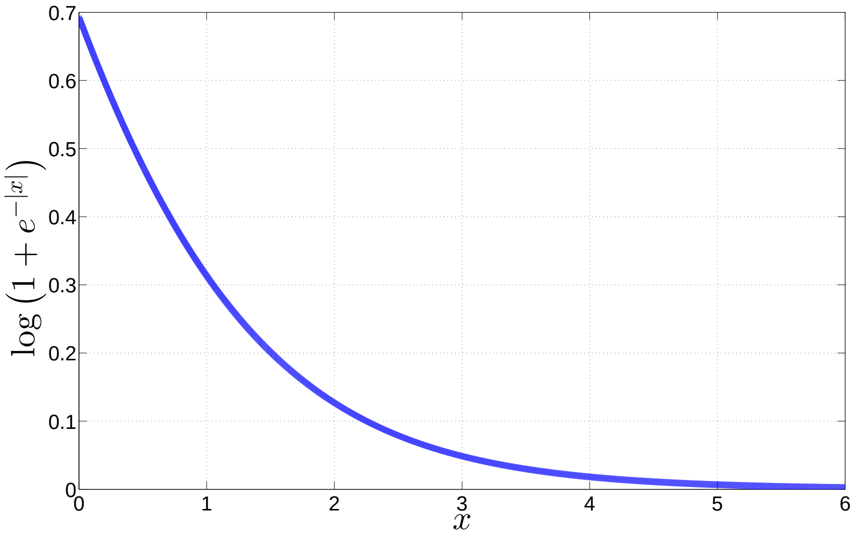


Figure 4.4: A plot of function $\log\left(1+e^{-|x|}\right)$.

tude during decoding depends on the finest precision of quantization. This means increasing the quantization range without improving the precision is not beneficial.

In order to avoid dealing with the $\phi(x)$ function, a variety of other CN update rules, most of which are approximations to the SPA, have been proposed. Some of these approximation are based on the following equivalent version of the SPA CN update rule represented by (4.7) or (4.8),

$$L_{j\to i} = \underset{i'\in\mathcal{N}_j\setminus i}{\boxplus} L_{i'\to j} \tag{4.9}$$

where $\boxplus$ is the pairwise "box-plus" operator defined as

$$\begin{aligned} U\boxplus V &= \log\left(\frac{1+e^{U+V}}{e^U+e^V}\right) \\ &= \text{sign}(U)\text{sign}(V)\cdot\{\min(|U|,|V|)+s(|U|,|V|)\} && (4.10) \\ &= \text{sign}(U)\text{sign}(V)\min(|U|,|V|)+s(U,V) && (4.11) \end{aligned}$$

and

$$s(x,y) = \log\left(1+e^{-|x+y|}\right) - \log\left(1+e^{-|x-y|}\right). \tag{4.12}$$

The proof of equivalence between (4.7) and (4.9) can be found in [28]. We call such an implementation *box-plus* SPA decoding. The formulation above does not have the precision problem that (4.7) and (4.8) have, and, in fact, in 64-bit double-precision floating-point implementation, the maximum magnitude of a message that can be supported is approximately 1.8×10^{308}, which is the largest double-precision value supported by the IEEE 754 standard.

Unlike the $\phi(x)$ function, the function $\log\left(1+e^{-|x|}\right)$, as shown in Fig. 4.4, can be well quantized or approximated with piecewise linear functions [27–29]. Moreover, if the term $s(x,y)$ is omitted, the box-plus SPA becomes the min-sum algorithm, and if we replace this term with a constant, we get the offset min-sum algorithm. As we will show later, the magnitude of the function $s(x,y)$ is upper bounded by a constant.

From the CN update rule described in (4.9) and (4.11), it can be seen that box-plus SPA decoding can be considered as min-sum decoding with a small correction factor. The following lemma relating CN messages in min-sum and SPA decoding was first shown in [30] for SPA decoding with the CN update equation described in (4.7). Although we know that the box-plus SPA and the SPA using the $\phi(x)$ function are equivalent, we include here, for the sake of completeness, a proof of the lemma for box-plus SPA, as follows.

Lemma 4.6 *Given the same input messages, the CN output in box-plus SPA decoding has the same sign as, but smaller magnitude, than that of min-sum decoding.*

Proof: Since $s(x,y) = 0$ when $xy = 0$, it can be seen from (4.10) that the lemma is true if the inequality $\min(x,y) + s(x,y) > 0$ holds for any positive real values x and y. Assuming $x \geqslant y > 0$, the following inequalities are equivalent

$$\begin{aligned} \min(x,y) + s(x,y) >& \quad 0 \\ \Leftrightarrow \quad \log e^{y} + \log \tfrac{1+e^{-x+y}}{1+e^{-x-y}} >& \quad 0 \\ \Leftrightarrow \quad e^{y} + e^{-x} - 1 - e^{-x+y} >& \quad 0 \\ \Leftrightarrow \quad (e^{y}-1)(1-e^{-x}) >& \quad 0. \end{aligned}$$

Since $e^{y} > 1$ and $e^{-x} < 1$, the final inequality holds. This proves the lemma. ■

From Lemma 4.6, we can see that SPA decoding can be thought of as min-sum decoding with a small correction factor that does not change the sign of the CN output of the min-sum algorithm.

Lemma 4.7 *For any real values x and y, the magnitude of $s(x,y)$ is bounded by a constant. Specifically, $-\log 2 < s(x,y) < \log 2$.*

Proof: If x and y have the same sign (i.e., $xy > 0$), we have $|x+y| > |x-y|$ and hence

$s(x,y) < 0$. Without loss of generality, if we assume $x \geqslant y > 0$, then

$$\begin{aligned} s(x,y) &= \log \frac{1+e^{-x-y}}{1+e^{-x+y}} \\ &= \log \frac{e^x + e^{-y}}{e^x + e^y} \\ &\geqslant \log \frac{e^x + e^{-x}}{e^x + e^x} \\ &> \log \frac{1}{2}. \end{aligned}$$

Therefore, when $xy > 0$, we have $-\log 2 < s(x,y) < 0$. When $xy < 0$, it can also be similarly shown that $0 < s(x,y) < \log 2$, and for the case $xy = 0$, we have $s(x,y) = 0$. ■

As we discussed earlier, no matter how one designs the fixed-point implementation of the original SPA using the $\phi(x)$ function, or even with the floating-point implementation, the function $\left|x - \phi^{-1}(\phi(x))\right|$ is unbounded. Even if we saturate both the input and the output of the $\phi(x)$ function, the value of $\left|x - \phi^{-1}(\phi(x))\right|$ is still unbounded and linear in x. Therefore, the CN output of a practical implementation of (4.7) or (4.8) can significantly differ from the true computed value. However, since box-plus SPA decoding can be considered as min-sun decoding with a correction factor, the implementation error mainly comes from the computation and quantization of the correction factor, which is a small bounded value, as shown in Lemma 4.7.

With Lemma 4.6 and Lemma 4.7, we can now extend Theorem 4.4 to SPA decoding.

Theorem 4.8 *Let G be the Tanner graph of a variable-regular LDPC code that contains a subgraph S induced by a trapping set. When S satisfies the k-separation assumption and all VNs outside S receive the correct transmitted symbols from the BSC, with proper scaling of all initial LLRs, the SPA decoder can successfully correct all of the VNs in S that receive incorrect symbols from the BSC, provided k is large enough.*

Proof: From Lemma 4.6, we know that SPA decoding can be considered as min-sum decoding with a small correction factor which does not change the sign of the original min-sum output. Moreover, the magnitude of the CN output of SPA decoding is always less than or equal to that of min-sum decoding. To compute the output for a CN of degree d_c, the box-plus SPA uses the pairwise box-plus operation (4.11) at most $\log(d_c - 1)$ times. Hence, the difference between output messages of the SPA and the min-sum algorithm is upper bounded by $\bar{s} \triangleq \lceil \log(d_c - 1) \rceil \cdot \log 2$, where $\lceil x \rceil$ is the smallest integer that is greater than x.

By applying an approach similar to that used in the proof of Theorem 4.4, we can lower bound the magnitude of messages L_l in SPA decoding as follows

$$\begin{aligned}
|L_l| &> |L_0| + (d_v-1)\left(|L_{l-1}| - \bar{s}\right) \\
&> (d_v-1)\left(|L_{l-1}| - \bar{s}\right) \\
&> (d_v-1)^l |L_0| - \bar{s}\sum_{i=1}^{l}(d_v-1)^i \\
&= (d_v-1)^l |L_0| - \bar{s}(d_v-1)\frac{(d_v-1)^l-1}{(d_v-1)-1} \\
&= (d_v-1)^l\left(|L_0| - \frac{d_v-1}{d_v-2}\bar{s}\right) + \bar{s}\frac{d_v-1}{d_v-2}.
\end{aligned}$$

Since all input messages to the decoder from the BSC have the same magnitude, if we scale the magnitudes of all initial messages such that

$$|L_0| > \frac{d_v-1}{d_v-2}\bar{s} = \frac{d_v-1}{d_v-2}\cdot\lceil\log(d_c-1)\rceil\cdot\log 2, \tag{4.13}$$

then the magnitudes of messages sent towards c_r in the computation tree $T_k(v_r)$ grow exponentially in the number of iterations, with base $d_v - 1$. Hence, using the same reasoning as in the proof of Theorem 4.4, it can be shown that, if k is large enough and there is no limit on the magnitudes of messages, the correct messages outside the trapping set eventually overcome the incorrect messages passed within the trapping set, thereby correcting all erroneous VNs in the trapping set. ■

Remark 4.2 As will be shown in the simulation results, linear scaling of the input LLRs to the SPA decoder will indeed affect the decoding performance, because the correction factor $s(x, y)$ is not linear in either x or y.

Remark 4.3 If an offset min-sum decoder has a constant or bounded offset value, the proof of Theorem 4.8 can be directly applied to obtain a similar result.

The proof of Theorem 4.8 can be adapted to prove an analogous result for the AWGNC, stated in the following corollary.

Corollary 4.9 *Let G be the Tanner graph of a variable-regular LDPC code that contains a subgraph S induced by a trapping set. When S satisfies the k-separation assumption and the messages from the AWGNC to all VNs outside S are correct, with proper scaling of all initial*

LLRs, the SPA decoder can successfully correct all erroneous VNs in S, provided k is large enough.

Proof: In analogy to the proof of Lemma 4.5, let $|L_0|$ be the minimum magnitude of all input LLRs to the decoder from the AWGNC, and linearly scale magnitudes of all input messages such that (4.13) is satisfied. Then, using reasoning along the lines of the proof of Theorem 4.8, we can show that the magnitudes of correct messages outside the trapping set still grow exponentially with $d_v - 1$ as the base, and eventually they correct all erroneous VNs in the trapping set. ■

In [16], by applying a linear system model and density evolution, the authors obtained some statistical lower bounds on the exponential growth rate of correct messages in SPA decoding on the AWGNC.

For most LDPC codes, the trapping sets typically satisfy the k-separation assumption only for small values of k. Nevertheless, as described more fully in Section 4.4, in our 64-bit double-precision floating-point computer simulations of min-sum decoding and box-plus SPA decoding applied to several LDPC codes traditionally associated with high error floors, we have not observed, in tens of billions of channel realizations of both the BSC and the AWGNC, any decoding failure in which the error patterns correspond to the support of a small trapping set. Moreover, when we force every VN in a trapping set to be in error and all other VNs to be correct, the floating-point decoders can successfully decode, whereas a decoder implementation that limits the magnitude of messages can not resolve the errors in the trapping set and fails to decode to the correct codeword.

4.3 New Quantized Decoders with Low Error Floors

In hardware implementations of iterative MP decoding algorithms, a modest number of bits are used to represent messages, precluding high-resolution representation of messages over a large numerical range. As reported in the literature, most hardware implementations and their computer-based simulations use some form of uniform quantization. We will consider uniform quantizers with quantization step Δ and q-bit representation of quantization levels, with one of the q bits denoting the sign. The quantized values are $l\Delta$ for $-N \leqslant l \leqslant N$, where $N = 2^{q-1} - 1$.

It has been noticed that error floors can be lowered when more bits are used to represent messages, either by reducing the step size or by increasing the saturation level [15]. However, as explained earlier in Section 4.2.2, the maximum magnitude of supported input to the $\phi(x)$

function depends on the quantization resolution at the output. Therefore, uniform quantization can not support large values as inputs of the $\phi(x)$ function due to the limited output resolution. Hence, substantially increasing the saturation level was not considered as a feasible mechanism for reducing error floors.

Even for min-sum decoding, which does not have such a numerical problem caused by the $\phi(x)$ function, there appears to be no quantization method in the literature that reduces error floors specifically by means of a deliberate, significant extension of the quantization range, although it has been noticed that the saturation level could affect the error floor performance of min-sum decoding. Although the authors of [12] claimed that four-bit uniform quantization suffices to obtain close to the performance of floating-point min-sum decoder, which they consider to be the ideal min-sum decoder, for most codes over a wide range of SNR, our simulation results, shown in the next section, demonstrate that uniform-quantized min-sum decoding and its variants still have high error floors even with eight quantization bits.

In the remaining part of this section, we propose a novel quantization method that substantially extends the quantization range while maintaining precision in the representation of values of small magnitude. Moreover, we will show that, even with a small number of quantization bits, the new method provides significant improvement over the floating-point implementations.

As shown in the proof of Theorems 4.4 and 4.8 and their corollaries, when a trapping set satisfies the k-separation assumption for a large value of k, the magnitudes of correct messages outside the trapping set grow exponentially in the number of iterations. Therefore, it would be desirable for the message quantizer to capture, at least to some extent, the exponential increase of these message magnitudes while retaining precision in the representation of messages with smaller magnitudes. To this end, we propose a new $(q+1)$*-bit quasi-uniform* quantization method that adds an additional bit to q-bit uniform quantization to indicate a change of step size in the representation of large message magnitudes. Hence, the messages after quantization will belong to an alphabet of size $2^{q+1}-1$. Specifically, the $(q+1)$-bit quasi-uniform quantization rule is given by

$$Q(L) = \begin{cases} (l, 0), \quad \text{if } l\Delta - \frac{\Delta}{2} < L \leqslant l\Delta + \frac{\Delta}{2} \\ (N, 0), \quad \text{if } N\Delta - \frac{\Delta}{2} < L < dN\Delta \\ (-N, 0), \text{if } -dN\Delta < L \leqslant -N\Delta + \frac{\Delta}{2} \\ (r, 1), \quad \text{if } d^r N\Delta \leqslant L < d^{r+1}N\Delta \\ (-r, 1), \text{if } -d^{r+1}N\Delta < L \leqslant -d^r N\Delta \\ (N+1, 1), \quad \text{if } L \geqslant d^{N+1}N\Delta \\ (-N-1, 1), \quad \text{if } L \leqslant -d^{N+1}N\Delta \end{cases} \tag{4.14}$$

where $N = 2^{q-1} - 1$, $-N+1 \leqslant l \leqslant N-1$, $1 \leqslant r \leqslant N$, and d is a quantization parameter within the range $(1, d_v - 1]$. Generally, the values represented by the $(q+1)$-bit quasi-uniform quantization messages $(l, 0)$ are $l\Delta$, and the values of messages $(\pm r, 1)$ are $\pm d^r N\Delta$, respectively. For messages within the range of $[-N\Delta, N\Delta]$, the new quasi-uniform quantizer provides the same precision as a q-bit uniform quantizer with quantization step Δ. For messages outside that range, non-uniform quantization with exponentially increasing step sizes of the form $d^r N\Delta$ is used to allow reliable messages to be more accurately represented. Hence, the extra bit in $(q+1)$-bit quantization can be viewed as an *indicator bit*, which indicates whether the quantized message is in the uniform quantization range or in the non-uniform quantization range. Note that, in the non-uniform quantization range, the value with smallest magnitude is used to represent quantized values in each interval.

Table 4.1 shows an example of (3+1)-bit quasi-uniform quantization with $\Delta = 1$ and $d = 3$. The first bit is the sign bit, and the last bit indicates whether the uniform or exponential step size is used. The uniform quantization range in this example is from -3 to 3 with uniform step size 1, and the exponential quantization range is above 3 or below -3 with nonuniform step size $3 \cdot (3^r - 3^{r-1})$ for $1 \leqslant r \leqslant 4$. For example, all values within the non-uniform quantization interval $[27, 81)$ would be quantized to 27. The decimal values are used in the VN and CN update computations, and then the corresponding quantized binary messages are passed between VNs and CNs.

In comparison to the modified and optimized quantization methods proposed in [12–14], the $(q+1)$-bit quasi-uniform quantizer can represent values of much greater magnitudes. Since the range of uniformly quantized messages in MP decoders is small in practice, the correct messages outside a trapping set could reach the saturation level within a few iterations. As a result, even though correct, these messages may not be large enough to offset the contribution of incorrect incoming messages for problematic VNs. Hence, even after optimization of the step

Table 4.1: (3+1)-bit quasi-uniform quantization with $\Delta = 1$ and $d = 3$.

message	**range**	**value**	**message**	**range**	**value**
0000	(-0.5,0.5]	0	1110	(-0.5,0.5]	0
0010	(0.5,1.5]	1	1100	(-1.5,-0.5]	-1
0100	(1.5,2.5]	2	1010	(-2.5,-1.5]	-2
0110	(2.5,9)	3	1000	(-9,-2.5]	-3
0001	[9,27)	9	1111	(-27,-9]	-9
0011	[27,81)	27	1101	(-81,-27]	-27
0101	[81,243)	81	1011	(-243,-81]	-81
0111	[243,∞)	243	1001	($-\infty$,-243]	-243

and range of a uniform quantizer, the decoder may not produce the same error floor performance as a floating-point MP decoder. In contrast, the saturation levels of the proposed (q+1)-bit quasi-uniform quantizer are greatly extended, allowing the correct messages outside a trapping set to grow large enough to overcome all incorrect messages reaching the problematic VNs from other VNs within the trapping set. It should be noted, however, that the uniform quantization step included in our quasi-uniform quantization affects the error-rate performance in the waterfall region, where the magnitudes of messages are generally small and the precision of the messages is important.

We can further extend the idea of (q+1)-bit quasi-uniform quantization to a more general form. The (q+1)-bit quasi-uniform quantization has $n = q+1$ bits in total to represent $N = 2^{n-1}$ different levels of magnitudes, and as described in (4.14), $N/2$ levels are allocated to the uniform quantization range and the other $N/2$ levels have exponential step sizes. The general symmetric n-bit quasi-uniform quantization can also represent $N = 2^{n-1}$ different magnitudes, but it can have any number, say N_u, of levels in the uniform quantization range and the remaining $N - N_u$ levels in the exponential quantization range. With a quantization rule similar to (4.14), the quantized values of the general n-bit quasi-uniform quantization are $l\Delta$ for $-N_u < l < N_u$, $d^{l-N_u+1}N_u\Delta$ for $l \geqslant N_u$, and $-d^{l-N_u+1}N_u\Delta$ for $l \leqslant -N_u$. Unlike the (q+1)-bit quasi-uniform quantization, the general n-bit quasi-uniform quantization does not have a specific indicator bit, and therefore, it is more flexible in the sense that it can allocate more levels to the uniform range or to the exponential rage to have best performance. Table 4.2 shows an example of the general 4-bit quasi-uniform quantization with $N_u = 5$, $\Delta = 1$, and $d = 3$. The uniform quantization range in this example is from -4 to 4 with uniform step size 1, and the exponential range is

Table 4.2: 4-bit quasi-uniform quantization with $\Delta = 1$ and $d = 3$.

message	range	value	message	range	value
0000	(-0.5,0.5]	0	1111	(-0.5,0.5]	0
0001	(0.5,1.5]	1	1110	(-1.5,-0.5]	-1
0010	(1.5,2.5]	2	1101	(-2.5,-1.5]	-2
0011	(2.5,3.5]	3	1100	(-3.5,-2.5]	-3
0100	(3.5,12)	4	1011	(-12,-3.5]	-4
0101	[12,36)	12	1010	(-36,-12]	-12
0110	[36,108)	36	1001	(-108,-36]	-36
0111	[108,∞)	108	1000	($-\infty$,-108]	-108

above 4 or below -4 with exponential step size $4 \cdot (3^r - 3^{r-1})$ for $1 \leqslant r \leqslant 3$.

Although the motivation for the proposed quasi-uniform quantization method came from an analysis of message-passing decoder behavior on trapping sets that satisfy the k-separation assumption for large k, a property not satisfied by trapping sets in practical LDPC codes, the simulation results in the next section demonstrate a significant reduction in the error floors of three representative LDPC codes, with no increase in the decoding complexity.

4.4 Numerical Results

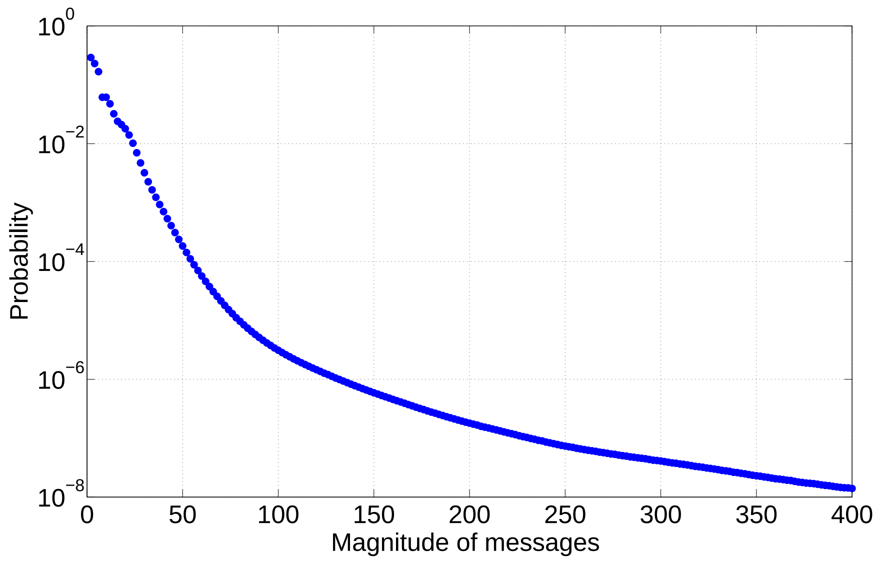
To demonstrate the improved performance offered by our proposed quasi-uniform quantization method, we compare its error-rate performance to that of uniform quantization with several types of MP decoders applied to three known LDPC codes on the BSC and the AWGNC. The three LDPC codes we evaluated are a rate-0.3 (640,192) quasi-cyclic (QC) LDPC code [10], the rate-0.5 (2640,1320) Margulis LDPC code [18], and MacKay's (4095,3358) regular LDPC code of rate 0.82 (the 4095.737.3.101 code in [32]). The frame error rate (FER) curves are based on Monte Carlo simulations that generated at least 200 error frames for each point in the plots, and the maximum number of decoding iterations was set to 200 unless otherwise indicated.

The (640,192) QC-LDPC code, designed by Han and Ryan [10], is a variable-regular code with variable degree 5 and check degrees ranging from 5 to 9. It has 64 isomorphic (5,5) trapping sets and 64 isomorphic (5,7) trapping sets. We applied our exhaustive trapping set search algorithm [31] to this code, and these are the only two types of (a, b) trapping set for $a \leqslant 15$ and $b \leqslant 7$. The error floor starts relatively high for MP decoders with limited message range, so it is quite easy to reach the error floor with Monte Carlo simulation.

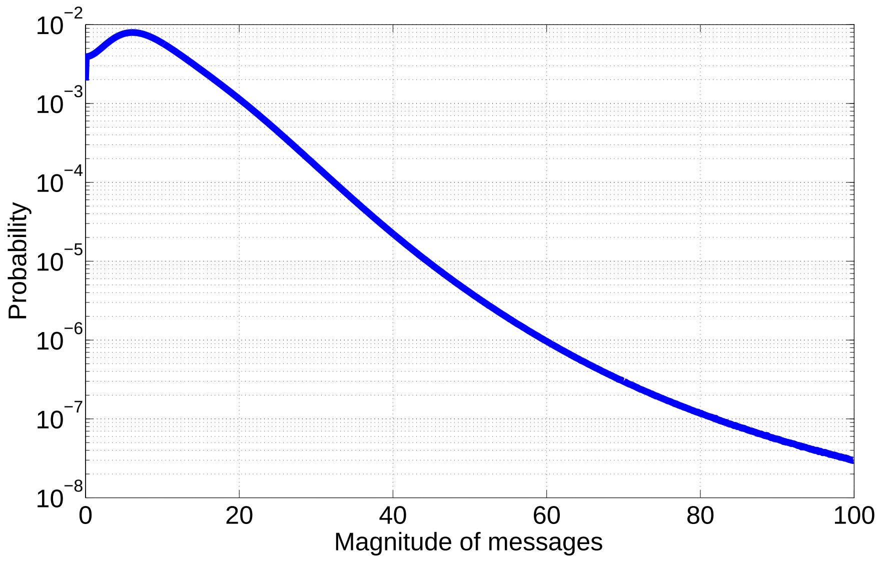
Fig. 4.5 shows the probability density function (pdf) of the magnitude of messages passed within the iterative message-passing decoders. Fig. 4.5(a) shows the pdf for the min-sum decoder applied to the (640,192) QC-LDPC code on the BSC with $p = 0.03$, where the magnitude of all input LLRs is scaled to 1. Fig. 4.5(b) shows the pdf of the SPA decoder applied to the Margulis code of length 2640 on the AWGNC with $E_b/N_0 = 2.25$ dB. The data in each figure were obtained by using the corresponding floating-point MP decoders on sequences of received symbols from more than 10 million channel realizations, and gathering all the messages passed on all edges during all decoding iterations to generate the pdf. In the simulation, the iterative MP decoder stops when a codeword is found or when it reaches the maximum number of iterations, namely 200. We can see from the figures that there is a considerable number of messages with large magnitude, which can be either favorable or unfavorable, and by further examining the simulation data, we found that such strong messages, in general, help to successfully decode the received symbols, as suggested by the idealized theoretical analysis described in Section 4.2.

Figs. 4.6–4.8 show simulation results for various types of quantized min-sum decoders and floating-point MS decoders. For the BSC, we scaled the magnitudes of decoder input messages from the channel to 1, since, for linear decoders such as Gallager-B and min-sum, the scaling of channel input messages does not affect the decoding performance. For attenuated and offset min-sum decoding, we can compensate for the scaling by adjusting the attenuation and the offset factor, respectively. The uniform quantization step size Δ is set to 1 or 0.5. So, for example, when $\Delta = 1$, the 3-bit uniform quantizer produces values $\{\pm 3, \pm 2, \pm 1, 0\}$, and the (3+1)-bit quasi-uniform quantizer with $d = 3$ yields the values described in Table 4.1. In the simulation, the parameter d was heuristically chosen by testing different values, and when q is large, a small d would be enough to represent a large range of magnitudes.

In Fig. 4.6, we see that the slopes of the error floors resulting from uniform quantization are close to that of the Gallager-B decoder error floor. This is because, when most messages saturate at the same magnitude, min-sum decoding essentially degenerates to Gallager-B decoding, relying solely upon the signs of messages. Comparing uniform quantizers with the same number of bits but different step sizes, we notice that smaller step size produces better waterfall performance but higher error floor. This observation can be explained by the saturation level of these quantizers. For example, 3-bit and 4-bit uniform quantization with step size $\Delta = 1$ saturates at 3 and 7, and with $\Delta = 0.5$ saturates at 1.5 and 3.5, respectively. The stronger messages, i.e.,



(a) Min-sum decoder on the (640,192) QC-LDPC code over BSC of $p = 0.03$ and $|LLR| = 1$.



(b) SPA decoder on the Margulis code of length 2640 over AWGNC of $E_b/N_0 = 2.25$ dB.

Figure 4.5: Probability density function of magnitude of messages.

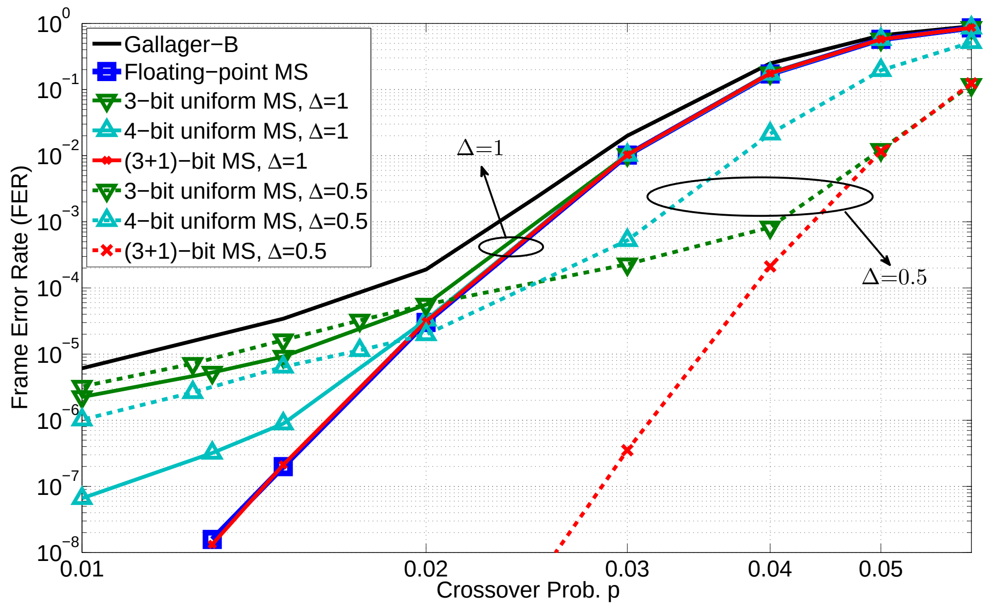


Figure 4.6: FER results of min-sum (MS) decoder on the (640,192) QC-LDPC code on BSC. Uniform quantization step $\Delta = 1$ or 0.5, and $d = 3$ in (q+1)-bit quasi-uniform quantization.

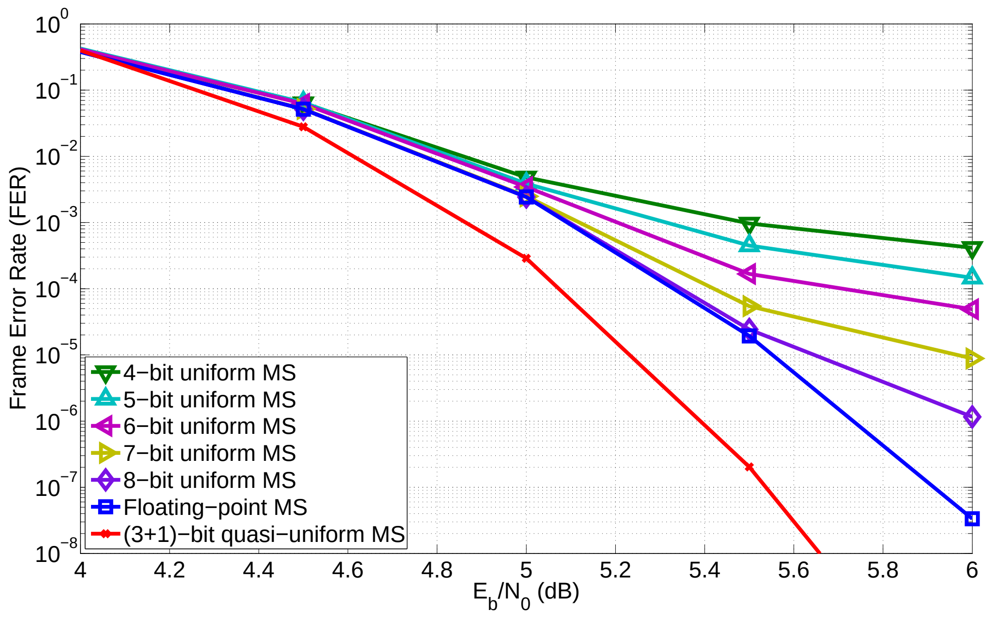


Figure 4.7: FER results of min-sum (MS) decoder on the (640,192) QC-LDPC code on AWGNC. The offset factor $\beta = 0.5$, uniform quantization step $\Delta = 0.5$, and $d = 3$ in (q+1)-bit quasi-uniform quantization.

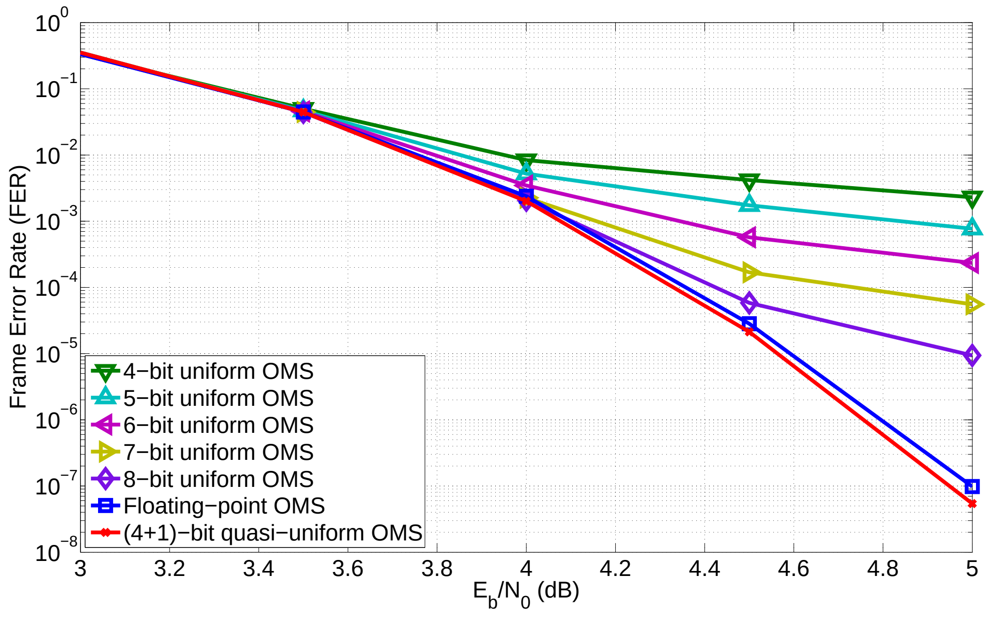


Figure 4.8: FER results of offset min-sum (OMS) decoder on the (640,192) QC-LDPC code on AWGNC. The offset factor $\beta = 0.5$, uniform quantization step $\Delta = 0.5$, and $d = 2$ in (q+1)-bit quasi-uniform quantization.

messages with larger magnitudes, can be helpful or harmful to the decoding process, depending on whether they are correct or not. The correct strong messages can help overcome the incorrectly received bits, but the strong incorrect messages would negatively influence the correctly received bits. In the error-floor region, when the channel condition is good, very few bits are received incorrectly, and as we showed in Theorems 4.4 and 4.8, large saturation levels allow correct messages to build up and overcome the incorrect messages in trapping sets. Therefore, in Fig. 4.6, the error floors produced by the different uniform quantizers are strictly in the order of their corresponding saturation levels.

However, in the waterfall region where many bits are received incorrectly, reducing the saturation level limits the propagation of strong incorrect messages. Moreover, in this specific case, another reason for the better performance of the quantization with step size $\Delta = 0.5$ is that, since the magnitudes of input LLRs to the min-sum decoder from the BSC are scaled to 1, the appearance of nonintegral saturated messages, together with a small saturation level, reduces the possibility of the summation of messages at a VN being zero, which could cause oscillatory behavior in the decoder. We can see from Fig. 4.6 that MS decoding with (3+1)-bit quasi-uniform quantization and $\Delta = 0.5$ performs even better than the floating-point MS decoder. This

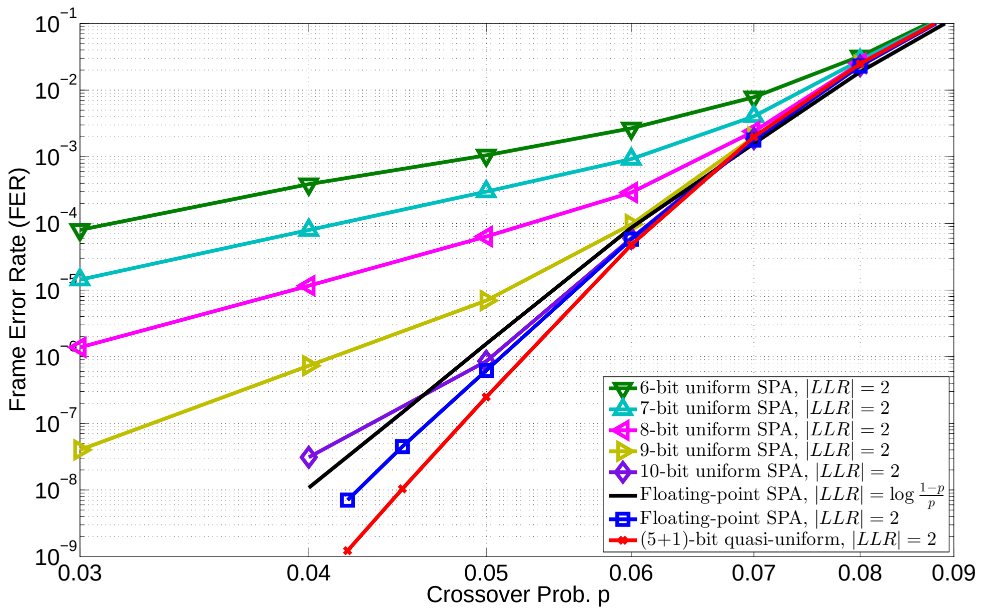


Figure 4.9: FER results of SPA decoder on the (640,192) QC-LDPC code on BSC. Uniform quantization step $\Delta = 0.25$, and $d = 1.3$ in $(q$+1)-bit quasi-uniform quantization.

is because its inherent uniform quantization with small saturation level improves the waterfall performance, as just explained, and the incorrect messages are not able to grow large enough to reach the next non-uniform quantization level. In other words, the proposed quasi-uniform quantization with carefully chosen parameters allows the correct messages to grow exponentially and limits the growth of incorrect messages. Similar results can also be found in Fig. 4.7, where (3+1)-bit quasi-uniform quantization also outperforms the floating-point MS decoder. However, we want to point out that such gains are highly code-dependent, and we conjecture that codes of higher variable degree would benefit more from the quasi-uniform quantization.

In Figs. 4.9 – Fig. 4.12, we show the simulation results of the quasi-uniform quantization method applied to the SPA decoder. In the simulation of quantized SPA decoding, the input LLRs and the messages passed between CNs and VNs are quantized values, but all the CN updates are done with floating-point computation of the box-plus update rule in (4.9). Therefore, the simulation results for SPA decoding with quantized messages shown here can be considered the best error-rate performance possible with any fixed-point implementation of quantized SPA decoding.

Figs. 4.9 and 4.10 show performance results for the (640,192) QC-LDPC code on the BSC and AWGNC, where the non-uniform quantization parameter of the quasi-uniform quan-

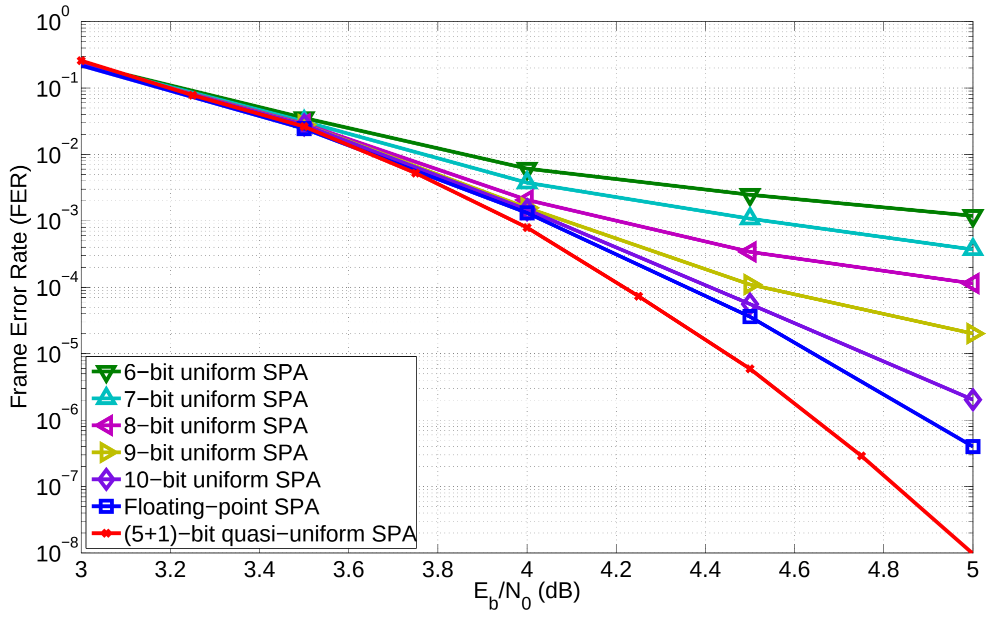


Figure 4.10: FER results of SPA decoder on the (640,192) QC-LDPC code on AWGNC. The uniform quantization step $\Delta = 0.25$, and $d = 1.5$ in (q+1)-bit quasi-uniform quantization.

tizer was set to $d = 1.5$ because, with $q \geqslant 5$, a large range of message values is covered, even with such a small d. We see that, on both channels, the SPA decoder with quasi-uniform quantization yields FER results very close to those obtained with the float-point SPA implementation. Decoding with uniform quantization, on the other hand, suffers from high error floors.

In Fig. 4.9, the magnitude of input LLRs from the BSC to quantized SPA decoders was scaled to 2; as we pointed out in earlier sections, such scaling could have an impact on the error-rate performance. The figure compares two floating-point SPA decoders that use different scalings of the input LLR magnitude. One uses the exact LLR value whose magnitude is $|\log \frac{1-p}{p}|$, where p is the channel error probability; the other scales the magnitude of all input LLRs to 2. We can see from the figure that the floating-point SPA decoder with scaled input LLRs from the BSC has slightly better performance, especially when the channel quality is good, i.e., where p is small. We note that the choice of the input LLR magnitude is heuristic and depends on the underlying LDPC code. Scaling the magnitude to 2 in this case may not be optimal, but we indeed found that scaling the magnitude to 1 would greatly degrade the error-rate performance.

In Fig. 4.11, we show results for SPA decoding of the length-2640 Margulis code. For this example, we adopted the following two-piece linear approximation from [29] in the compu-

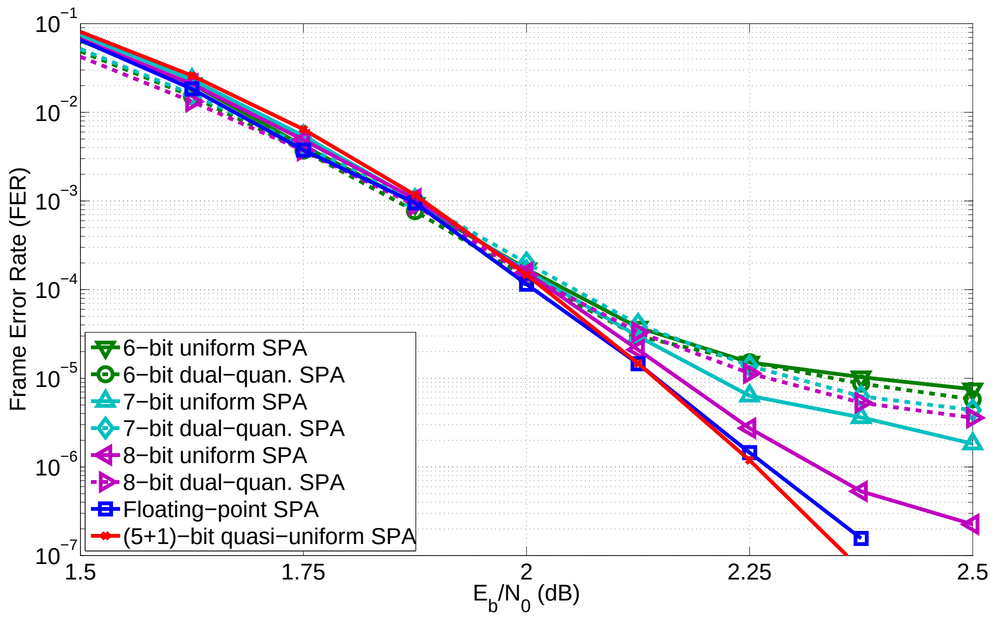


Figure 4.11: FER results of approximate-SPA decoder on the Margulis code of length 2640 on AWGNC. Uniform quantization step $\Delta = 0.25$, and $d = 1.3$ in (q+1)-bit quasi-uniform quantization.

tation of the $s(x, y)$ function in (4.12) for box-plus SPA decoding,

$$\ln\left(1 + e^{-|x|}\right) = \begin{cases} 0.6 - 0.24|x|, & \text{if } |x| < 2.5 \\ 0, & \text{otherwise.} \end{cases} \tag{4.15}$$

The approximated box-plus SPA decoder ran about five times faster than the floating-point SPA decoder, with a performance penalty of less than 0.02 dB in the waterfall region. In Fig. 4.11, we also include the dual quantization SPA decoding proposed by Zhang *et al.* [14], where the $\phi(x)$ function is quantized into a mapping table, denoted as $\bar{\phi}(x)$. Following the notation in [14], we use dual quantization with parameters Q4.2/1.5, Q5.2/1.6, and Q6.2/1.7 for 6-bit, 7-bit, and 8-bit quantizers, respectively. The Q$m.f$ quantizer uses uniform quantization to represent a signed fixed-point number with m bits to the left of the radix point for the integer part and f bits to the right of the radix point for the fractional part. For example, a Q4.2 quantizer has uniform quantization step size of 0.25 and a range $[-7.75, 7.75]$. Hence, all the quantization methods compared here have the same uniform step size of 0.25 when quantizing the channel input LLRs.

From the plot of the $\phi(x)$ function in Fig. 4.3, we can see that the saturation level $\bar{\phi}(0)$ is limited by the quantization step size, because it is desirable to have $\bar{\phi}(0) < x$ for all x satisfying

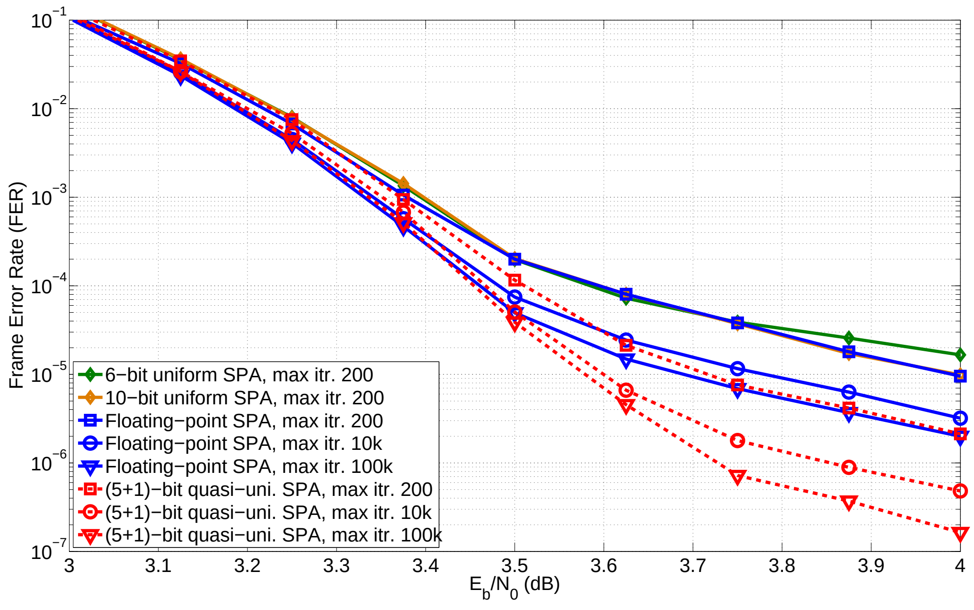


Figure 4.12: FER results of approximate-SPA decoder on (4095,3358) LDPC code on AWGNC, where $\Delta = 0.5$ and $d = 1.3$.

$\bar{\phi}(x) = 0$. In other words, in the dual quantization scheme, the saturation level has to match the resolution of the quantizer; otherwise the error-rate performance in both the waterfall region and the error-floor region will be significantly degraded. Based on error-rate simulations using a range of saturation levels for dual quantization methods, we chose the saturation level for $\bar{\phi}(x) = 0$ to be 5.5, 7, and 8 for the 6-bit, 7-bit, and 8-bit quantizers, respectively. Moreover, due to the use of a mapping table with limited precision for the quantized $\phi(x)$ function, the error-rate performance in the error-floor region of SPA decoding with any dual quantization scheme is strictly worse than that of box-plus SPA decoding with uniform quantization, assuming the same number of bits and the same quantization step size are used in the representation of the channel LLR. The gap is greater when decoders are allowed to use more quantization bits, as clearly shown in the simulation results.

In Figs. 4.10 and 4.11, the proposed (5+1)-bit quasi-uniform quantization again has the best error-floor performance among all quantized decoders, and ever better than the double-precision floating-point box-plus SPA decoder. We observed from the simulation data that the floating-point SPA generally requires more iterations to decode a codeword than the quasi-uniform quantized SPA, especially in the high SNR region. Since the maximum number of iterations is set to 200, the quasi-uniform quantized SPA is able to outperform its floating-point counterpart. The fast convergence of the quasi-uniform quantized SPA derives from its non-uniform step size. From the idealized theoretical analysis, we know that the exponential growth rate of correct messages is larger than that of incorrect messages. In quasi-uniform quantization with proper parameters, the correct messages can reach the higher magnitude level earlier than the incorrect messages, and the incorrect messages are more likely to be quantized to lower magnitude levels. Hence, the correct messages overcome incorrect messages faster, and the decoder can converge to a codeword after fewer iterations.

Fig. 4.12 compares the error-rate performance of the floating-point SPA and the (5+1)-bit quasi-uniform SPA for MacKay's (4095,3358) LDPC code when the maximum number of iterations is set to different values. This code has a high error floor when decoded with floating-point SPA for maximum 200 iterations. We can see that the error floor is reduced when the number of iterations increases. This again confirms our idealized theoretical analysis that the correct messages outside the trapping sets grow faster than incorrect messages within the trapping set. The proposed quasi-uniform quantization provides superior error floor performance in this case, and with only 200 iterations, it gives the same error floor as the floating-point one with 100,000 iterations. All the examples in this section indicate that the floating-point implementation of MP decoding is not necessarily better than a quantized fixed-point implementation.

In all of the decoding failures observed when using the quasi-uniform quantizer with MS decoding and SPA decoding, no error pattern corresponded to the support of a small trapping set. With uniform quantization, on the other hand, almost all of the decoding failures corresponded to small trapping sets when the channel error probability of the BSC was small or the SNR of the AWGNC was high. We also compared decoder performance on sequences in which every VN in a single trapping set of type (5,5) or (5,7) of the (640,192) code was incorrect, with all other VNs set to correct values. In all cases, the large-range message-passing decoder and the message-passing decoder with the proposed quasi-uniform quantization method decoded successfully, while decoders with the uniform quantizer failed. The same results were also obtained for the (12,4) and (14,4) trapping sets in the Margulis code.

We emphasize The analytical and numerical results in this chapter are only for variable-regular LDPC codes. Extension of this analysis to variable-irregular LDPC codes does not appear to be straightforward. Moreover, partly due to the low error floor of well-designed irregular LDPC codes, our simulation results using quasi-uniform quantizer do not show a significant improvement in error-rate performance over the uniform quantizer with the same number of bits.

4.5 Conclusion

Trapping sets and other error-prone substructures are known to influence the error-rate performance of LDPC codes with iterative message-passing decoding. In this chapter, we have shown that the use of uniform quantization in iterative MP decoding can be a significant factor contributing to the error floor phenomenon in LDPC code performance. An analysis of iterative MP decoding in an idealized setting suggests that decoder message saturation plays a key role in the occurrence of errors in small trapping sets, leading to observed error floor behaviors. To address this problem, we proposed a novel quasi-uniform quantization method that effectively extends the dynamic range of the quantizer. Without modifying the CN and VN update rules or adding extra stages to standard iterative decoding algorithms, the use of this quantizer was shown to significantly lower the error floors of several well-studied LDPC codes when used with various iterative MP decoding algorithms on the BSC and AWGNC. Simulation results confirmed that this new quantization method can significantly reduce the error floors of these codes with essentially no increase in decoding complexity.

Acknowledgment

This chapter is in part a reprint of the material in the papers: Xiaojie Zhang and Paul H. Siegel, "Quantized Min-Sum Decoders with Low Error Floor for LDPC codes," in *Proc. IEEE International Symposium on Information Theory*, Cambridge, MA, July 2–5, 2012, pp. 2871–2875., and Xiaojie Zhang and Paul H. Siegel, "Will the real error floor please stand up?" in *Proc. IEEE International Conference Signal Processing and Communication*, Bangalore, India, July 22–25, 2012, pp. 1–5. The dissertation author was the primary investigator and author of these paper.

Bibliography

[1] D. Divsalar and C. Jones, "Protograph based low error floor LDPC coded modulation," in *Proc. IEEE Military Commun. Conf.*, vol. 1, Oct. 2005, pp. 378–385.

[2] J. Lu and J. M. F. Moura, "Structured LDPC codes for high-density recording: large girth and low error floor," *IEEE Trans. Magnetics*, vol. 42, pp. 208–213, Feb. 2006.

[3] S. K. Chilappagari, S. Sankaranarayanan, and B. Vasic, "Error floors of LDPC codes on the binary symmetric channel," in *Proc. IEEE Int. Conf. on Commun.*, Istanbul, Turkey, Jun. 2006, pp. 1089–1094.

[4] S. Laendner and O. Milenkovic, "Algorithmic and combinatorial analysis of trapping sets in structured LDPC codes," in *Proc. 2005 Intl. Conf. Wireless Networks, Commun., Mobile Comp.*, Jun. 2005, pp. 630–635.

[5] V. Savin, "Iterative LDPC decoding using neighborhood reliabilities," in *Proc. IEEE IEEE Int. Symp. Inform. Theory (ISIT)*, Nice, France, Jun. 2007, pp. 221–225.

[6] A. Casado, M. Griot, and R. Wesel, "Informed dynamic scheduling for belief-propagation decoding of LDPC codes," in *Proc. IEEE Int. Conf. on Commun.*, Glasgow, UK, Jun. 2007, pp. 932–937.

[7] E. Cavus and B. Daneshrad, "A performance improvement and error floor avoidance technique for belief propagation decoding of LDPC codes," in *Proc. IEEE Intl. Symp. on Pers., Indoor and Mobile Radio Comm.*, Berlin, Germany, Sept. 2005, pp. 2386–2390.

[8] G. Kyung and C. Wang, "Exhaustive search for small fully absorbing sets and the corresponding low error-floor decoder," in *Proc. IEEE Int. Symp. Inform. Theory (ISIT)*, Austin, TX, Jul. 2010, pp. 739–743.

[9] N. Varnica, M. P. C. Fossorier, and A. Kavcic, "Augmented belief propagation decoding of low-density parity-check codes," *IEEE Trans. Commun.*, vol. 55, no. 7, pp. 1308–1317, Jul. 2007.

[10] Y. Han and W. E. Ryan, "Low-floor decoders for LDPC codes," *IEEE Trans. Commun.*, vol. 57, no. 6, pp. 1663–1673, Jun. 2009.

[11] Z. Zhang, L. Dolecek, B. Nikolic, V. Anantharam, and M. Wainwright, "Lowering LDPC error floors by postprocessing," in *Proc. IEEE Glob. Telecom. Conf.*, Cannes, France, Dec. 2008, pp. 1–6.

[12] J. Zhao, F. Zarkeshvari, and A. Banihashemi, "On implementation of min-sum algorithm and its modifications for decoding LDPC codes," *IEEE Trans. Commun.*, vol. 53, no. 4, pp. 549–554, Apr. 2005.

[13] T. Zhang, Z. Wang, and K. Parhi, "On finite precision implementation of LDPC codes decoder," in *Proc. IEEE ISCAS*, May 2001, pp. 201–205.

[14] Z. Zhang, L. Dolecek, B. Nikolić, V. Anatharam, and M. Wainwright, "Design of LDPC decoders for improved low error rate performance: quantization and algorithm choices," *IEEE Treans. Wireless Commun.*, vol. 8, no. 11, pp. 3258–3268, Nov. 2009.

[15] Z. Zhang, "Design of LDPC decoders for improved low error rate performance," Ph.D. dissertation, Univ. of California at Berkeley, 2009.

[16] B. Butler and P. Siegel, "Error floor approximation for LDPC codes in the AWGN channel," in *Proc. Annual Allerton Conference on Communication, Control, and Computing*, Monticello, IL, Sep. 2011, pp. 204–211.

[17] C. Di, D. Proietti, E. Telatar, T. Richardson, and R. Urbanke, "Finite length analysis of low-density parity-check codes on the binary erasure channel," *IEEE Trans. Inf. Theory*, vol. 48, no. 6, pp. 1570–1579, Jun. 2002.

[18] D. MacKay and M. Postol, “Weakness of Margulis and Ramanujan-Margulis low-density parity check codes,” *Electron. Notes Theor. Comp. Sci.*, vol. 74, 2003.

[19] T. Richardson, “Error-floors of LDPC codes,” in *Proc. of the 41st Annual Allerton Conference on Communication, Control, and Computing, Monticello*, IL, Oct. 1–3, 2003, pp. 1426–1435.

[20] L. Dolecek, Z. Zhang, V. Anantharam, M. Wainwright, and B. Nikolic, “Analysis of absorbing sets and fully absorbing sets of array-based LDPC codes,” *IEEE Trans. Inform. Theory*, vol. 56, no. 1, pp. 181–201, Jan. 2010.

[21] P. O. Vontobel and R. Koetter, “Graph-cover decoding and finite-length analysis of message-passing iterative decoding of LDPC codes,” CoRR, arxiv.org/abs/cs.IT/0512078.

[22] L. Dolecek, Z. Zhang, M. Wainwright, and V. Anatharam. “Evaluation of the low frame error rate performance of LDPC codes using importance sampling,” in *Proc. IEEE Inform. Theory Workshop (ITW)*, Lake Tahoe, CA, Sep. 2007, pp. 202–207.

[23] B. Frey, R. Koetter, and A. Vardy, “Signal-space characterization of iterative decoding,” *IEEE Trans. Inform. Theory*, vol. 47, no. 2, pp. 766-781, Feb. 2001.

[24] S. K. Planjery, D. Declercq, S. K. Chilappagari, and B. Vasic, “Multilevel decoders surpassing belief propagation on the binary symmetric channel,” in *Proc. IEEE Int. Symp. Inform. Theory (ISIT)*, Austin, TX, Jul. 2010, pp. 769–773.

[25] J. Chen, A. Dholakia, E. Eleftheriou, M. Fossorier, and X. Hu, “Reduced-complexity decoding of LDPC codes,” *IEEE Trans. Communications*, vol. 53, no. 8, pp. 1288–1299, August 2005.

[26] *IEEE Standard for Floating-Point Arithmetic*, IEEE standard 754-2008, Aug 29, 2008.

[27] B. Butler and P. Siegel, ”Numerical problems of belief propagation decoders and solutions,” accepted to Globecomm 2012.

[28] X. Hu, E. Eleftheriou, D. Arnold, and A. Dholakia, ”Efficient implementations of the sum-product algorithm for decoding LDPC codes,” in *Proc. IEEE Global Telecommun. Conf.*, vol. 2, San Antonio, Nov. 2001, pp. 1036–1036E.

[29] G. Richter, G. Schmidt, M. Bossert, and E. Costa, “Optimization of a reduced-complexity decoding algorithm for LDPC codes by density evolution,” in *Proc. IEEE Int. Conf. on Commun.*, vol. 1, Seoul, May 2005, pp. 642–646.

[30] J. Chen and M. Fossorier, “Near optimum unversal belief propagation based decoding of low-density parity check codes,” *IEEE Trans. Communications*, vol. 50, no. 3, pp. 406–414, Mar. 2002.

[31] X. Zhang and P. H. Siegel, “Efficient algorithms to find all small error-prone substructures in LDPC codes,” in *Proc. IEEE IEEE Glob. Telecom. Conf.*, Huston, TX, Dec. 2011, pp. 1–6.

[32] D. J. C. MacKay, *Encyclopedia of Sparse Graph Codes*. [Online]. Available: http://www.inference.phy.cam.ac.uk/mackay/codes/data.html

Chapter 5

Adaptive Cut Generation Algorithm for Improved Linear Programming Decoding of Binary Linear Codes

In the original formulation of LP decoding proposed by Feldman *et al.* [1], the number of constraints in the LP problem is linear in the block-length but exponential in the maximum check node degree, and the authors also argued that the number of useful constraints could be reduced to polynomial in code length. The computational complexity of the original LP formulation therefore can be prohibitively high, motivating the design of computationally simplified decoding algorithms that can achieve the same error-rate performance with a smaller number of constraints. For example, efficient polynomial-time algorithms can be used for solving the original LP formulation [2]. An alternative LP formulation whose size is linear in the check node degree and code length can also be obtained by changing the graphical representation of the code [3,4]; namely, all check nodes of high degree are replaced by dendro-subgraphs (trees) with an appropriate number of auxiliary degree-3 check nodes and degree-2 variable nodes. Several other low-complexity LP decoders were also introduced in [5], suggesting that LP solvers with complexity similar to the min-sum algorithm and the sum-product algorithm are feasible.

Another approach is to add linear constraints in an adaptive and selective way during the LP formulation [6]. Such an adaptive linear programming (ALP) decoding approach also allows the adaptive incorporation of linear constraints generated by redundant parity checks (RPC) into the LP problem, making it possible to reduce the feasible space and improve the system performance. A linear inequality derived from an RPC that eliminates a pseudocodeword solution is referred to as a "cut."

An algorithm proposed in [6] uses a random walk on a subset of the code factor graph to find these RPC cuts. However, the random nature of this algorithm limits its efficiency. In fact, experiments show that the average number of random trials required to find an RPC cut grows exponentially with the length of the code.

Recently, the authors of [7] proposed a separation algorithm that derives Gomory cuts from the IP formulation of the decoding problem and finds cuts from RPCs which are generated by applying Gaussian elimination to the original parity-check matrix. In [8], a cutting-plane method was proposed to improve the fractional distance of a given binary parity-check matrix – the minimum weight of nonzero vertices of the fundamental polytope – by adding redundant rows obtained by converting the parity-check matrix into row echelon form after a certain column permutation. However, we have observed that the RPCs obtained by the approach in [8] are not able to produce enough cuts to improve the error-rate performance relative to the separation algorithm when they are used in conjunction with either ALP decoding or the separation algorithm. A detailed survey on mathematical programming approaches for decoding binary linear codes can be found in [9].

In this chapter, we greatly improve the error-correcting performance of LP decoding by designing algorithms that can efficiently generate cut-inducing RPCs and find possible cuts from such RPCs. First, we derive a new necessary condition and a new sufficient condition for a parity-check to provide a cut at a given pseudocodeword. These conditions naturally suggest an efficient algorithm that can be used to find, for a given pseudocodeword solution to an LP problem, the unique cut (if it exists) among the parity inequalities associated with a parity check. This algorithm was previously introduced by Taghavi *et al.* [10, Algorithm 2] and, independently and in a slightly different form, by Wadayama [11, Fig. 6].

The conditions also serve as the motivation for a new, more efficient adaptive cut-inducing RPC generation algorithm that identifies useful RPCs by performing specific elementary row operations on the original parity-check matrix of the binary linear code. By adding the corresponding linear constraints into the LP problem, we can significantly improve the error-rate performance of the LP decoder, even approaching the ML decoder performance in the high-SNR region for some codes. Finally, we modify the ALP decoder to make it more efficient when being combined with the new cut-generating algorithm. Simulation results demonstrate that the proposed decoding algorithms significantly improve the error-rate performance of the original LP decoder.

The remainder of the chapter is organized as follows. In Section 5.1, we review the original formulation of LP decoding and several adaptive LP decoding algorithms. Section 5.2 presents the new necessary condition and new sufficient condition for a parity-check to induce a cut, as well as their connection to the efficient cut-search algorithm. In Section 5.3, we describe our proposed algorithm for finding RPC-based cuts. Section 5.4 presents our simulation results, and Section 5.5 concludes the chapter.

5.1 Adaptive Linear Programming Decoding

In the original formulation of LP decoding presented in [1], every check node j generates $2^{|\mathcal{N}_j|-1}$ parity inequalities that are used as linear constraints in the LP problem described in (2.14). The total number of constraints and the complexity of the original LP decoding problem grows exponentially with the maximum check node degree. So, even for binary linear codes with moderate check degrees, the number of constraints in the original LP decoding could be prohibitively large. In the literature, several approaches to reducing the complexity of the original LP formulation have been described [2–6]. We will use adaptive linear programming (ALP) decoding [6] as the foundation of the improved LP decoding algorithms presented in later sections. The ALP decoder exploits the structure of the LP decoding problem, reflected in the statement of the following lemma.

Lemma 5.1 (Theorem 1 in [6]) *If at any given point $\mathbf{u} \in [0,1]^n$, one of the parity inequalities introduced by a check node j is violated, the rest of the parity inequalities from this check node are satisfied with strict inequality.*

Definition 5.2 *Given a parity-check node j, a set $\mathcal{V} \subseteq \mathcal{N}_j$ of odd cardinality, and a vector $\mathbf{u} \in [0,1]^n$ such that the corresponding parity inequality of the form* (2.12) *or* (2.13) *does not hold, we say that the constraint is* violated *or, more succinctly, a* cut *at $\mathbf{u}$.* [1]

In [6], an efficient algorithm for finding cuts at a vector $\mathbf{u} \in [0,1]^n$ was presented. It relies on the observation that violation of a parity inequality (2.13) at $\mathbf{u}$ implies that

$$|\mathcal{V}| - 1 < \sum_{i \in \mathcal{V}} u_i \leqslant |\mathcal{V}| \tag{5.1}$$

[1]In the terminology of [9], if (2.13) does not hold for a pseudocodeword $\mathbf{u}$, then the vector $(\mathbf{r}, t) \in \mathbb{R}^n \times \mathbb{R}$, where $r_i = 1$ for all $i \in \mathcal{V}$, $r_i = -1$ for all $i \in \mathcal{N}_j \setminus \mathcal{V}$, $r_i = 0$ otherwise, and $t = |\mathcal{V}| - 1$, is a *valid cut*, separating $\mathbf{u}$ from the codeword polytope.

and

$$0 \leqslant \sum_{i \in \mathcal{N}_j \setminus \mathcal{V}} u_i < u_v, \text{ for all } v \in \mathcal{V}. \tag{5.2}$$

where $\mathcal{V}$ is an odd-sized subset of $\mathcal{N}_j$.

Given a parity check j, the algorithm first puts its neighboring variables in $\mathbf{u}$ into non-increasing order, i.e., $u_{j_1} \geqslant \ldots \geqslant u_{j_n}$, for $u_{j_i} \in \mathcal{N}_j$. It then successively considers subsets of odd cardinality having the form $\mathcal{V} = \{u_{j_1}, \ldots, u_{j_{2k+1}}\} \subseteq \mathcal{N}_j$, increasing the size of $\mathcal{V}$ by two each step, until a cut (if one exists) is found. This algorithm can find a cut among the constraints corresponding to a check node j by examining at most $|\mathcal{N}_j|/2$ inequalities, rather than exhaustively checking all $2^{|\mathcal{N}_j|-1}$ inequalities in the original formulation of LP decoding.

The ALP decoding algorithm starts by solving the LP problem with the same objective function as the ML decoding, but with only the following constraints

$$\begin{cases} 0 \leqslant u_i & \text{if} \quad \gamma_i \geqslant 0 \\ u_i \leqslant 1 & \text{if} \quad \gamma_i < 0. \end{cases} \tag{5.3}$$

The solution of this initial LP problem can be obtained simply by making a hard decision on the components of a received vector. The ALP decoding algorithm starts with this point, searches every check node for cuts, adds all the cuts found during the search as constraints into the LP problem, and solves it again. This procedure is repeated until an optimal integer solution is generated or no more cuts can be found (see [6] for more details). Adaptive LP decoding has exactly the same error-correcting performance as the original LP decoding.

5.2 Cut Conditions

In this section, we derive a necessary condition and a sufficient condition for a parity inequality to be a cut at $\mathbf{u} \in [0,1]^n$. We also show their connection to the efficient cut-search algorithm proposed by Taghavi *et al.* [10, Algorithm 2] and Wadayama [11, Fig. 6]. This algorithm is more efficient than the search technique from [6] that was mentioned in Section 5.1.

Consider the original parity inequalities in (2.12) given by Feldman *et al.* in [1]. If a parity inequality derived from check node j induces a cut at $\mathbf{u}$, the cut can be written as

$$\sum_{i \in \mathcal{V}} (1 - u_i) + \sum_{i \in \mathcal{N}_j \setminus \mathcal{V}} u_i < 1, \tag{5.4}$$

for some $\mathcal{V} \subseteq \mathcal{N}_j$ with $|\mathcal{V}|$ odd.

From (5.4) and Lemma 5.1, we can derive the following necessary condition for a parity-check constraint to induce a cut.

Theorem 5.3 *Given a nonintegral vector* $\mathbf{u}$ *and a parity check* j*, let* $\mathcal{S} = \{i \in \mathcal{N}_j | 0 < u_i < 1\}$ *be the set of nonintegral neighbors of* j *in the Tanner graph, and let* $\mathcal{T} = \{i \in \mathcal{N}_j | u_i > \frac{1}{2}\}$*. A necessary condition for parity check* j *to induce a cut at* $\mathbf{u}$ *is*

$$\sum_{i \in \mathcal{T}} (1 - u_i) + \sum_{i \in \mathcal{N}_j \setminus \mathcal{T}} u_i < 1. \tag{5.5}$$

This is equivalent to

$$\sum_{i \in \mathcal{S}} \left| \frac{1}{2} - u_i \right| > \frac{1}{2} \cdot |\mathcal{S}| - 1 \tag{5.6}$$

where, for $x \in \mathbb{R}$, $|x|$ *denotes the absolute value.*

Proof: For a given vector $\mathbf{u}$ and a subset $\mathcal{X} \subseteq \mathcal{N}_j$, define the function

$$g(\mathcal{X}) = \sum_{i \in \mathcal{X}} (1 - u_i) + \sum_{i \in \mathcal{N}_j \setminus \mathcal{X}} u_i.$$

If parity-check j incudes a cut at $\mathbf{u}$, there must be a set $\mathcal{V} \subseteq \mathcal{N}_j$ of odd cardinality such that (5.4) holds. This means that $g(\mathcal{V}_{\text{cut}}) < 1$. Now, it is easy to see that the set $\mathcal{T}$ minimizes the function $g(\mathcal{X})$, from which it follows that $g(\mathcal{T}) \leqslant g(\mathcal{V}_{\text{cut}}) < 1$. Therefore, inequality (5.5) must hold in order for parity check j to induce a cut.

For $\frac{1}{2} \leqslant u_i \leqslant 1$, we have

$$\frac{1}{2} - \left| \frac{1}{2} - u_i \right| = \frac{1}{2} - \left(u_i - \frac{1}{2} \right) = 1 - u_i,$$

and for $0 \leqslant u_i \leqslant \frac{1}{2}$, we have

$$\frac{1}{2} - \left| \frac{1}{2} - u_i \right| = \frac{1}{2} - \left(\frac{1}{2} - u_i \right) = u_i.$$

Hence, (5.5) can be rewritten as

$$\sum_{i \in \mathcal{S}} \left(\frac{1}{2} - \left| \frac{1}{2} - u_i \right| \right) < 1$$

or equivalently,

$$\frac{1}{2} \cdot |\mathcal{S}| - \sum_{i \in \mathcal{S}} \left| \frac{1}{2} - u_i \right| < 1$$

which implies inequality (5.6). ■

Remark 5.1 Theorem 5.3 shows that to see whether a parity-check node could provide a cut at a pseudocodeword $\mathbf{u}$ we only need to examine its fractional neighbors.

Reasoning similar to that used in the proof of Theorem 5.3 yields a sufficient condition for a parity-check node to induce a cut at $\mathbf{u}$.

Theorem 5.4 *Given a nonintegral vector* $\mathbf{u}$ *and a parity check* j*, let* $\mathcal{S} = \{i \in \mathcal{N}_j | 0 < u_i < 1\}$ *and* $\mathcal{T} = \{i \in \mathcal{N}_j | u_i > \frac{1}{2}\}$*. If the inequality*

$$\sum_{i \in \mathcal{T}} (1 - u_i) + \sum_{i \in \mathcal{N}_j \setminus \mathcal{T}} u_i + 2 \cdot \min_{i \in \mathcal{S}} \left| \frac{1}{2} - u_i \right| < 1 \tag{5.7}$$

holds, there must be a violated parity inequality derived from parity check j*. This sufficient condition can be written as*

$$\sum_{i \in \mathcal{S}} \left| \frac{1}{2} - u_i \right| - 2 \cdot \min_{i \in \mathcal{S}} \left| \frac{1}{2} - u_i \right| > \frac{1}{2} \cdot |\mathcal{S}| - 1. \tag{5.8}$$

Proof: Lemma 5.1 implies that, if parity check j gives a cut at $\mathbf{u}$, then there is at most one odd-sized set $\mathcal{V} \subseteq \mathcal{N}_j$ that satisfies (5.4). From the proof of Theorem 5.3, we have $g(\mathcal{T}) \leqslant g(\mathcal{X})$ for all $\mathcal{X} \subseteq \mathcal{N}_j$. If $|\mathcal{T}|$ is even, we need to find one element $i^* \in \mathcal{N}_j$ such that inserting it into or removing it from $\mathcal{T}$ would result in the minimum increment to the value of $g(\mathcal{T})$. Obviously, $i^* = \arg\min_{i \in \mathcal{N}_j} \left| \frac{1}{2} - u_i \right|$, and the increment is $2 \cdot \left| \frac{1}{2} - u_{i^*} \right|$. If more than one i minimizes the expression $\left| \frac{1}{2} - u_i \right|$, we choose one arbitrarily as i^*. Hence, setting

$$\mathcal{V} = \begin{cases} \mathcal{T} \setminus \{i^*\}, & \text{if } i^* \in \mathcal{T} \\ \mathcal{T} \cup \{i^*\}, & \text{if } i^* \notin \mathcal{T} \end{cases}$$

we have $g(\mathcal{V}) = g(\mathcal{T}) + 2 \cdot \left| \frac{1}{2} - u_{i^*} \right| \geqslant g(\mathcal{T})$. If inequality (5.7) holds, then $g(\mathcal{T}) \leqslant g(\mathcal{V}) < 1$. Since either $|\mathcal{T}|$ or $|\mathcal{V}|$ is odd, (5.7) is a sufficient condition for parity-check constraint j to induce a cut at $\mathbf{u}$. Arguing as in the latter part of the proof of Theorem 5.3, it can be shown that (5.7) is equivalent to (5.8). ■

Theorem 5.3 and Theorem 5.4 provide a necessary condition and a sufficient condition, respectively, for a parity-check node to produce a cut at any given vector $\mathbf{u}$. It is worth pointing out that (5.5) becomes a necessary and sufficient condition for a parity check to produce a cut when $|\mathcal{T}|$ is odd, and (5.7) becomes a necessary and sufficient condition when $|\mathcal{T}|$ is even. Together, they suggest a highly efficient technique for finding cuts, the Cut-Search Algorithm (CSA) described in Algorithm 5.1 . If there is a violated parity inequality, the CSA returns the set $\mathcal{V}$ corresponding to the cut; otherwise, it returns an empty set.

As mentioned above, the CSA was used by Taghavi *et al.* [10, Algorithm 2] in conjunction with ALP decoding, and by Wadayama [11, Fig. 6] as a feasibility check in the context of

Algorithm 5.1 Cut-Search Algorithm (CSA)

```
Input: parity-check node $j$ and vector $\mathbf{u}$
Output: variable node set $\mathcal{V}$
 1: $\mathcal{V} \leftarrow \mathcal{T} = \{i \in \mathcal{N}_j | u_i > \frac{1}{2}\}$ and $\mathcal{S} \leftarrow \{i \in \mathcal{N}_j | 0 < u_i < 1\}$
 2: if $|\mathcal{V}|$ is even then
 3:     if $\mathcal{S} \neq \varnothing$ then
 4:         $i^* \leftarrow \arg\min_{i \in \mathcal{S}} \left|\frac{1}{2} - u_i\right|$
 5:     else
 6:         $i^* \leftarrow$ arbitrary $i \in \mathcal{N}_j$
 7:     end if
 8:     if $i^* \in \mathcal{V}$ then
 9:         $\mathcal{V} \leftarrow \mathcal{V} \setminus \{i^*\}$
10:     else
11:         $\mathcal{V} \leftarrow \mathcal{V} \cup \{i^*\}$
12:     end if
13: end if
14: if $\sum_{i \in \mathcal{V}} (1 - u_i) + \sum_{i \in \mathcal{N}_j \setminus \mathcal{V}} u_i < 1$ then
15:     Found the violated parity inequality on parity-check node $j$
16: else
17:     There is no violated parity inequality on parity-check node $j$
18:     $\mathcal{V} \leftarrow \emptyset$
19: end if
20: return $\mathcal{V}$
```

interior point decoding. In addition to providing another perspective on the CSA, the necessary condition and sufficient condition proved in Theorems 1 and 2, respectively, serve as the basis for a new adaptive approach to finding cut-inducing RPCs, as described in the next section.

5.3 LP Decoding with Adaptive Cut-Generating Algorithm

5.3.1 Generating Redundant Parity Checks

Although the addition of a redundant row to a parity-check matrix does not affect the $\mathbb{F}_2$-nullspace and, therefore, the linear code it defines, different parity-check matrix representations of a linear code may give different fundamental polytopes underlying the corresponding LP relaxation of the ML decoding problem. This fact inspires the use of cutting-plane techniques to improve the error-correcting performance of the original LP and ALP decoders. Specifically, when the LP decoder gives a nonintegral solution (i.e., a pseudocodeword), we try to find the RPCs that introduce cuts at that point, if such RPCs exist. The cuts obtained in this manner are called *RPC cuts*. The effectiveness of this method depends on how closely the new relaxation approximates the ML decoding problem, as well as on the efficiency of the technique used to search for the cut-inducing RPCs.

An RPC can be obtained by modulo-2 addition of some of the rows of the original parity-check matrix, and this new check introduces a number of linear constraints that may give a cut. In [6], a random walk on a cycle within the subgraph defined by the nonintegral entries in a pseudocodeword served as the basis for a search for RPC cuts. However, there is no guarantee that this method will find a cut (if one exists) within a finite number of iterations. In fact, the average number of random trials needed to find an RPC cut grows exponentially with the code length.

The IP-based separation algorithm in [7] performs Gaussian elimination on a submatrix comprising the columns of the original parity-check matrix that correspond to the nonintegral entries in a pseudocodeword in order to get redundant parity checks. In [8], the RPCs that potentially provide cutting planes are obtained by transforming a column-permuted version of the submatrix into row echelon form. The chosen permutation organizes the columns according to descending order of their associated nonintegral pseudocodeword entries, with the exception of the column corresponding to the largest nonintegral entry, which is placed in the rightmost position of the submatrix [8, p. 1010]. This approach was motivated by the fact that a parity check j provides a cut at a pseudocodeword if there exists a variable node in $\mathcal{N}_j$ whose value is greater than the sum of the values of all of the other neighboring variable nodes [8, Lemma 2]. However, when combined with ALP decoding, the resulting "cutting-plane algorithm" does not provide sufficiently many cuts to surpass the separation algorithm in error-rate performance.

Motivated by the new derivation of the CSA based on the conditions in Theorems 5.3 and 5.4, we next propose a new algorithm for generating cut-inducing RPCs. When used with ALP decoding, the cuts have been found empirically to achieve near-ML decoding performance in the high-SNR region for several short-to-moderate length LDPC codes. However, application of these new techniques to codes with larger block lengths proved to be prohibitive computationally, indicating that further work is required to develop practical methods for enhanced LP decoding of longer codes.

Given a nonintegral solution of the LP problem, we can see from Theorems 5.3 and 5.4 that an RPC with a small number of nonintegral neighboring variable nodes may be more likely to satisfy the necessary condition for providing a cut at the pseudocodeword. Moreover, the nonintegral neighbors should have values either close to 0 or close to 1; in other words, they should be as far from $\frac{1}{2}$ as possible.

Let $\mathbf{p} = (p_1, p_2, \ldots, p_n) \in [0,1]^n$ be a pseudocodeword solution to LP decoding, with a nonintegral positions, b zeros, and $n-a-b$ ones. We first group entries of $\mathbf{p}$ according to whether their values are nonintegral, zero, or one. Then, we sort the nonintegral positions in ascending order according to the value of $\left|\frac{1}{2} - p_i\right|$ and define the permuted vector $\mathbf{p}' = \Pi(\mathbf{p})$ satisfying the following ordering

$$\left|\frac{1}{2} - p_1'\right| \leqslant \ldots \leqslant \left|\frac{1}{2} - p_a'\right|, \tag{5.9}$$

$$p_{a+1}' = \cdots = p_{a+b}' = 0,$$

and

$$p_{a+b+1}' = \cdots = p_n' = 1.$$

By applying the same permutation Π to the columns of the original parity-check matrix $\mathbf{H}$, we get

$$\mathbf{H}' \triangleq \Pi(\mathbf{H}) = \left(\mathbf{H}^{(\mathrm{f})} | \mathbf{H}^{(0)} | \mathbf{H}^{(1)}\right) \tag{5.10}$$

where $\mathbf{H}^{(\mathrm{f})}$, $\mathbf{H}^{(0)}$, and $\mathbf{H}^{(1)}$ consist of columns of $\mathbf{H}$ corresponding to positions of $\mathbf{p}'$ with nonintegral values, zeros, and ones, respectively.

The following familiar definition from matrix theory will be useful [12, p. 10].

Definition 5.5 *A matrix is in* reduced row echelon form *if its nonzero rows (i.e., rows with at least one nonzero element) are above any all-zero rows, and the leading entry (i.e., the first nonzero entry from the left) of a nonzero row is the only nonzero entry in its column and is always strictly to the right of the leading entry of the row above it.*

By applying a suitable sequence of elementary row operations Φ (over $\mathbb{F}_2$) to $\mathbf{H}'$, we get

$$\bar{\mathbf{H}} \triangleq \Phi(\mathbf{H}') = \left(\bar{\mathbf{H}}^{(\mathrm{f})} | \bar{\mathbf{H}}^{(0)} | \bar{\mathbf{H}}^{(1)}\right), \tag{5.11}$$

where $\bar{\mathbf{H}}^{(\mathrm{f})}$ is in reduced row echelon form. Applying the inverse permutation Π^{-1} to columns of $\bar{\mathbf{H}}$, we get an equivalent parity-check matrix

$$\tilde{\mathbf{H}} = \Pi^{-1}(\bar{\mathbf{H}}) \tag{5.12}$$

whose rows are likely to be cut-inducing RPCs, for the reasons stated above.

Multiple nonintegral positions in the pseudocodeword $\mathbf{p}$ could have values with the same distance from $\frac{1}{2}$, i.e., $|\frac{1}{2} - p_i| = |\frac{1}{2} - p_j|$ for some $i \neq j$. In such a case, the ordering of the nonintegral positions in (5.9) is not uniquely determined. Hence, the set of RPCs generated by operations (5.10)–(5.12) may depend upon the particular ordering reflected in the permutation Π. Nevertheless, if the decoder uses a fixed, deterministic sorting rule such as, for example, a stable sorting algorithm, then the decoding error probability will be independent of the transmitted codeword.

The next theorem describes a situation in which a row of $\tilde{\mathbf{H}}$ is guaranteed to provide a cut.

Theorem 5.6 *If there exists a weight-one row in submatrix $\bar{\mathbf{H}}^{(\mathrm{f})}$, the corresponding row of the equivalent parity-check matrix $\tilde{\mathbf{H}}$ is a cut-inducing RPC.*

Proof: Given a pseudocodeword $\mathbf{p}$, suppose the jth row of submatrix $\bar{\mathbf{H}}^{(\mathrm{f})}$ has weight one and the corresponding nonintegral position in $\mathbf{p}$ is p_i. Since it is the only nonintegral position in $\mathcal{N}_j$, the left-hand side of (5.8) is equal to $-\left|\frac{1}{2} - p_i\right|$. Since $0 < p_i < 1$, this is larger than $-\frac{1}{2}$, the right-hand side. Hence, according to Theorem 5.4, RPC j satisfies the sufficient condition for providing a cut. In other words, there must be a violated parity inequality induced by RPC j. ■

Remark 5.2 Theorem 5.6 is equivalent to [7, Theorem 3.3]. The proof of the result shown here, though, is considerably simpler, thanks to the application of Theorem 5.4.

Although Theorem 5.6 only ensures a cut for rows with weight one in submatrix $\bar{\mathbf{H}}^{(\mathrm{f})}$, rows in $\bar{\mathbf{H}}^{(\mathrm{f})}$ of weight larger than one may also provide RPC cuts. Hence, the CSA should be applied on every row of the redundant parity-check matrix $\tilde{\mathbf{H}}$ to search for all possible RPC cuts. The approach of generating a redundant parity-check matrix $\tilde{\mathbf{H}}$ based on a given pseudocodeword and applying the CSA on each row of this matrix is called adaptive cut generation (ACG). Combining ACG with ALP decoding, we obtain the ACG-ALP decoding algorithm described in Algorithm 5.2 . Beginning with the original parity-check matrix, the algorithm iteratively applies ALP decoding. When a point is reached when no further cuts can be produced from the original parity-check matrix, the ACG technique is invoked to see whether any RPC cuts can be generated. The ACG-ALP decoding iteration stops when no more cuts can be found either from the original parity-check matrix or in the form of redundant parity checks.

Algorithm 5.2 Adaptive Linear Programming with Adaptive Cut-Generation (ACG-ALP) Decoding Algorithm

Input: cost vector γ, original parity-check matrix $\mathbf{H}$

Output: Optimal solution of current LP problem

```
1: Initialize the LP problem with the constraints in (5.3).
2: Solve the current LP problem, and get optimal solution $x^*$.
3: Apply **Algorithm 1 (CSA)** on each row of $\mathbf{H}$.
4: if No cut is found and $x^*$ is nonintegral then
5:     Construct $\tilde{\mathbf{H}}$ associated with $x^*$ according to (5.10)–(5.12).
6:     Apply **Algorithm 1 (CSA)** to each row of $\tilde{\mathbf{H}}$.
7: end if
8: if No cut is found then
9:     Terminate.
10: else
11:     Add cuts that are found into the LP problem as constraints, and go to line 2.
12: end if
```

5.3.2 Reducing the Number of Constraints in the LP Problem

In the ALP decoding, the number of constraints in the LP problem grows as the number of iterations grows, increasing the complexity of solving the LP problem. For ACG-ALP decoding, this problem becomes more severe since the algorithm generates additional RPC cuts and uses more iterations to successfully decode inputs on which the ALP decoder has failed.

From Lemma 5.1, we know that a binary parity-check constraint can provide at most one cut. Hence, if a binary parity check gives a cut, all other linear inequalities introduced by this parity check in previous iterations can be removed from the LP problem. The implementation of this observation leads to a *modified ALP (MALP)* decoder referred to as the MALP-A decoder [10]. This decoder improves the efficiency of ALP decoding, where only cuts associated with the original parity-check matrix are used. However, with ACG-ALP decoding, different RPCs may be generated adaptively in every iteration and most of them give only one cut throughout the sequence of decoding iterations. As a result, when MALP-A decoding is combined with the ACG technique, only a small number of constraints are removed from the LP problem, and the decoding complexity is only slightly reduced.

Definition 5.7 *A linear inequality constraint of the form* $\mathbf{a}^T\mathbf{x} \geqslant b$ *is called* active *at point* $\mathbf{x}^*$ *if* *it holds with equality, i.e.,* $\mathbf{a}^T\mathbf{x}^* = b$*, and is called* inactive *otherwise.*

For an LP problem with a set of linear inequality constraints, the optimal solution $\mathbf{x}^* \in [0,1]^n$ is a vertex of the polytope formed by the hyperplanes corresponding to all active constraints. In other words, if we set up an LP problem with only those active constraints, the optimal solution remains the same. Therefore, a simple and intuitive way to reduce the number of constraints is to remove all inactive constraints from the LP problem at the end of each iteration, regardless of whether or not the corresponding binary parity check generates a cut. This approach is called MALP-B decoding [10]. By combining the ACG technique and the MALP-B algorithm, we obtain the ACG-MALP-B decoding algorithm. It is similar to the ACG-ALP algorithm described in Algorithm 5.2 but includes one additional step that removes all inactive constraints from the LP problem, as indicated in Line 3 of Algorithm 5.3 .

Since adding further constraints into an LP problem reduces the feasible space, the minimum value of the cost function is non-decreasing as a function of the number of iterations. In our computer simulations, the ACG-MALP-B decoding algorithm was terminated when no further cuts could be found. (See Fig. 5.6 for statistics on the average number of iterations required to decode one codeword of the (155,64) Tanner LDPC code.)

In our implementation of both MALP-B and ACG-MALP-B decoding, we have noticed that a considerable number of the constraints deleted in previous iterations are added back into the LP problem in later iterations, and, in fact, many of them are added and deleted several times. We have observed that MALP-B-based decoding generally takes more iterations to decode a codeword than ALP-based decoding, resulting in a tradeoff between the number of iterations and the size of the constituent LP problems. MALP-B-based decoding has the largest number of iterations and the smallest LP problems to solve in each iteration, while ALP-based decoding has a smaller number of iterations but larger LP problems.

Although it is difficult to know in advance which inactive constraints might become cuts in later iterations, there are several ways to find a better tradeoff between the MALP-B and ALP techniques to speed up LP decoding. This tradeoff, however, is highly dependent on the LP solver used in the implementation. For example, we used the Simplex solver from the open-source GNU Linear Programming Kit (GLPK) [13], and found that the efficiency of iterative ALP-based decoders is closely related to the total number of constraints used to decode one codeword, i.e., the sum of the number of constraints used in all iterations. This suggests a new criterion for the removal of inactive constraints whose implementation we call the MALP-C decoder.

Algorithm 5.3 ACG-MALP-B/C Decoding Algorithm

```
Input: cost vector $\gamma$, original parity-check matrix $\mathbf{H}$
Output: Optimal solution of current LP problem
 1: Initialize LP problem with the constraints in (5.3).
 2: Solve the current LP problem, get optimal solution $x^*$.
 3: ACG-MALP-B only: remove all inactive constraints from the LP problem.
 4: ACG-MALP-C only: remove inactive constraints that have above-average slack values from
    the LP problem.
 5: Apply CSA only on rows of $\mathbf{H}$ that have not introduced constraints.
 6: if No cut is found and $x^*$ is nonintegral then
 7:     Construct $\tilde{\mathbf{H}}$ according to $x^*$
 8:     Apply CSA on each row of $\tilde{\mathbf{H}}$.
 9: end if
10: if No cut is found then
11:     Terminate.
12: else
13:     Add found cuts into LP problem as constraints, and go to line 2.
14: end if
```

In MALP-C decoding, instead of removing all inactive constraints from the LP problem in each iteration, we remove only the linear inequality constraints with slack variables that have above-average values, as indicated in Line 4 of Algorithm 5.3 . The ACG-MALP-B and ACG-MALP-C decoding algorithms are both described in Algorithm 5.3 , differing only in the use of Line 3 or Line 4. Although all three of the adaptive variations of LP decoding discussed in this chapter – ALP, MALP-B, and MALP-C – have the exact same error-rate performance as the original LP decoder, they may lead to different decoding results for a given received vector when combined with the ACG technique, as shown in the next section.

5.4 Numerical Results

To demonstrate the improvement offered by our proposed decoding algorithms, we compared their error-correcting performance to that of ALP decoding (which, again, has the same performance as the original LP decoding), BP decoding (two cases, using the sum-product algorithm with a maximum of 100 iterations and 1000 iterations, respectively), the separation algo-

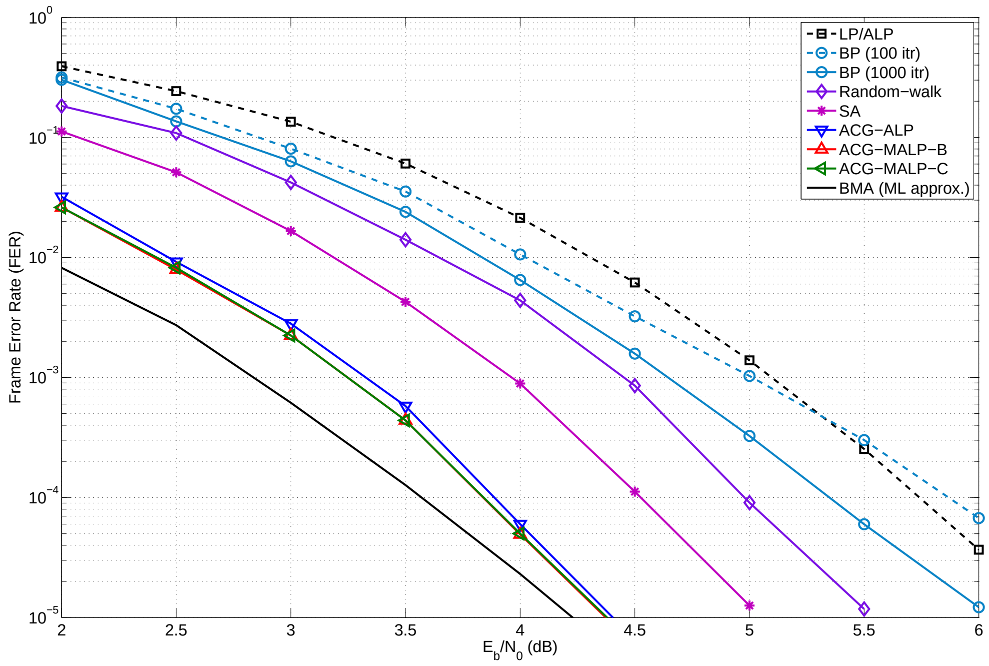


Figure 5.1: FER versus E_b/N_0 for random (3,4)-regular LDPC code of length 100 on the AWGN channel.

rithm (SA) [7], the random-walk-based RPC search algorithm [6], and ML decoding for various LDPC codes on the additive white Gaussian noise (AWGN) channel. We use the Simplex algorithm from the open-source GLPK [13] as our LP solver. The LDPC codes we evaluated are MacKay's rate-$\frac{1}{2}$, (3,6)-regular LDPC codes with lengths 96 and 408, respectively [14]; a rate-$\frac{1}{4}$, (3,4)-regular LDPC code of length 100; the rate-$\frac{2}{5}$, (3,5)-regular Tanner code of length 155 [15]; and a rate-0.89, (3,27)-regular high-rate LDPC code of length 999 [14].

The proposed ACG-ALP, ACG-MALP-B, and ACG-MALP-C decoding algorithms are all based on the underlying cut-searching algorithm (Algorithm 5.1) and the adaptive cut-generation technique of Section 5.3.1. Therefore, their error-rate performance is very similar. However, their performance may not be identical, because cuts are found adaptively from the output pseudocodewords in each iteration and the different sets of constraints used in the three proposed algorithms may lead to different solutions of the corresponding LP problems.

In our simulation, the LP solver uses double-precision floating-point arithmetic, and therefore, due to this limited numerical resolution, it may round some small nonzero vector coordinate values to 0 or output small nonzero values for vector coordinates which should be 0. Similar rounding errors may occur for coordinate values close to 1. Coordinates whose values

Table 5.1: Frame Errors of ACG-ALP decoder on MacKay's random (3,6)-regular LDPC code of length 96 on the AWGN channel.

E_b/N_0 (dB)	**Transmitted Frames**	**Error Frames**	**Pseudo-codewords**	**Incorrect Codewords**
3.0	1,136,597	3,000	857	2,143
3.5	4,569,667	3,000	395	2,605
4.0	16,724,921	3,000	103	2,897
4.5	54,952,664	3,000	12	2,988
5.0	185,366,246	3,000	0	3,000
5.5	665,851,530	3,000	0	3,000

get rounded to integers by the LP solver might lead to some "false" cuts – parity inequalities not actually violated by the exact LP solution. This is because such rounding by the LP solver would decrease the left-hand side of parity inequality (2.12). On the other hand, when coordinates that should have integer values are given nonintegral values, the resulting errors would increase the left-hand side of parity inequality (2.12), causing some cuts to be missed. Moreover, this would also increase the size of the submatrix $\mathbf{H}^{(f)}$ in (5.10), leading to higher complexity for the ACG-ALP decoding algorithm.

To avoid such numerical problems in our implementation of the CSA, we used $1 - 10^{-6}$ instead of 1 on the right-hand side of the inequality in line 14 of Algorithm 5.1 . Whenever the LP solver outputs a solution vector, coordinates with value less than 10^{-6} were rounded to 0 and coordinates with value larger than $1 - 10^{-6}$ were rounded to 1. The rounded values were then used in the cut-search and RPC-generation steps in the decoding algorithms described in previous sections. If such a procedure were not applied, and if, as a result, false cuts were to be produced, the corresponding constraints, when added into the LP problem to be solved in the next step, would leave the solution vector unchanged, causing the decoder to become stuck in an endless loop. We saw no such behavior in our decoder simulations incorporating the prescribed thresholding operations.

Finally, we want to point out that there exist LP solvers, such as *QSopt_ex Rational LP Solver* [16], that produce exact rational solutions to LP instances with rational input. However, such solvers generally have higher computational overhead than their floating-point counterparts. For this reason, we did not use an exact rational LP solver in our empirical studies.

Fig. 5.1 shows the simulation results for the length-100, regular-(3,4) LDPC code whose

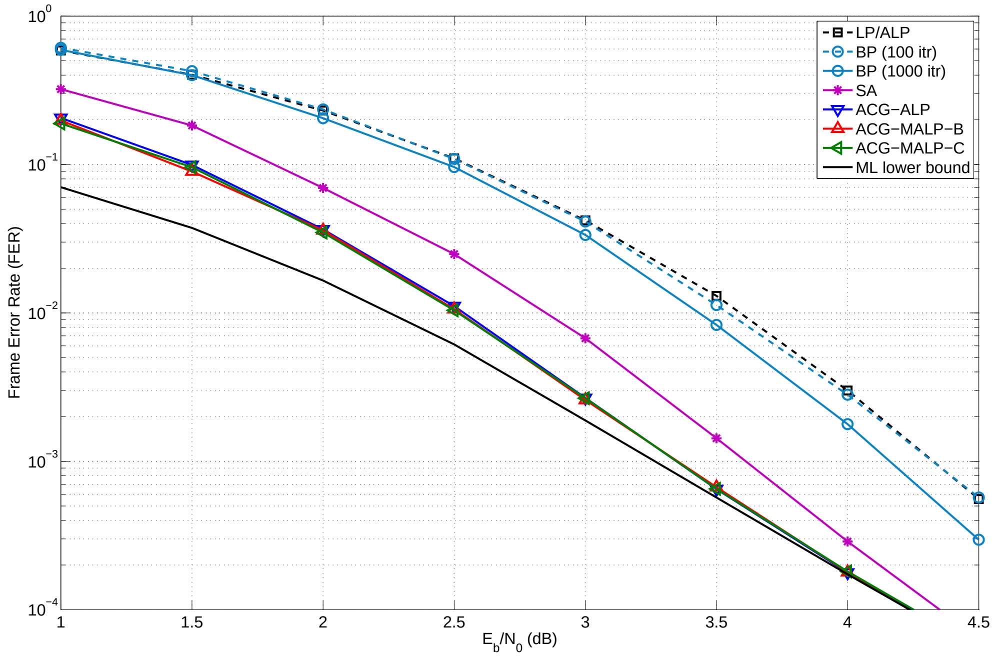


Figure 5.2: FER versus E_b/N_0 for MacKay's random (3,6)-regular LDPC code of length 96 on the AWGN channel.

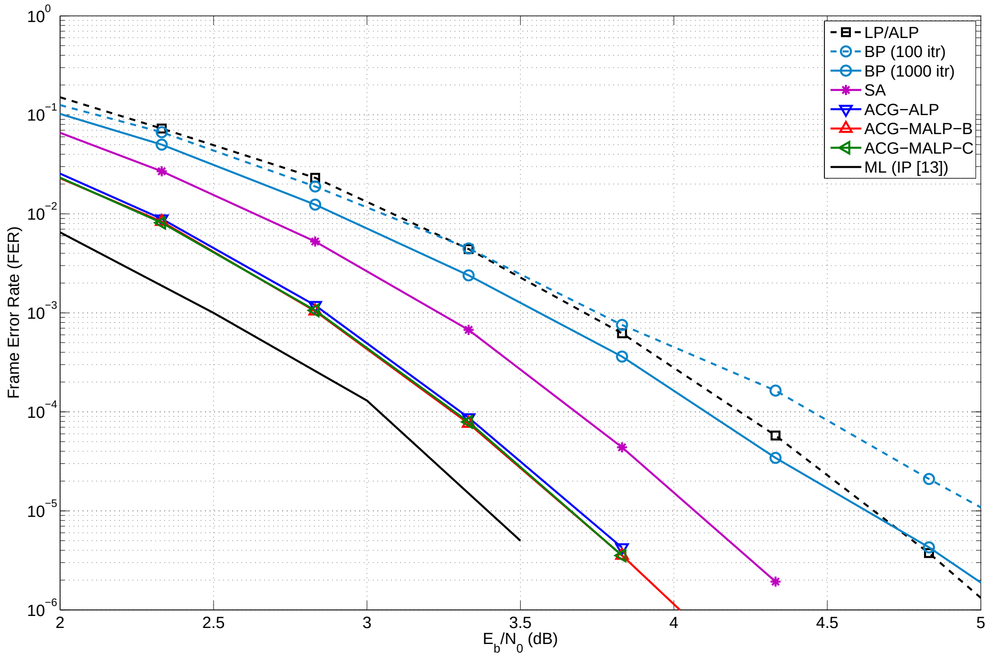


Figure 5.3: FER versus E_b/N_0 for (155,64) Tanner LDPC code on the AWGN channel.

FER performance was also evalutated in [6] and [7]. We can see that the proposed algorithms have a gain of about 2 dB over the original LP and ALP decoder. They also perform significantly better than both the separation algorithm and the random-walk algorithm. The figure also shows the results obtained with the Box-and-Match soft-decision decoding algorithm (BMA) [17], whose FER performance is guaranteed to be within a factor of 1.05 times that of ML decoding. We conclude that the performance gap between the proposed decoders and ML decoding is less than 0.2 dB at an FER of 10^{-5}.

In Fig. 5.2, we show simulation results for MacKay's length-96, (3,6)-regular LDPC code (the 96.33.964 code from [14]). Again, the proposed ALP-based decoders with ACG demonstrate superior performance to the original LP, BP, and SA decoders over the range of SNRs considered. Table 5.1 shows the actual frame error counts for the ACG-ALP decoder, with frame errors classified as either pseudocodewords or incorrect codewords; the ACG-MALP-B and ACG-MALP-C decoder simulations yielded very similar results. We used these counts to obtain a lower bound on ML decoder performance, also shown in the figure, by dividing the number of times the ACG-ALP decoder converged to an incorrect codeword by the total number of frames transmitted. Since the ML certificate property of LP decoding implies that ML decoding would have produced the same incorrect codeword in all of these instances, this ratio represents a lower bound on the FER of the ML decoder. We note that, when E_b/N_0 is greater than 4.5 dB, all decoding errors correspond to incorrect codewords, indicating that the ACG-ALP decoder has achieved ML decoding performance for the transmitted frames.

Fig. 5.3 compares the performance of several different decoders applied to the (3,5)-regular, (155,64) Tanner code, as well as the ML performance curve from [7]. It can be seen that the proposed ACG-ALP-based algorithms narrow the 1.25 dB gap between the original LP decoding and ML decoding to approximately 0.25 dB.

We also considered two longer codes, MacKay's rate-$\frac{1}{2}$, random (3,6)-regular LDPC code of length 408 (the 408.33.844 code from [14]) and a rate-0.89 LDPC code of length 999 (the 999.111.3.5543 code from [14]). Because of the increased complexity of the constituent LP problems, we only simulated the ACG-MALP-B and ACG-MALP-C decoders. In Fig. 5.4, it is confirmed that the proposed decoding algorithms provide significant gain over the original LP decoder and the BP decoder, especially in the high-SNR region. The results for the high-rate LDPC code, as shown in Fig. 5.5, again show that the proposed decoding algorithms approaches ML decoding performance for some codes, where the ML lower bound is obtained using the same technique as in Fig. 5.2. However, for the code of length 408, we found that the majority

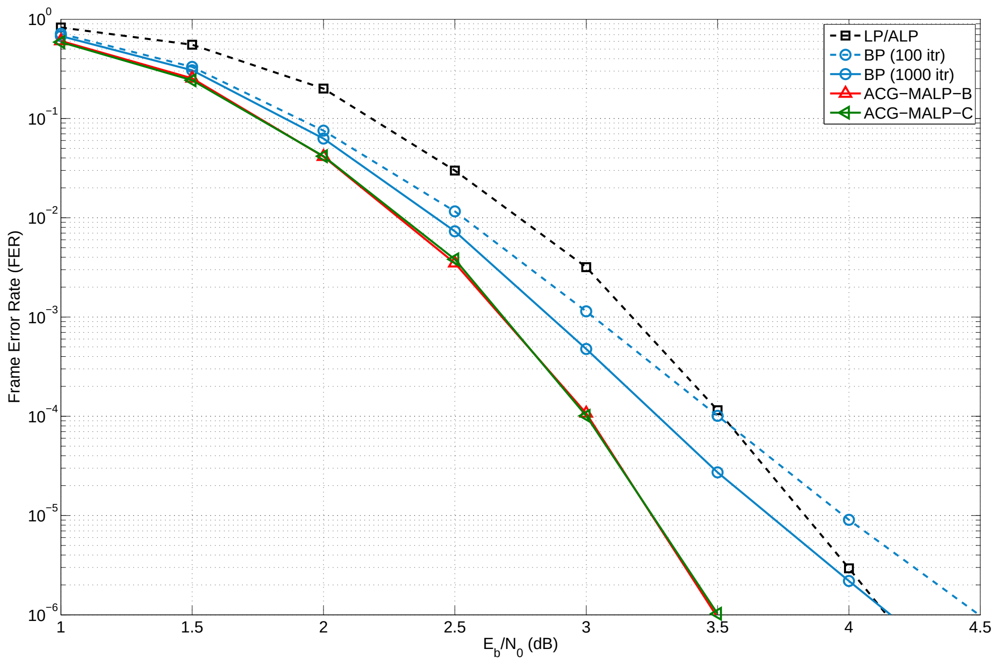


Figure 5.4: FER versus E_b/N_0 for MacKay's random (3,6)-regular LDPC code of length 408 on the AWGN channel.

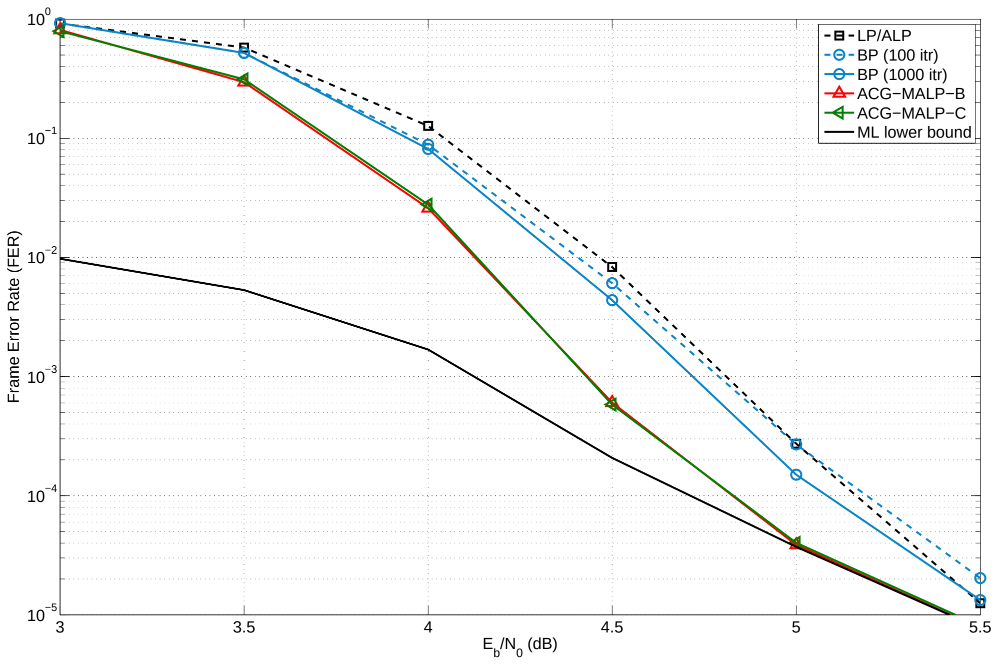


Figure 5.5: FER versus E_b/N_0 for MacKay's random (3,27)-regular LDPC code of length 999 on the AWGN channel.

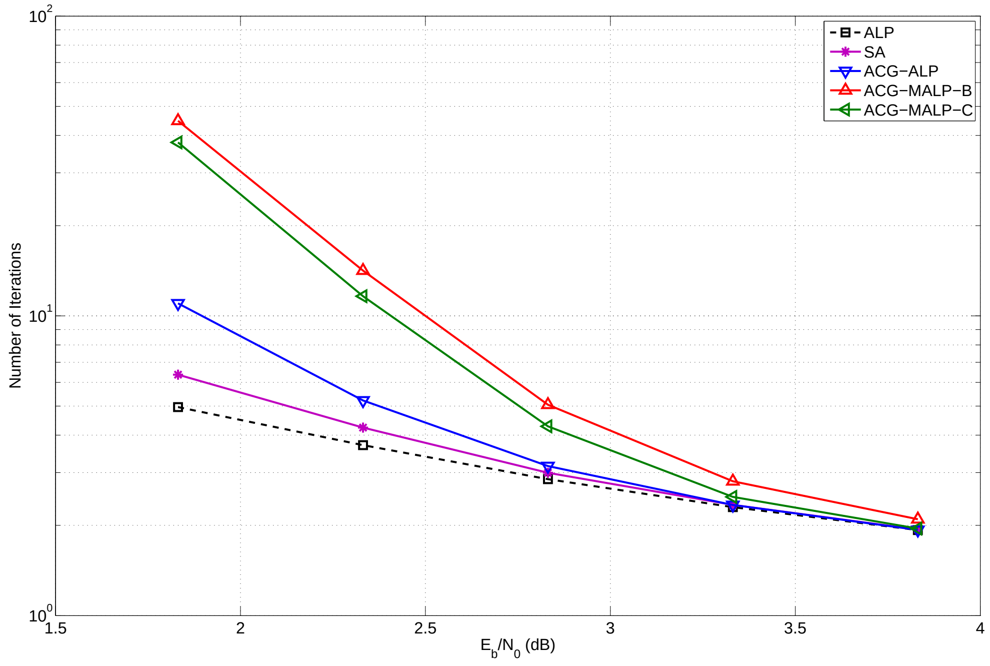
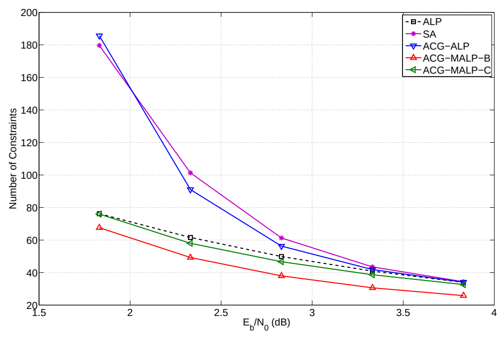


Figure 5.6: Average number of iterations for decoding one codeword of (155,64) Tanner LDPC code.

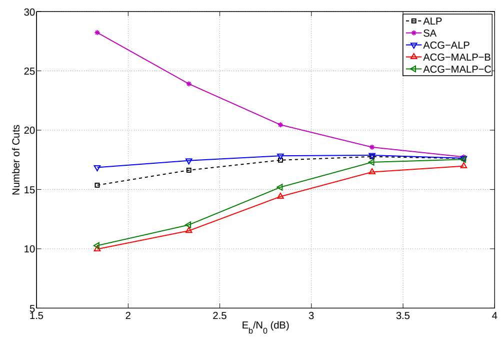
of decoding failures corresponded to pseudocodewords, so, in constrast to the case of the length-96 and length-999 MacKay codes discussed above, the frame error data do not provide a good lower bound on ML decoder performance to use as a benchmark.

Since the observed improvements in ACG-ALP-based decoder performance comes from the additional RPC cuts found in each iteration, these decoding algorithms generally require more iterations and/or the solution of larger LP problems in comparison to ALP decoding. In the remainder of this section, we empirically investigate the relative complexity of our proposed algorithms in terms of such statistics as the average number of iterations, the average size of constituent LP problems, and the average number of cuts found in each iteration. All statistical data presented here were obtained from simulations of the Tanner (155,64) code on the AWGN channel. We ran all simulations until at least 200 frame errors were counted.

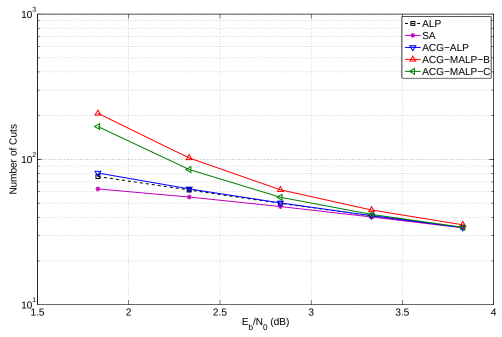
In Fig. 5.6, we compare the average number of iterations needed, i.e., the average number of LP problems solved, to decode one codeword. Fig. 5.7(a) compares the average number of constraints in the LP problem of the final iteration that results in either a valid codeword or a pseudocodeword with no more cuts to be found. In Fig. 5.7(b), we show the average number of cuts found and added into the LP problem in each iteration. Fig. 5.7(c) and Fig. 5.7(d) show the



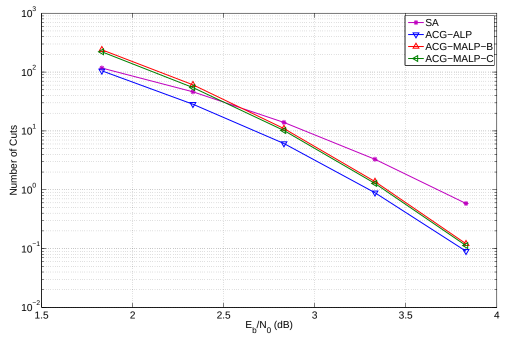
(a) Average number of constraints in final iteration



(b) Average number of cuts found per iteration



(c) Average number of cuts found from $\mathbf{H}$ for decoding one codeword



(d) Average number of cuts found from RPCs for decoding one codeword

Figure 5.7: Average number of constraints/cuts during decoding iterations for decoding one frame of (155,64) Tanner LDPC code.

Table 5.2: The average accumulated number of constraints in all iterations of decoding one codeword of (155,64) Tanner code on the AWGN channel

$\mathbf{E_b/N_0}$ **(dB)**	**ACG-ALP**	**ACG-MALP-B**	**ACG-MALP-C**
1.83	5495.8	5223.3	4643.1
2.33	1401.2	1387.3	1217.0
2.83	339.7	326.9	300.9
3.33	111.0	106.4	105.4
3.83	64.3	58.8	62.8

average number of cuts found from the original parity-check matrix $\mathbf{H}$ and from the generated RPCs, respectively.

From Fig. 5.6 and Fig. 5.7(a), we can see that, as expected, the ACG-ALP decoder takes fewer iterations to decode a codeword on average than the ACG-MALP-B/C decoders, but the ACG-MALP-B/C decoders have fewer constraints in each iteration, including the final iteration. We have observed that the ACG-MALP-B/C decoders require a larger number of iterations to decode than the ACG-ALP decoder, and fewer cuts are added into the constituent LP problems in each iteration on average, as reflected in Fig. 5.7(b). This is because there are some iterations in which the added constraints had been previously removed. Among all three proposed ACG-based decoding algorithms, we can see that the ACG-ALP decoder has the largest number of constraints in the final iteration and needs the least overall number of iterations to decode, while ACG-MALP-B decoding has the smallest number of constraints but requires the largest number of iterations. The ACG-MALP-C decoder offers a tradeoff between those two: it has fewer constraints than the ACG-ALP decoder and requires fewer iterations than the ACG-MALP-B decoder. If we use the accumulated number of constraints in all iterations to decode one codeword as a criterion to judge the efficiency of these algorithms during simulation, then ACG-MALP-C decoding is more efficient than the other two algorithms in the low and moderate SNR regions, as shown in Table 5.2. Note that the ACG-MALP-B decoder is most efficient at high SNR where the decoding of most codewords succeeds in a few iterations and the chance of a previously removed inactive constraint being added back in later iterations is quite small. Hence, ACG-MALP-B decoding is preferred in the high-SNR region.

Fig. 5.8 presents an alternative way of comparing the complexity of the decoding algorithms. It shows the average decoding time when we implement the algorithms using C++ code

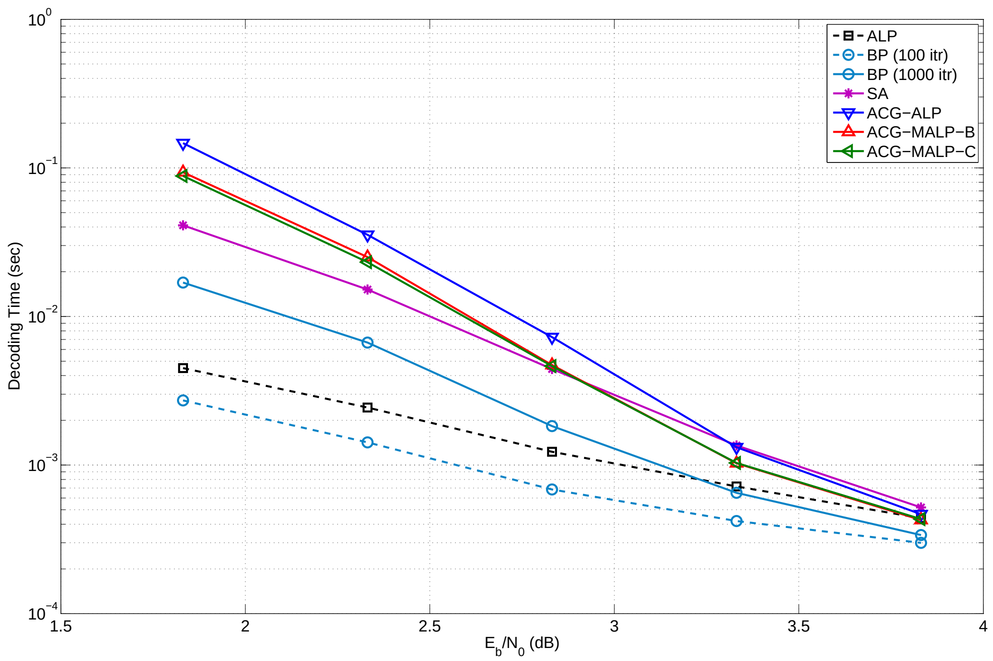


Figure 5.8: Average simulation time for decoding one codeword of (155,64) Tanner LDPC code.

on a desktop PC, with GLPK as the LP solver. The BP decoder is implemented in software with messages represented as double-precision floating-point numbers, and the exact computation of sum-product algorithm is used, without any simplification or approximation. The BP decoder iterations stop as soon as a codeword is found, or when the maximum allowable number of iterations – here set to 100 and 1000 – have been attempted without convergence. The simulation time is averaged over the number of transmitted codewords required for the decoder to fail on 200 codewords.

We observe that the ACG-MALP-B and ACG-MALP-C decoders are both uniformly faster than ACG-ALP over the range of SNR values considered, and, as expected from Table 5.2, ACG-MALP-C decoding is slightly more efficient than ACG-MALP-B decoding in terms of actual running time. Of course, the decoding time depends both on the number of LP problems solved and the size of these LP problems, and the preferred trade-off depends heavily upon the implementation, particularly the LP solver that is used. Obviously, the improvement in error-rate performance provided by all three ACG-based decoding algorithms over the ALP decoding comes at the cost of increased decoding complexity. As SNR increases, however, the average decoding complexity per codeword of the proposed algorithms approaches that of the ALP decoder. This is because, at higher SNR, the decoders can often successfully decode the received frames without generating RPC cuts.

Fig. 5.6 shows that the ACG-ALP decoder requires, on average, more iterations than the SA decoder. Our observations suggest that this is a result of the fact that the ACG-ALP decoder can continue to generate new RPC cuts after the number of iterations at which the SA decoder can no longer do so and, hence, stops decoding. The simulation data showed that the additional iterations of the ACG-ALP decoder often resulted in a valid codeword, thus contributing to its superiority in perfomance relative to the SA decoder.

From Fig. 5.7(b), it can be seen that the ACG-ALP-based decoding algorithms generate, on average, fewer cuts per iteration than the SA decoder. Moreover, as reflected in Fig. 5.7(c) and 5.7(d), the ACG-ALP decoders find more cuts from the original parity-check matrix and generate fewer RPC cuts per codeword. These observations suggest that the CSA is very efficient in finding cuts from a given parity check, while the SA decoder tends to generate RPCs even when there are still some cuts other than the Gomory cuts that can be found from the original parity-check matrix. This accounts for the fact, reflected in Fig. 5.8, that the SA becomes less efficient as SNR increases, when the original parity-check matrix usually can provide enough cuts to decode a codeword. The effectiveness of our cut-search algorithm permits the ACG-ALP-based decoders to successfully decode most codewords in the high-SNR region without generating RPCs, resulting in better overall decoder efficiency.

Due to limitations on our computing capability, we have not yet tested our proposed algorithms on LDPC codes of length greater than 1000. We note that, in contrast to [6] and [10], we cannot give an upper bound on the maximum number of iterations required by the ACG-ALP-based decoding algorithms because RPCs and their corresponding parity inequalities are generated adaptively as a function of intermediate pseudocodewords arising during the decoding process. Consequently, even though the decoding of short-to-moderate length LDPC codes was found empirically to converge after an acceptable number of interations, some sort of constraint on the maximum number of iterations allowed may have to be imposed when decoding longer codes. Finally, we point out that the complexity of the algorithm for generating cut-inducing RPCs lies mainly in the Gaussian elimination step, but as applied to binary matrices, this requires only logical operations which can be executed quite efficiently.

5.5 Conclusion

In this chapter, we derived a new necessary condition and a new sufficient condition for a parity-check constraint in a linear block code parity-check matrix to provide a violated parity inequality, or cut, at a pseudocodeword produced by LP decoding. Using these results, we presented an efficient algorithm to search for such cuts and proposed an effective approach to generating cut-inducing redundant parity checks (RPCs). The key innovation in the cut-generating approach is a particular transformation of the parity-check matrix used in the definition of the LP decoding problem. By properly re-ordering the columns of the original parity-check matrix and transforming the resulting matrix into a "partial" reduced row echelon form, we could efficiently identify RPC cuts that were found empirically to significantly improve the LP decoder performance. We combined the new cut-generation technique with three variations of adaptive LP decoding, providing a tradeoff between the number of iterations required and the number of constraints in the constituent LP problems. Frame-error-rate (FER) simulation results for several LDPC codes of length up to 999 show that the proposed adaptive cut-generation, adaptive LP (ACG-ALP) decoding algorithms outperform other enhanced LP decoders, such as the separation algorithm (SA) decoder, and significantly narrow the gap to ML decoding performance for LDPC codes with short-to-moderate block lengths.

Acknowledgment

This chapter is in part a reprint of the material in the paper: Xiaojie Zhang and Paul H. Siegel, "Adaptive Cut Generation Algorithm for Improved Linear Programming Decoding of Binary Linear Codes," *IEEE Transactions on Information Theory*, vol. 58, no. 10, pp. 6581–6594, Oct. 2012. The dissertation author was the primary investigator and author of this paper.

Bibliography

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

[2] J. Feldman, "Decoding error-correcting codes via linear programming," Ph.D. dissertation, Dept. Electr. Eng. Comput. Sci., Mass. Inst. Technol., Cambridge, MA, 2003.

[3] K. Yang, X. Wang, and J. Feldman, "A new linear programming approach to decoding linear block codes," *IEEE Trans. Information Theory*, vol. 54, no. 3, pp. 1061–1072, Mar. 2008.

[4] M. Chertkov and M. Stepanov, “Pseudo-codeword landscape,” in *Proc IEEE Int. Symp. Inf. Theory (ISIT)*, Nice, France, Jun. 2007, pp. 1546–1550.

[5] P. O. Vontobel and R. Koetter, “On low-complexity linear-programming decoding of LDPC codes,” *Europ. Trans. Telecomm.*, vol. 18, pp. 509–517, Apr. 2007.

[6] M. H. Taghavi and P. H. Siegel, “Adaptive methods for linear programming decoding,” *IEEE Trans. Information Theory*, vol. 54, no. 12, pp. 5396–5410, Dec. 2008.

[7] A. Tanatmis, S. Ruzika, H. W. Hamacher, M. Punekar, F. Kienle, and N. Wehn, “A separation algorithm for improved LP-decoding of linear block codes,” *IEEE Trans. Information Theory*, vol. 56, no. 7, pp. 3277–3289, Jul. 2010.

[8] M. Miwa, T. Wadayama, and I. Takumi, “A cutting-plane method based on redundant rows for improving fractional distance,” *IEEE J. Sel. Areas Commun.*, vol. 27, no. 6, pp. 1105–1012, Aug. 2009.

[9] M. Helmling, S. Ruzika, and A. Tanatmis, ”Mathematical programming decoding of binary linear codes: theory and algorithms,” arXiv:1107.3715 [cs.IT], to appear in *IEEE Trans. Information Theory*.

[10] M. H. Taghavi, A. Shokrollahi, and P. H. Siegel, “Efficient implementation of linear programming decoding,” *IEEE Trans. Information Theory*, vol. 57, no. 9, pp. 5960–5982, Sep. 2011.

[11] T. Wadayama, “Interior point decoding for linear vector channels based on convex optimization,” *IEEE Trans. Information Theory*, vol. 56, no. 10, pp. 4905–4921, Oct. 2010.

[12] R. A. Horn and C. R. Johnson, *Matrix Analysis*. Cambridge, UK: Cambridge University Press, 1990.

[13] GNU Linear Programming Kit, [Online]. Available: http://www.gnu.org/software/glpk

[14] D. J. C. MacKay, *Encyclopedia of Sparse Graph Codes*. [Online]. Available: http://www.inference.phy.cam.ac.uk/mackay/codes/data.html

[15] R. M. Tanner, D. Sridhara, and T. Fuja, “A class of group-structured LDPC codes,” in *Proc. Int. Symp. Commun. Theory and Appl. (ISCTA)*, Ambleside, U.K., Jul. 2001, pp. 365–369.

[16] QSopt Linear Programming Solver, [Online]. Available: http://www2.isye.gatech.edu/~wcook/qsopt/index.html

[17] A. Valembois and M. Fossorier, “Box and match techniques applied to soft decision decoding,” *IEEE Trans. Information Theory*, vol. 50, no. 5, pp. 796–810, May 2004.

Chapter 6

Efficient Iterative LP Decoding of LDPC Codes with Alternating Direction Method of Multipliers

The existing general-purpose LP solvers do not take advantage of the structure in LP decoding problem for LDPC codes which has very sparse linear inequalities as constraints. As a result, the use of off-the-shelf LP solver is computationally inefficient in this application. Moreover, for practical hardware implementations, the comparability of the LP decoding algorithm with parallel computation with values of coarse precision is highly appreciated. Therefore, an efficient LP solver that is specially designed for LP decoding problem of LDPC codes is needed. It should support parallel implementation, operate with values of coarse precision, and be flexible to accommodate different complexity requirement for various applications and scenario.

Alternating direction method of multipliers (ADMM) is a classic optimization technique that was developed in the 1970s [1], which combines the benefits of dual decomposition and augmented Lagrangian methods for constrained optimization [2, 3]. Unlike optimization algorithms such as interior-point methods for constrained problems, ADMM could be very slow to converge to solutions of high accuracy. However, it usually converges very fast to moderate accuracy, which is sufficient for many applications such as LP decoding of LDPC codes, where only integral solutions are of importance and proper rounding can be applied to non-integral solutions. The use of ADMM in LP decoding was first suggested by Barman et al. [5]. The LP problem in decoding LDPC codes is a constrained convex optimization problem, which is readily solved by ADMM techniques [2, Chapter 5]. The key part in solve constraints optimization problems with

ADMM is the method used to project a vector of real values to the constrained convex space, which is known as the fundamental polytope for LP decoding [4]. In [5], Barman *et al.* proposed a projection algorithm, which was further improved in [6]. This algorithm is based on the fact that a constrained convex polytope defined by a parity check can be expressed as the convex hull of a set of "slices," where each slice is the convex hull of all binary vectors of the same even weight. Hence, for a given vector, the algorithm first sorts the coordinates of a vector in descending order and characterizes two slices whose convex hull contains the projection; then, it finds the projection of the ordered vector by solving a quadratic program. However, the ordering of all coordinates that is required to project any given vector increases the complexity of the algorithm.

In this chapter, we propose a novel and more efficient projection algorithm for ADMM-based LP decoding. Based on the cut search algorithm (CSA) introduced in [7], the proposed algorithm can efficiently determine whether a vector is inside the check polytope without first ordering its coordinates. At the same time, it give the hyperplane that the projection must be on if the vector is outside the polytope. Our software implementation of ADMM-based LP decoding demonstrate that the proposed projection algorithm is far more computational efficient than the "two-slice" projection algorithm proposed in [5] and [6]. Based on the property of pseudocodeword we found, several methods can be applied to ADMM-based LP decoding to further improve the decoding efficiency for practical implementation. We also extend the method of penalizing the objective function of LP decoding [8] into a more general form as well as the way of combing it with ADMM-based LP decoding. Comparing to the penalty functions proposed in [8], our new approach provides better error-rate performance, especially on irregular LDPC codes, without any complexity increase.

The remainder of the chapter is organized as follows. In Section 6.1, we review the formulation of LP decoding and its ADMM presentation. In Section 6.2, we describe the proposed projection algorithm. Section 6.3 discusses several methods of further improving the decoding efficiency as well as the error-rate performance. Section 6.4 presents some complexity analysis and numerical results, and Section 6.5 concludes the chapter.

6.1 ADMM Formulation of LP Decoding Problem

In order to facilitate the formulation of ADMM-base LP decoding, we first introduce some useful definitions.

Definition 6.1 *Let the $d_j \times n$ binary matrix $\mathbf{T}_j$ be the transfer matrix corresponding to check j such that*

$$T_{j(k,l)} = \begin{cases} 1 & \text{if } \mathcal{N}_j(k) = l \\ 0 & \text{otherwise.} \end{cases}$$

The transfer matrix $\mathbf{T}_j$ selects out all d_j neighboring variables of check j. For example, if the jth row of parity-check matrix $\mathbf{H}$ is $\mathbf{h}_j = (0,1,0,1,0,1,0)$, then the corresponding transfer matrix is

$$\mathbf{T}_j = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}.$$

Hence, $\mathbf{T}_j\mathbf{x}$ has the dimension of $|\mathcal{N}_j|$, where for a set $\mathcal{X}$, $|\mathcal{X}|$ denotes its cardinality.

Definition 6.2 *The* check polytope*, $\mathcal{P}_d$, is the convex hull of all binary vectors of length d with an even number of 1s, i.e.,*

$$\mathcal{P}_d \triangleq \text{conv}\left(\left\{\mathbf{x} \in \{0,1\}^d \mid \mathbf{x} \text{ has an even number of 1s}\right\}\right).$$

Note that the definition of check polytope is similar to that of *parity polytope* [9] where the number of 1s in $\mathbf{x}$ is odd instead.

With these notations, we can rewrite the LP decoding problem (2.11) into the following form

$$\begin{aligned} \text{minimize} \quad & \boldsymbol{\gamma}^T\mathbf{u} \\ \text{subject to} \quad & \mathbf{T}_j\mathbf{u} \in \mathcal{P}_{d_j} \quad \forall j \in \mathcal{J}. \end{aligned} \tag{6.1}$$

Beside using general-purpose LP solvers, another way to solve the optimization problem (6.1) is to use ADMM, which is intended to combine the decomposability of dual ascent with the superior convergence properties of the method of multipliers [2]. In order to make the LP decoding problem perfectly fit the ADMM template given in [2, p. 33], we first rewrite (6.1) as follows

$$\begin{aligned} \text{minimize} \quad & \boldsymbol{\gamma}^T\mathbf{x} \\ \text{subject to} \quad & \mathbf{T}_j\mathbf{x} = \mathbf{z}_j,\ \mathbf{z}_j \in \mathcal{P}_{d_j},\ \forall j \in \mathcal{J}. \end{aligned} \tag{6.2}$$

Then, the augmented Lagrangian of (6.2) is

$$L_\rho(\mathbf{x},\mathbf{z},\boldsymbol{\lambda}) = \boldsymbol{\gamma}^T\mathbf{x} + \sum_{j\in\mathcal{J}} \boldsymbol{\lambda}_j^T(\mathbf{T}_j\mathbf{x} - \mathbf{z}_j) + \frac{\rho}{2}\sum_{j\in\mathcal{J}} \|\mathbf{T}_j\mathbf{x} - \mathbf{z}_j\|_2^2, \tag{6.3}$$

where $\boldsymbol{\lambda}_j \in \mathbb{R}^{d_j}$ is *dual variable* and $\rho > 0$ is called the *penalty parameter*.

So, the ADMM solves this problem by iterating the following equations on k:

$$\mathbf{x}^{k+1} := \underset{\mathbf{x}}{\operatorname{argmin}} L_\rho \left(\mathbf{x}, \mathbf{z}^k, \boldsymbol{\lambda}^k\right) \tag{6.4}$$

$$\mathbf{z}^{k+1} := \underset{\forall j : \mathbf{z}_j \in \mathcal{P}_{d_j}}{\operatorname{argmin}} L_\rho \left(\mathbf{x}^{k+1}, \mathbf{z}, \boldsymbol{\lambda}^k\right) \tag{6.5}$$

$$\boldsymbol{\lambda}_j^{k+1} := \boldsymbol{\lambda}_j^k + \rho \left(\mathbf{T}_j \mathbf{x}^{k+1} - \mathbf{z}_j^{k+1}\right), \ \forall j \in \mathcal{J} \tag{6.6}$$

In words, the augmented Lagrangian L_ρ is first minimized with respect to $\mathbf{x}$ while keeping $\mathbf{z}$ and $\boldsymbol{\lambda}$ fixed at $\mathbf{z}^k$ and $\boldsymbol{\lambda}^k$, respectively. Then, by fixing $\mathbf{x}$ at the new value $\mathbf{x}^{k+1}$ and $\boldsymbol{\lambda}$ still at $\boldsymbol{\lambda}^k$, L_ρ is minimized with respect to $\mathbf{z}$ such that $\mathbf{z}_j \in \mathcal{P}_{d_j}$ for all $j \in \mathcal{J}$. Lastly, the scaled dual variable $\boldsymbol{\lambda}$ is updated with the new $\mathbf{x}^{k+1}$ and $\mathbf{z}^{k+1}$. It has been proved in [3, p. 256] that, for optimization problem (6.2), the $\mathbf{x}^k$, $\boldsymbol{\lambda}^k$, and $\mathbf{z}^k$ in the ADMM iterates all converge to their unique optimal values.

Denote by $\mathbf{y} = \boldsymbol{\lambda}/\rho$ the *scaled dual variable*, the augmented Lagrangian can be expressed as

$$L_\rho(\mathbf{x}, \mathbf{z}, \mathbf{y}) = \boldsymbol{\gamma}^T \mathbf{x} + \frac{\rho}{2} \sum_{j \in \mathcal{J}} \|\mathbf{T}_j \mathbf{x} - \mathbf{z}_j + \mathbf{y}_j\|_2^2 - \frac{\rho}{2} \sum_{j \in \mathcal{J}} \|\mathbf{y}_j\|_2^2. \tag{6.7}$$

Using the scaled dual variable, the ADMM iterates described in (6.4)–(6.6) can be further simplified as

$$\mathbf{x}^{k+1} := \underset{\mathbf{x}}{\operatorname{argmin}} \left(\boldsymbol{\gamma}^T \mathbf{x} + \frac{\rho}{2} \sum_{j \in \mathcal{J}} \|\mathbf{T}_j \mathbf{x} - \mathbf{z}_j^k + \mathbf{y}_j^k\|_2^2\right) \tag{6.8}$$

$$\mathbf{z}_j^{k+1} := \underset{\mathbf{z}_j \in \mathcal{P}_{d_j}}{\operatorname{argmin}} \|\mathbf{T}_j \mathbf{x}^{k+1} + \mathbf{y}_j^k - \mathbf{z}_j\|_2^2, \ \forall j \in \mathcal{J} \tag{6.9}$$

$$\mathbf{y}_j^{k+1} := \mathbf{y}_j^k + \mathbf{T}_j \mathbf{x}^{k+1} - \mathbf{z}_j^{k+1}, \ \forall j \in \mathcal{J}, \tag{6.10}$$

To compute the minimum in (6.8), differentiating (6.8) with respect to $\mathbf{x}$ and setting the result to zero, we obtain

$$\boldsymbol{\gamma} + \rho \sum_{j \in \mathcal{J}} \mathbf{T}_j^T \left(\mathbf{T}_j \mathbf{x} - \mathbf{z}_j^k + \mathbf{y}_j^k\right) = 0,$$

which implies

$$\sum_{j \in \mathcal{J}} \mathbf{T}_j^T \mathbf{T}_j \mathbf{x} = \sum_{j \in \mathcal{J}} \mathbf{T}_j^T \left(\mathbf{z}_j^k - \mathbf{y}_j^k\right) - \frac{\boldsymbol{\gamma}}{\rho}.$$

We then replace the update rule for $\mathbf{x}$ in (6.8) as

$$\mathbf{x}^{k+1} := \mathbf{T}^{-1} \times \left(\sum_{j \in \mathcal{J}} \mathbf{T}_j^T \left(\mathbf{z}_j^k - \mathbf{y}_j^k\right) - \frac{\boldsymbol{\gamma}}{\rho}\right), \tag{6.11}$$

where $\mathbf{T} = \sum_{j\in\mathcal{J}} \mathbf{T}_j^T \mathbf{T}_j$. From Definition 6.1, it is easy to see that $\mathbf{T}$ is an $n \times n$ diagonal matrix with $\mathbf{T}_{(i,i)} = d_i,\ i \in \mathcal{I}$. Therefore, $\mathbf{T}^{-1}$ is also an $n \times n$ diagonal matrix, with $\mathbf{T}^{-1}_{(i,i)} = 1/d_i,\ i \in \mathcal{I}$. In (6.9), the minimization of $\mathbf{z}^j$ is actually finding the Euclidean projection of vector $(\mathbf{T}_j\mathbf{x}^{k+1} + \mathbf{y}_j^k) \in \mathbb{R}_j^d$ in the check polytope $\mathcal{P}_{d_j}$. We denote by $\Pi_{\mathcal{P}_d}(\mathbf{u})$ the Euclidean projection of vector $\mathbf{u}$ on $\mathcal{P}_d$.

From (6.9) and (6.10), it can be seen that, since $\mathbf{T}_j\mathbf{x}$ selects variable x_i in $\mathbf{x}$ which are neighbors of check j in the Tanner graph, i.e., $i \in \mathcal{N}_j$, the update of $\mathbf{z}_j$ and $\mathbf{y}_j$ is analogues to a check update in message-passing decoding, which requires only local information from neighboring nodes. Let $(z_j)_i$, $i \in \mathcal{N}_j \subseteq \mathcal{I}$, be the element of $\mathbf{z}_j$ that corresponds to x_i, i.e., the ith coordinate of $\mathbf{T}_j^T\mathbf{z}_j$. Note that $(z_j)_i$ is not the ith element of $\mathbf{z}_j$ but the element corresponding to the ith element in $\mathbf{x}$. We define $(y_j)_i$ similarly. Then, we can rewrite (6.11) in the form of update rule for each coordinate of $\mathbf{x}$ as follows

$$x_i^{k+1} := \frac{1}{d_i}\left(\sum_{j\in\mathcal{N}_i}\left((z_j)_i^k - (y_j)_i^k\right) - \frac{\gamma_i}{\rho}\right). \tag{6.12}$$

From (6.12), we can see that the update of x_i only requires information from neighboring checks of variable i.

Hence, we can express the ADMM decoding in the form of an iterative message-passing algorithm, as described in Algorithm 6.1 . Each check j updates its outgoing messages based on the received messages from its neighboring variables (i.e., x_i for $i \in \mathcal{N}_j$), and its locally stored $\mathbf{y}_j$ from previous iteration. For the variable update, as shown in (6.17), the computation on each variable only involves the incoming messages from neighboring checks. Similar to other message-passing decoding algorithms, one can compute the updates for all the checks simultaneously, as well as for all the variables.

In Step 1 of Algorithm 6.1 , there can be different ways to initialize $\mathbf{x}$. For example, we can set

$$x_i = \begin{cases} 0 & \text{if } \gamma_i \geqslant 0 \\ 1 & \text{if } \gamma_i < 0. \end{cases} \tag{6.20}$$

If this $\mathbf{x}$ is not a valid codeword, then one starts the ADMM decoding; otherwise, it is the ML codeword. The scaled dual variable $\mathbf{y}_j$ can be set to be all zeros for all $j \in \mathcal{J}$. The initialization of $\mathbf{z}_j$ does not affect the ADMM decoding. Note that different initializations of $\mathbf{x}$, $\mathbf{z}_j$, and $\mathbf{y}_j$ can only affect the number of iterations required to converge, and they would not change the optimal solution obtained by ADMM (within the feasibility tolerances).

Algorithm 6.1 Iterative ADMM-based LP decoding algorithm

1: **Initialization:** Properly initialize $\mathbf{x}$, $\mathbf{z}_j$, and $\mathbf{y}_j$ $\forall j \in \mathcal{J}$.

2: **Check update:** For each check $j \in \mathcal{J}$, update $\mathbf{z}_j$ and $\mathbf{y}_j$ using (6.9) and (6.10), respectively. We rewrite them as follows

$$\mathbf{w} \leftarrow \mathbf{T}_j\mathbf{x} + \mathbf{y}_j \tag{6.13}$$

$$\mathbf{z}_j \leftarrow \Pi_{\mathcal{P}_{d_j}}(\mathbf{w}) \tag{6.14}$$

$$\mathbf{y}_j \leftarrow \mathbf{w} - \mathbf{z}_j, \tag{6.15}$$

where (6.13) uses only the values of neighboring variables of check j. So, $\mathbf{T}_j\mathbf{x}$ can also be viewed as vector $[x_i]_{i\in\mathcal{N}_j}$ which has x_i, $i \in \mathcal{N}_j$, as it elements. Then, the message transmitted to neighboring variables is

$$L_{j\to i} \leftarrow (z_j)_i - (y_j)_i. \tag{6.16}$$

3: **Variable update:** For each variable $i \in \mathcal{I}$, the messages transmitted to its neighboring checks are the same, and is computed as (6.12) which can be rewritten as follows

$$x_i \leftarrow \frac{1}{d_i}\left(\sum_{j'\in\mathcal{N}_i} L_{j'\to i} - \frac{\gamma_i}{\rho}\right). \tag{6.17}$$

4: **Stopping criteria:** If

$$\sum\nolimits_{j\in\mathcal{J}} \|\mathbf{T}_j\mathbf{x} - \mathbf{z}_j\|_2 < \epsilon^{\text{pri}} \tag{6.18}$$

and

$$\sum\nolimits_{j\in\mathcal{J}} \|\mathbf{z}_j^k - \mathbf{z}_j^{k-1}\|_2 < \epsilon^{\text{dual}}, \tag{6.19}$$

where $\epsilon^{\text{pri}} > 0$ and $\epsilon^{\text{dual}} > 0$ are feasibility tolerances, the ADMM converges and $\mathbf{x}$ is the optimal solution; otherwise, go to Step 2.

In Algorithm 6.1 , except for the projection Π_{d_j} in (6.14), the computations are quite straightforward, involving only additions and multiplications, and are standard update procedures of ADMM taken from [2]. However, the projection Π_{d_j} of a given vector onto the check polytope of dimension d_j has to be specially designed, and the efficiency of the projection algorithm directly determines the complexity of ADMM decoding. In next section, we will introduce the key contribution in this chapter, which is an efficient Euclidean projection algorithm which projects any given vector onto the check polytope.

6.2 Efficient Projection onto Check Polytope

In [5] and [6], two projection algorithms were proposed, based on the so-called "two-slice" representation of any vector in the check polytope, according to which any given vector can be expressed as a convex combination of two binary vectors of Hamming weight r and $r+2$, for some even integer r. The Majorization Theorem [10] is used to characterize the convex hull of the two slices, where a sorting on all coordinates of the given vector is required to decide whether it is within the check polytope. If the given vector is outside the polytope, the projection algorithm includes two steps: first projected the vector onto a scaled version of one of the two slices, and then project the residual onto another scaled slice [5]. However, it is complex to find the proper scaling value. This projection was further improved in [6], where the projection problem is then expressed as a quadratic program after the two slices have been characterized by the majorization method.

In this section, we proposed a novel and more efficient projection algorithm which utilizes the cut search algorithm (CSA) introduced in [7], and we will show that the cut found by CSA is the facet of the polytope on which the projection point must be located. Then, we will give an efficient algorithm that projects the vector onto this facet.

6.2.1 Cut Search Algorithm

As we mentioned in Chapter 2, the check polytope $\mathcal{P}_d$ of dimension d can be described by a set of box constraints and parity inequalities as follows

$$0 \leqslant u_i \leqslant 1,\ \forall i \in [d] \tag{6.21}$$

$$\sum_{i\in\mathcal{V}} u_i - \sum_{i\in\mathcal{V}^c} u_i \leqslant |\mathcal{V}| - 1,\ \forall \mathcal{V} \subseteq [d] \text{ with } |\mathcal{V}| \text{ odd.} \tag{6.22}$$

Algorithm 6.2 Cut search algorithm

```
Input: vector $\mathbf{u} \in [0,1]^d$.
Output: indicator vector $\boldsymbol{\theta} \in \{-1,1\}^d$ of cutting set $\mathcal{V}$ (if exists)
 1: $\theta_i \leftarrow \begin{cases} 1 & \text{if } u_i > \frac{1}{2} \\ -1 & \text{otherwise} \end{cases}$
 2: if the cardinality of set $\{i : \theta_i > 0\}$ is even then
 3:     $i^* \leftarrow \underset{i}{\operatorname{argmin}} \left|\frac{1}{2} - u_i\right|$.
 4:     $\theta_{i^*} \leftarrow -\theta_{i^*}$.
 5: end if
 6: $\mathcal{V} = \{i : \theta_i > 0\}$.
 7: if $\boldsymbol{\theta}^T\mathbf{u} > |\mathcal{V}| - 1$ then
 8:     return $\boldsymbol{\theta}$ is the indicator vector of cutting set $\mathcal{V}$, and $\mathbf{u} \notin \mathcal{P}_d$.
 9: else
10:     return No cut found, and $\mathbf{u} \in \mathcal{P}_d$.
11: end if
```

Define by $\boldsymbol{\theta}_{\mathcal{V}}$ the *indicator vector* of set $\mathcal{V}$ such that

$$\theta_{V,i} = \begin{cases} 1 & \text{if } i \in \mathcal{V} \\ -1 & \text{if } i \in \mathcal{V}^c, \end{cases}$$

where $\mathcal{V}^c = [d] \setminus \mathcal{V}$ is the complement of $\mathcal{V}$. Then, the inequality (6.22) can be written as

$$\boldsymbol{\theta}_{\mathcal{V}}^T\mathbf{u} \leqslant |\mathcal{V}| - 1.$$

For a given vector $\mathbf{u} \in [0,1]^d$, if there exists a set $\mathcal{V} \in [d]$ of odd cardinality such that

$$\boldsymbol{\theta}_{\mathcal{V}}^T\mathbf{u} > |\mathcal{V}| - 1, \tag{6.23}$$

then $\mathbf{u}$ is outside the check polytope. The violated parity inequality (6.23) is called a *cut* on $\mathbf{u}$, and the corresponding $\mathcal{V}$ of odd cardinality is called a *cutting set*. The following result shows that, for a give vector, there exists at most one cutting set.

Proposition 6.3 (Th. 1 in [11]) *If at any given point* $\mathbf{u} \in [0,1]^d$*, one of the parity inequalities in* (6.22) *is violated, the rest of them are satisfied with strict inequality.*

With the notation introduced in this chapter, the efficient cut search algorithm, described in Chapter 5, can be rewritten as Algorithm 6.2 . It is obvious that, to determine whether a given vector is inside the check polytope or not, the algorithm only needs to visit every coordinate once, and no ordering of the coordinates is required.

6.2.2 Projection to Check Polytope

It can be seen from Definition 6.2 that the check polytope $\mathcal{P}_d$ lies inside the $[0,1]^d$ hypercube, i.e., $\mathcal{P}_d \subset [0,1]^d$. This means that all points outside the hypercube are also outside the check polytope. To find the check polytope projection of a vector outside the hypercube, we first check with CSA whether or not its projection on the hypercube is also in the check polytope. If the hypercube projection is outside the check polytope, the cut found by CSA can be further used to find the check polytope projection, as will be shown later in this subsection.

The projection of any vector onto the hypercube can be done easily by applying a threshold on all coordinates. Let $\mathbf{z} = \Pi_{[0,1]^d}(\mathbf{u})$ be the $[0,1]^d$ projection of vector $\mathbf{u} \in \mathbb{R}^d$; then we have

$$z_i = \begin{cases} 1 & \text{if } u_i > 1 \\ 0 & \text{if } u_i < 0 \\ u_i & \text{otherwise.} \end{cases} \tag{6.24}$$

It is easy to verify that the $\mathbf{z}$ computed in (6.24) is indeed the Euclidean projection of $\mathbf{u}$ to the $[0,1]^d$ hypercube. If $\mathbf{z} \in \mathcal{P}_d$, then $\mathbf{z}$ is exactly the Euclidean projection of $\mathbf{u}$ to the check polytope $\mathcal{P}_d$, and the projection is done. Therefore, in the remaining part of this subsection, we focus on vectors whose $[0,1]^d$ projection $\mathbf{z} = \Pi_{[0,1]^d}(\mathbf{u})$ is outside the check polytope, i.e., $\mathbf{z} \notin \mathcal{P}_d$. From Proposition 6.3, there exists only one set $\mathcal{V}$ of odd cardinality such that $\boldsymbol{\theta}_{\mathcal{V}}^T\mathbf{z} > |\mathcal{V}| - 1$. We have the following relationship between $\mathbf{u}$ and its $[0,1]^d$ projection $\mathbf{z}$.

Proposition 6.4 *Given a vector $\mathbf{u} \in \mathbb{R}^d$, let $\mathbf{z} = \Pi_{[0,1]^d}(\mathbf{u})$. If there exits a cut $\boldsymbol{\theta}_{\mathcal{V}}^T\mathbf{z} > |\mathcal{V}| - 1$, then $u_i \geqslant z_i$ for all $i \in \mathcal{V}$ and $u_i \leqslant z_i$ for all $i \in \mathcal{V}^c$. This implies $\boldsymbol{\theta}_{\mathcal{V}}^T\mathbf{u} \geqslant \boldsymbol{\theta}_{\mathcal{V}}^T\mathbf{z}$, where the equality holds if and only if $\mathbf{z} = \mathbf{u}$.*

Proof: Suppose there exists $j \in \mathcal{V}$ such that $u_j < z_j$. Then by (6.24), $z_j = 0$. Since $0 \leqslant z_i \leqslant 1$ for all $i \in [d]$, we have $\boldsymbol{\theta}_{\mathcal{V}}^T\mathbf{z} \leqslant \sum_{i\in\mathcal{V}} z_i \leqslant |\mathcal{V}| - 1$. This contradicts the assumption that $\boldsymbol{\theta}_{\mathcal{V}}^T\mathbf{z} > |\mathcal{V}| - 1$; hence $u_i \geqslant z_i$ for all $i \in \mathcal{V}$. Similarly, we can show that $u_i \leqslant z_i$ for all $i \in \mathcal{V}^c$. ■

Proposition 6.4 can be extended in the following corollary.

Corollary 6.5 *If $\mathcal{V}$ is the cutting set on $\mathbf{z} = \Pi_{[0,1]^d}(\mathbf{u})$, then $u_k > \sum_{i\in\mathcal{V}^c} u_i$ holds for any $k \in \mathcal{V}$.*

Proof: Notice that $z_k > \sum_{i\in\mathcal{V}^c} z_i$ for any $k \in \mathcal{V}$, because $|\mathcal{V}| - 1 < \sum_{i\in\mathcal{V}} z_i - \sum_{i\in\mathcal{V}^c} z_i \leqslant \sum_{i\in\mathcal{V}} z_i \leqslant |\mathcal{V}|$ holds for cutting set $\mathcal{V}$. Then, by Proposition 6.4, the corollary is proved. ■

The proposed efficient projection algorithm is based on the following result.

Theorem 6.6 *For a given vector $\mathbf{u} \in \mathbb{R}^d$, let $\mathbf{z} = \Pi_{[0,1]^d}(\mathbf{u})$. If there exists a cutting set $\mathcal{V}$ on $\mathbf{z}$ such that $\boldsymbol{\theta}_{\mathcal{V}}^T\mathbf{z} > |\mathcal{V}| - 1$, then the Euclidean projection of $\mathbf{u}$ to the check polytope $\mathcal{P}_d$ must be on the facet of $\mathcal{P}_d$ given by $\mathcal{V}$, i.e., $\Pi_{\mathcal{P}_d}(\mathbf{u}) \in \mathcal{F}_{\mathcal{V}} \triangleq \left\{\mathbf{x} \in [0,1]^d \mid \boldsymbol{\theta}_{\mathcal{V}}^T\mathbf{x} = |\mathcal{V}| - 1\right\}$.*

Proof: First of all, from Proposition 6.3, we know that the cutting set $\mathcal{V}$ that gives a cut on $\mathbf{z}$ is unique, and $\mathbf{x} \in \mathcal{P}_d$ if $\mathbf{x} \in \mathcal{F}_{\mathcal{V}} \cap [0,1]^d$. Suppose that $\mathbf{w} \in \mathcal{P}_d$ is the projection of $\mathbf{u}$ to $\mathcal{P}_d$ but it is not on the facet $\mathcal{F}_{\mathcal{V}}$; then, it must satisfy $\boldsymbol{\theta}_{\mathcal{V}}^T\mathbf{w} < |\mathcal{V}| - 1$.

Let $a = \boldsymbol{\theta}_{\mathcal{V}}^T\mathbf{z} - |\mathcal{V}| + 1 > 0$, $b = |\mathcal{V}| - 1 - \boldsymbol{\theta}_{\mathcal{V}}^T\mathbf{w} > 0$, and let vector $\mathbf{v} = \frac{b}{a+b}\mathbf{z} + \frac{a}{a+b}\mathbf{w}$. Then,

$$\boldsymbol{\theta}_{\mathcal{V}}^T\mathbf{v} = \frac{b}{a+b}\boldsymbol{\theta}_{\mathcal{V}}^T\mathbf{z} + \frac{a}{a+b}\boldsymbol{\theta}_{\mathcal{V}}^T\mathbf{w} = |\mathcal{V}| - 1,$$

which implies $\mathbf{v} \in \mathcal{F}_{\mathcal{V}}$. The vector $\mathbf{v}$ is also in $[0,1]^d$ because both $\mathbf{z}$ and $\mathbf{w}$ are in the convex set $[0,1]^d$. Hence, by Proposition 6.3, we have $\mathbf{v} \in \mathcal{P}_d$.

Since $\mathbf{z}$ is the projection of $\mathbf{u}$ on the $[0,1]^d$ hypercube, then for any $\mathbf{x} \in [0,1]^d$, we have the following equality from (6.24),

$$|u_i - x_i| = |u_i - z_i| + |z_i - x_i|, \quad 1 \leqslant i \leqslant d. \tag{6.25}$$

From the definition of $\mathbf{v}$, we have

$$z_i - v_i = \frac{a}{a+b}(z_i - w_i), \quad 1 \leqslant i \leqslant d. \tag{6.26}$$

Setting $\mathbf{x} = \mathbf{v}$ and substituting (6.26) into (6.25), we get

$$\begin{aligned} |u_i - v_i| &= |u_i - z_i| + \frac{a}{a+b}|z_i - w_i| \\ &\leqslant |u_i - z_i| + |z_i - w_i| \\ &= |u_i - w_i|. \end{aligned}$$

Since $\mathbf{z} \neq \mathbf{w}$, the above inequality can not hold with equality for all i, hence $\|\mathbf{u} - \mathbf{v}\|_2^2 < \|\mathbf{u} - \mathbf{w}\|_2^2$. This contradicts the assumption that $\mathbf{w}$ is the Euclidean projection of $\mathbf{u}$ on $\mathcal{P}_d$. ■

Remark 6.1 Since the check polytope is convex, the Euclidean projection of any vector on it is unique.

Now the only thing needed to complete the projection is an efficient algorithm to solve the following optimization problem for any given vector $\mathbf{u} \notin \mathcal{P}^d$,

$$\begin{aligned} \text{minimize} \quad & \|\mathbf{u} - \mathbf{x}\|_2^2 \\ \text{subject to} \quad & 0 \leqslant x_i \leqslant 1, \quad i = 1, \ldots, d \\ & \boldsymbol{\theta}^T \mathbf{x} = b, \end{aligned} \tag{6.27}$$

where $\boldsymbol{\theta}$ is the indicator vector of cutting set $\mathcal{V}$ (here we omitted the subscript $\mathcal{V}$ of $\boldsymbol{\theta}_{\mathcal{V}}$ to simply the notation) and $b = |\mathcal{V}| - 1$.

Since the optimization problem (6.27) has differentiable convex objective and constraint functions, any points that satisfy the Karush-Kuhn-Tucker (KKT) conditions are optimal solutions [12]. Introducing Lagrange multipliers $\boldsymbol{\lambda} \in \mathbb{R}^d$ for the inequality constraints $\mathbf{x} \geqslant 0$, $\boldsymbol{\mu} \in \mathbb{R}^d$ for the inequality constraints $\mathbf{x} \leqslant 1$, and a multiplier $\nu \in \mathbb{R}$ for the equality constraint $\boldsymbol{\theta}^T \mathbf{x} = b$, we obtain the KKT conditions

$$\mathbf{x} \geqslant 0, \quad \mathbf{x} \leqslant 1, \quad \boldsymbol{\theta}^T \mathbf{x} = b, \quad \boldsymbol{\lambda} \geqslant 0, \quad \boldsymbol{\mu} \geqslant 0,$$

$$\lambda_i x_i = 0, \ i = 1, \ldots, d,$$

$$\mu_i (x_i - 1) = 0, \ i = 1, \ldots, d, \tag{6.28}$$

$$(x_i - u_i) - \lambda_i + \mu_i + \nu\theta_i = 0, \ i = 1, \ldots, d, \tag{6.29}$$

where all the multipliers have been scaled by $1/2$ to normalize the coefficient of the term $(x_i - u_i)$. The solutions of $\mathbf{x}$, $\boldsymbol{\lambda}$, $\boldsymbol{\mu}$, and $\boldsymbol{\nu}$ can be found by directly solving the above equations. We can see that $\boldsymbol{\mu}$ acts as a slack variable in (6.29), so it can be eliminated by plugging it into (6.28), leaving

$$\mathbf{x} \geqslant 0, \quad \mathbf{x} \leqslant 1, \quad \boldsymbol{\theta}^T \mathbf{x} = b, \quad \boldsymbol{\lambda} \geqslant 0,$$

$$\lambda_i x_i = 0, \ i = 1, \ldots, d, \tag{6.30}$$

$$(\lambda_i - x_i + u_i - \nu\theta_i)(x_i - 1) = 0, \ i = 1, \ldots, d, \tag{6.31}$$

$$\lambda_i - x_i + u_i - \nu\theta_i \geqslant 0, \ i = 1, \ldots, d, \tag{6.32}$$

For (6.30) to hold, either $x_i = 0$ or $\lambda_i = 0$. From (6.32) we have $x_i \leqslant \lambda_i + u_i - \nu\theta_i$. When $u_i - \nu\theta_i \leqslant 0$, $x_i = 0$ must hold because of the constraint $x_i \geqslant 0$. If $x_i = 0$, we have

Algorithm 6.3 Solving optimization problem (6.27)

```
Input: vector $\mathbf{u} \in \mathbb{R}^d$, vector $\boldsymbol{\theta} \in \{-1,1\}^d$, and scaler $b \geqslant 0$.
Output: solution of optimization problem (6.27) in terms of $\nu^*$.
1: $T \leftarrow \{u_i - 1 \mid u_i > 1\} \cup \{-u_i \mid u_i < 0\}$, $\delta \leftarrow \boldsymbol{\theta}^T\mathbf{u} - b$, and $\zeta = d$.
2: if $T \neq \emptyset$ then
3:     Sort elements in $T = \{t_i\}$ such that $t_i \geqslant t_{i+1}$.
4:     for $i = 1$ to $|T|$ do
5:         if $\delta/\zeta > t_i$ then
6:             Exit to line 12.
7:         else
8:             $\delta \leftarrow \delta - t_i$ and $\zeta \leftarrow \zeta - 1$.
9:         end if
10:    end for
11: end if
12: return $\nu^* \leftarrow \delta/\zeta$ is optimal solution.
```

$0 \leqslant \lambda_i = x_i - (u_i - \nu\theta_i)$ by (6.31), which implies $u_i - \nu\theta_i \leqslant x_i = 0$. Hence, $x_i = 0$ if and only if $u_i - \nu\theta_i \leqslant 0$.

For (6.31) to be satisfied, either $x_i = 1$ or $x_i = \lambda_i + u_i - \nu\theta_i$ must hold. Since $x_i \leqslant 1$ and $\lambda_i \geqslant 0$, if $u_i - \nu\theta_i \geqslant 1$, we must have $x_i = 1$. If $x_i = 1$, we have $\lambda_i = 0$ by (6.30) and $u_i - \nu\theta_i \geqslant x_i = 1$ by (6.32). So, $x_i = 1$ if and only if $u_i - \nu\theta_i \geqslant 1$.

When $0 < u_i - \nu\theta_i < 1$, from (6.30) and (6.31), we have $\lambda_i = 0$ and $x_i = u_i - \nu\theta_i$. Therefore, the solution to the optimization problem (6.27) is

$$x_i^* = \begin{cases} 1 & \text{if } u_i - \nu\theta_i \geqslant 1 \\ 0 & \text{if } u_i - \nu\theta_i \leqslant 0 \\ u_i - \nu\theta_i & \text{otherwise.} \end{cases} \tag{6.33}$$

The above solution can be simply written as

$$\mathbf{x}^* = \Pi_{[0,1]^d}\left(\mathbf{u} - \nu^*\boldsymbol{\theta}\right), \tag{6.34}$$

where ν^* is chosen such that $\boldsymbol{\theta}\mathbf{x}^* = b$.

We specifically design an efficient algorithm to compute the ν^*, as described in Algorithm 6.3 , which makes use of the following special relationship among the inputs $\boldsymbol{\theta}$, $\mathbf{u}$, and b. First, we use the indicator vector $\boldsymbol{\theta}_{\mathcal{V}}$ of the cutting set $\mathcal{V}$ on $\mathbf{z} = \Pi_{[0,1]^d}(\mathbf{u})$ as the input vector

Algorithm 6.4 Efficient check polytope projection algorithm

```
Input: vector $\mathbf{u} \in \mathbb{R}^d$.
Output: the projection $\Pi_{\mathcal{P}_d}(\mathbf{u}) \in \mathcal{P}_d$.
1: Run Algorithm 6.2 with input $\Pi_{[0,1]^d}(\mathbf{u})$.
2: if cutting set $\mathcal{V}$ is found then
3:     Run Algorithm 6.3 with input parameters $\mathbf{u}$, $\boldsymbol{\theta}_{\mathcal{V}}$, and $|\mathcal{V}| - 1$, then get output $\nu^*$.
4:     return $\Pi_{[0,1]^d}(\mathbf{u} - \nu^* \boldsymbol{\theta}_{\mathcal{V}})$.
5: else
6:     return $\Pi_{[0,1]^d}(\mathbf{u})$.
7: end if
```

$\boldsymbol{\theta}$. Then from Proposition 6.4, we know the input $b = |\mathcal{V}| - 1 < \boldsymbol{\theta}_{\mathcal{V}}^T \mathbf{z} \leqslant \boldsymbol{\theta}_{\mathcal{V}}^T \mathbf{u}$; moreover, $\theta_i = 1$ for any $u_i > 1$ and $\theta_i = -1$ for any $u_i < 0$. Finally, if we let $v_1 = \min_{i \in \mathcal{V}} \{u_i\}$ and $v_2 = \min_{i \in \mathcal{V}^c} \{u_i\}$, then from Corollary 6.5, $v_1 > v_2$ and $\nu^* \leqslant (v_1 - v_2)/2$. Based on these conditions, the optimal ν^* can always be found by Algorithm 6.3 .

By combining Algorithms 6.2 and 6.3 , the projection of any given point to the check polytope can be done as described in Algorithm 6.4 , which is used in (6.14) of Algorithm 6.1 . Comparing Algorithm 6.4 with the two-slice projection algorithm proposed in [6] whose efficiency has already been improved over its precursor in [5], it is obvious that the proposed projection algorithm is simpler and more efficient. In the two-slice algorithm, in order to check whether a given point is inside the check polytope, all the coordinates of the point have to be sorted in descending order. In contrast, the proposed algorithm can find the cutting facet on which the projection point is located, by simply visiting all coordinates only once. As for the projection of a point outside the check polytope, although the two-slice algorithm uses a similar format as in (6.34), its more complicated underlying optimization problem requires the ordering of a set of size $2d + 2$ and potentially needs to check all elements in the set before finding the optimal ν^* [6, Algorithm. 2]. However, in the proposed Algorithm 6.3 , only the coordinates whose values are outside the $[0, 1]$ range need to be taken into account, and the required computation is also much simpler then the two-slice algorithm. As we will show in Section **??**, with the same ADMM parameters, the iterative ADMM-based LP decoder with the proposed algorithm runs at least 3 times faster than the one with the improved two-slice projection algorithm in [6].

6.3 Improving the Performance of ADMM-based LP Decoding

The ADMM-based LP decoding described in the previous section is more efficient than the LP decoding with general-purpose LP solvers, but there are still several improvements that can be made to further speed up the decoding. In this section, we will discuss some of these methods that can significantly reduce the number of iterations required to decode a codeword.

6.3.1 Early Termination

In this subsection, we will show that the ADMM-based LP decoding can declare a decoding success even before the stopping criteria (6.18) and (6.19) are satisfied.

Definition 6.7 *The* fundamental polytope $\mathcal{P}(\mathbf{H})$ *of the parity-check matrix* $\mathbf{H}$ *is defined to be the set*

$$\mathcal{P}(\mathbf{H}) = \bigcap_{j\in\mathcal{J}} \left\{\mathbf{x} \mid \mathbf{T}_j\mathbf{x} \in \mathcal{P}_{d_j}\right\}. \tag{6.35}$$

The solution of LP problem (6.1) corresponds to a vertex of the fundamental polytope that minimizes the objective function. Codewords correspond to the vertices with all integral coordinates, and pseudocodewords correspond to the vertices that have non-integral coordinates.

Theorem 6.8 *For a codeword* $\mathbf{c}$ *of a binary linear block code defined by parity-check matrix* $\mathbf{H}$*, any pseudocodeword* $\mathbf{p}$ *that corresponds to a nonintegral vertex of* $\mathcal{P}(\mathbf{H})$ *must have at least one coordinate* i *satisfying* $|p_i - c_i| > \frac{1}{2}$.

Proof: For a binary linear block code, we can assume without loss of generality that $\mathbf{c}$ is the all-zero codeword. Then, we need to show that all pseudocodewords must have at least one coordinate with value greater then $\frac{1}{2}$.

It is obvious that every vertex of the fundamental polytope must sit on a facet that does not contain the all-zero vertex. Therefore, from (2.12), any vertex $\mathbf{p}$ of the fundamental polytope must satisfy the following equation

$$\sum_{i\in\mathcal{V}}(1-p_i) + \sum_{i\in\mathcal{N}_j\setminus\mathcal{V}} p_i = 1, \tag{6.36}$$

for some check j and $\mathcal{V} \subseteq \mathcal{N}_j$ with $|\mathcal{V}| > 1$ and $|\mathcal{V}|$ odd. It can be seen that, if $|\mathcal{V}| > 1$, (6.36) does not hold for any non-zero vector whose coordinates all have values less than $1/2$. This means that any non-zero vertex of the fundamental polytope must have at least one coordinate with value greater than $1/2$. This proves the theorem. ■

From Theorem 6.8, we can see that the precision of the solution to the LP problem (6.1) can be coarse. Define hard decision function $\Gamma_t(\mathbf{x})$ such that

$$\Gamma_a(x_i) = \begin{cases} 1 & \text{if } x_i > t \\ 0 & \text{if } x_i < t. \end{cases} \tag{6.37}$$

where t is the hard decision threshold. Then, we have the following corollary.

Corollary 6.9 *Let $\mathbf{x}^k$ be the variable values after k ADMM iterations, and let $\mathbf{x}^*$ be the solution to the LP problem* (6.2)*. If it can be guaranteed that $\max_i |x_i^k - x_i^*| < \frac{1}{4}$, and if $\max_i \{\frac{1}{2} - |x_i^k - \frac{1}{2}|\} < \frac{1}{4}$ and $\Gamma_{\frac{1}{2}}(\mathbf{x}^k)$ is a valid codeword, then $\mathbf{x}^* = \Gamma_{\frac{1}{2}}(\mathbf{x}^k)$.*

Although it is not easy to guarantee that $\max_i |x_i^k - x_i^*| < \frac{1}{4}$ after k iterations, the following proposition from optimization theory shows that $\mathbf{x}^k$ converges to $\mathbf{x}^*$.

Proposition 6.10 ([2,3]) *For the optimization problem* (6.2)*, the $\mathbf{x}^k$, $\mathbf{y}^k$, and $\mathbf{z}^k$ in the ADMM iterates all converge to their unique optimal values, and the following inequality holds*

$$V^k - V^{k+1} \geqslant \rho \sum_{j\in\mathcal{J}} \left(\|\mathbf{T}_j \mathbf{x}^{k+1} - \mathbf{z}_j^{k+1}\|_2^2 + \|\mathbf{z}_j^{k+1} - \mathbf{z}_j^k\|_2^2 \right), \tag{6.38}$$

where V^k is the Lyapunov function

$$V^k = \rho \sum_{j\in\mathcal{J}} \left(\|\mathbf{y}_j^k - \mathbf{y}_j^*\|_2^2 + \|\mathbf{z}_j^k - \mathbf{z}_j^*\|_2^2 \right), \tag{6.39}$$

and $\mathbf{y}^*$ and $\mathbf{z}^*$ are the corresponding optimal points.

From (6.38), we know that the Lyapunov function V^k decreases in each iteration, and hence $\mathbf{y}$ and $\mathbf{z}$ are bounded. It follows (6.38) and (6.39) that

$$V^0 \geqslant \rho \sum_{k=0}^{\infty} \sum_{j\in\mathcal{J}} \left(\|\mathbf{T}_j \mathbf{x}^{k+1} - \mathbf{z}_j^{k+1}\|_2^2 + \|\mathbf{z}_j^{k+1} - \mathbf{z}_j^k\|_2^2 \right).$$

Because V^0 is finite, the primal residual $\mathbf{T}_j \mathbf{x}^k - \mathbf{z}_j^k \to \mathbf{0}$ and the dual residual $\mathbf{z}_j^k - \mathbf{z}_j^{k-1} \to \mathbf{0}$ as $k \to 0$ for all $j \in \mathcal{J}$. This shows that both of the stopping criteria (6.18) and (6.19) in Algorithm 6.1 will eventually hold.

When using early termination in ADMM-based LP decoding, the variable vector $\mathbf{x}$ goes through a hard decision $\Gamma_{\frac{1}{2}}$ at the end of each iteration. If the binary vector $\Gamma_{\frac{1}{2}}(\mathbf{x})$ is a valid codeword, the decoder declares success and outputs $\Gamma_{\frac{1}{2}}(\mathbf{x})$ as the decoded codeword. With the initialization of $\mathbf{x}$ described in (6.46), we have not observed, in millions of received code frames, any cases where the variables satisfied the condition for early termination during the ADMM iterations but later converged to another codeword or pseudocodeword. If the accuracy of pseudocodeword solution is not of interest, the two stopping criteria in Algorithm 6.1 can be replaced by a limit on the maximum number of iterations when early termination is used.

6.3.2 Varying Penalty Parameter and Over-Relaxation

There are many variations on the standard ADMM algorithm proposed in the literature, and some of them can provide superior convergence in practice. In this subsection, we introduce two of these extensions, which have been rigorously studied by optimization theorist and the convergence results are still valid [2].

The first variation is to vary penalty parameter ρ in each iteration. Varying the penalty parameter makes the convergence less dependent on the initial choice of the penalty parameter, and hence improve the performance in practice. One simple and practical scheme [13] involves setting

$$\rho^{k+1} = \begin{cases} \tau^{\text{inc}}\rho^k & \text{if } \|\mathbf{T}_j\mathbf{x}^k - \mathbf{z}_j^k\|_2 > \mu\rho^k\|\mathbf{z}_j^k - \mathbf{z}_j^{k-1}\|_2 \\ \rho^k/\tau^{\text{dec}} & \text{if } \rho^k\|\mathbf{z}_j^k - \mathbf{z}_j^{k-1}\|_2 > \mu\|\mathbf{T}_j\mathbf{x}^k - \mathbf{z}_j^k\|_2 \\ \rho^k & \text{otherwise,} \end{cases} \tag{6.40}$$

where the typical choices of the parameters might be $\mu = 10$ and $\tau^{\text{inc}} = \tau^{\text{dec}} = 2$. The idea behind this method is to try to keep the primal and dual residual norms close to one anther as they both converge to zero [2]. Note that, when the penalty parameter is changed during the ADMM iteration, the scaled dual variable $\mathbf{y}^k = \boldsymbol{\lambda}^k/\rho$ must also be adjusted accordingly.

The other variation is called *over-relaxation*. It replaces the quantity $\mathbf{T}_j\mathbf{x}^k$ in the $\mathbf{z}$- and $\mathbf{y}$-updates in (6.9) and (6.10) with

$$\alpha^k\mathbf{T}_j\mathbf{x}^k + (1-\alpha^k)\mathbf{z}_j^k,$$

where $1 \leqslant \alpha^k \leqslant 2$ is a *relaxation parameter*. Analysis and experiments suggest that the convergence of ADMM can be improved with properly chosen relaxation parameter α^k. To include the over-relaxation into the ADMM-base LP decoding, one only needs to replace (6.13) with

$$\mathbf{w} \leftarrow \alpha^k\mathbf{T}_j\mathbf{x}^k + (1-\alpha^k)\mathbf{z}_j^k + \mathbf{y}_j. \tag{6.41}$$

In our implementation of over-relaxation, a fixed α is used throughout all iterations, and in the initialization step, all elements of $\mathbf{z}_j$ are set to $\frac{1}{2}$ for all $j \in \mathcal{J}$.

6.3.3 Symmetric Implementation of ADMM-based LP Decoding

In conventional LP decoding, the feasible space of variables is the fundamental polytope (6.35) which is inside the hypercube $[0,1]^n$. Since the ADMM-based LP decoding can be implemented as a message-passing technique and the early termination applies a hard decision on all variables at the end of each iteration, it would be desirable to have the range of variables be symmetric about zero, such that $x_i \in [-a, a]$ for some $a > 0$. Such a symmetric implementation could also simplify some computations during the ADMM iterations, as will be shown in the following part of this subsection.

Define bijection $\Phi_a : [0,1]^n \to [-a,a]^n$ such that

$$\Phi_a(\mathbf{x}) = a(2\mathbf{x} - \mathbf{1}), \quad \mathbf{x} \in [0,1]^n \tag{6.42}$$

and

$$\Phi_a^{-1}(\mathbf{y}) = \frac{1}{2}\left(\frac{\mathbf{y}}{a} + \mathbf{1}\right), \quad \mathbf{y} \in [-a,a]^n \tag{6.43}$$

where $\mathbf{1}$ is the all-one vector.

Theorem 6.11 *The vector* $\mathbf{x}^* \in [0,1]^n$ *is the solution to the optimization problem* (6.2) *if and only if* $\Phi_a(\mathbf{x}^*) \in [-a,a]^n$ *is the solution to the following LP problem*

$$\begin{aligned} \textit{minimize} \quad & \boldsymbol{\gamma}^T \mathbf{x} \\ \textit{subject to} \quad & \mathbf{T}_j \mathbf{x} = \mathbf{z}_j,\ \mathbf{z}_j \in \Phi_a(\mathcal{P}_{d_j}),\ \forall j \in \mathcal{J}. \end{aligned} \tag{6.44}$$

Proof: Since Φ_a is a bijection, for all $\mathbf{z} \in \Phi_a(\mathcal{P}(\mathbf{H}))$, we have $\Phi_a^{-1}(\mathbf{z}) \in \mathcal{P}(\mathbf{H})$. To show sufficiency, suppose $\mathbf{x}'$ is the solution to (6.44) such that $\boldsymbol{\gamma}^T\mathbf{x}' < \boldsymbol{\gamma}^T\Phi_a(\mathbf{x}^*)$. Then, by simple algebra, we have $\boldsymbol{\gamma}^T\Phi_a^{-1}(\mathbf{x}') < \boldsymbol{\gamma}^T\mathbf{x}^*$. This contradicts the assumption that $\mathbf{x}^*$ is the solution to (6.1), because $\Phi_a^{-1}(\mathbf{x}') \in \mathcal{P}(\mathbf{H})$. Therefore, $\Phi_a(\mathbf{x}^*)$ is the solution to (6.44). The necessity can be proved in a similar way. ■

Comparing LP problems (6.2) and (6.44), we can see that the only difference is the polytope constraint. Hence, the ADMM-based LP decoding procedure described in Algorithm 6.1 remains the same, while only the check polytope projection algorithm in (6.13) needs to be adjusted.

It is easy to verify that the projected check polytope $\Phi_a(\mathcal{P}_{d_j})$ can be described by the following box constraints and linear inequalities

$$-a \leqslant u_i \leqslant a,\ \forall i \in [d]$$

Algorithm 6.5 Cut search algorithm for $\Phi_a(\mathcal{P}_d)$

```
Input: vector $\mathbf{u} \in [-a, a]^d$.
Output: indicator vector $\boldsymbol{\theta} \in \{-1, 1\}^d$ of the cut (if exists)
1: $\theta_i \leftarrow \begin{cases} 1 & \text{if } u_i > 0 \\ -1 & \text{otherwise} \end{cases}$
2: if the cardinality of set $\{i : \theta_i > 0\}$ is even then
3:     $i^* \leftarrow \underset{i}{\operatorname{argmin}} |u_i|$.
4:     $\theta_{i^*} \leftarrow -\theta_{i^*}$.
5: end if
6: if $\boldsymbol{\theta}^T \mathbf{u} > a(d-2)$ then
7:     return $\boldsymbol{\theta}$, and $\mathbf{u} \notin \Phi_a(\mathcal{P}_d)$.
8: else
9:     return No cut found, and $\mathbf{u} \in \Phi_a(\mathcal{P}_d)$.
10: end if
```

and

$$\sum_{i\in\mathcal{V}} u_i - \sum_{i\in\mathcal{V}^c} u_i \leqslant a(d-2),\ \forall \mathcal{V} \subseteq [d] \text{ with } |\mathcal{V}| \text{ odd.}$$

Therefore, the cut search algorithm needs to be modified accordingly, as described in Algorithm 6.5 . As for the computation of ν^* when a cut is found, Algorithm 6.3 remains almost the same except that the set T defined in line 1 becomes

$$T \leftarrow \{u_i - a \mid u_i > a\} \cup \{a - u_i \mid u_i < -a\}. \tag{6.45}$$

The check polytope projection algorithm for $\Phi_a(\mathcal{P}_d)$ also needs to be modified, as shown in Algorithm 6.6 . Although, theoretically, the initial values of $\mathbf{x}$, $\mathbf{y}$, and $\mathbf{z}$ do not affect the optimal solution found by ADMM, to have better convergence performance in practice, we initialize $\mathbf{y}_j$ and $\mathbf{z}_j$ to be all-zero vectors for all $j \in \mathcal{J}$, and initialize $\mathbf{x}$ as follows

$$x_i = \begin{cases} -a & \text{if } \gamma_i \geqslant 0 \\ a & \text{if } \gamma_i < 0. \end{cases} \tag{6.46}$$

6.3.4 Improving Error-Rate Performance of LP Decoding

From the ML certificate of LP decoding, we know that if the solution to the optimization problem (6.2) is integral, it is an ML codeword. Therefore, it is desirable to modify the

Algorithm 6.6 Efficient check polytope projection algorithm for $\Phi_a(\mathcal{P}_d)$

```
Input: vector $\mathbf{u} \in \mathbb{R}^d$.
Output: the projection $\Pi_{\Phi_a(\mathcal{P}_d)}(\mathbf{u}) \in \Phi_a(\mathcal{P}_d)$.
1: Run Algorithm 6.5 with input $\Pi_{[-a,a]^d}(\mathbf{u})$.
2: if cutting set $\mathcal{V}$ is found then
3:     Run modified Algorithm 6.3 with input parameters $\mathbf{u}$, $\boldsymbol{\theta}_{\mathcal{V}}$, and $a(d-2)$, then get output $\nu^*$.
4:     return $\Pi_{[-a,a]^d}(\mathbf{u} - \nu^* \boldsymbol{\theta}_{\mathcal{V}})$.
5: else
6:     return $\Pi_{[-a,a]^d}(\mathbf{u})$.
7: end if
```

objective function to favor the integral solution while keeping the optimal solution unchanged. In [8], it was proposed that adding penalty terms with certain properties to the objective function can improve the error rate performance, although the extra penalty terms change the original LP problem into a nonlinear and non-convex optimization problem for which the ADMM is not guaranteed to find the global optimal. We have observed from our simulation results that, although the method proposed in [8] improves the error-rate performance in waterfall region, it also causes high error floors for some codes and has inferior performance in the high signal-to-noise ratio (SNR) region to that of LP decoding with the original objective function.

With the proposed symmetric implementation, the penalty terms in [8] can be easily extended into more general terms. We extend the ℓ_2 *penalty function* $\alpha\|\mathbf{x} - 0.5\|_2^2$ proposed in [8], which was found to have good performance, into a more general quadratic penalty term $-\mathbf{x}^T\mathbf{A}\mathbf{x}$, where $\mathbf{A}$ is a diagonal matrix with α_i, $1 \leqslant i \leqslant n$, as its diagonal elements. Hence, the LP problem (6.44) with proposed penalty term can be written as

$$\begin{aligned} \text{minimize} \quad & \boldsymbol{\gamma}^T\mathbf{x} - \mathbf{x}^T\mathbf{A}\mathbf{x} \\ \text{subject to} \quad & \mathbf{T}_j\mathbf{x} = \mathbf{z}_j,\ \mathbf{z}_j \in \Phi_a(\mathcal{P}_{d_j}),\ \forall j \in \mathcal{J}. \end{aligned} \tag{6.47}$$

Our simulation results show that having different α_i for each variable node can significantly improve the error-rate performance, especially for irregular LDPC codes.

Theorem 6.12 *A codeword* $\mathbf{c}^* \in \{0,1\}^n$ *is the solution to LP problem* (6.2) *if and only if* $\Phi_a(\mathbf{c}^*) \in \{-a,a\}^n$ *is the solution to optimization problem* (6.47).

Proof: To prove necessity, assume codeword $\mathbf{c}^*$ is the solution to (6.2). Then $\boldsymbol{\gamma}^T\mathbf{c}^* \leqslant \boldsymbol{\gamma}^T\mathbf{u}$

for all $\mathbf{u} \in \mathcal{P}(\mathbf{H})$. Let $\mathbf{v}^* = \boldsymbol{\gamma}^T \Phi_a(\mathbf{c}^*)$. From Theorem 6.11, we know $\boldsymbol{\gamma}^T \mathbf{v}^* \leqslant \boldsymbol{\gamma}^T \mathbf{v}$ for all $\mathbf{v} \in \Phi_a(\mathcal{P}(\mathbf{H}))$. Since $\mathbf{v}^* \in \{-a, a\}^n$ and $\mathbf{P}$ is a diagonal matrix, $-\mathbf{v}^{*T}\mathbf{A}\mathbf{v}^* \leqslant -\mathbf{v}^T\mathbf{A}\mathbf{v}$ for all $\mathbf{v} \in [-a, a]^n$, and hence $\boldsymbol{\gamma}^T \mathbf{v}^* - \mathbf{v}^{*T}\mathbf{A}\mathbf{v}^* \leqslant \boldsymbol{\gamma}^T \mathbf{v} - \mathbf{v}^T\mathbf{A}\mathbf{v}$ for all $\mathbf{v} \in \Phi_a(\mathcal{P}(\mathbf{H}))$. This proves the condition is necessary, and sufficiency can be similarly proved. ■

Theorem 6.12 shows that if the solution to (6.47) found by ADMM corresponds to a globally optimal codeword, which can be verified by running ADMM on LP problem (6.44) with the solution as initial $\mathbf{x}$, then the codeword is ML.

To apply ADMM to the optimization problem (6.47), only the $\mathbf{x}$-update (6.17) in Algorithm 6.1 needs to be modified, as follows:

$$x_i \leftarrow \frac{1}{d_i - \beta_i}\left(\sum_{j' \in \mathcal{N}_i} L_{j' \to i} - \frac{\gamma_i}{\rho}\right), \tag{6.48}$$

where $\beta_i = 2\alpha_i/\rho$. Note that, since the optimization problem (6.47) with quadratic penalty term is non-convex, the solution found by ADMM is a local optimal but might not be the global optimal. Hence, the choice of initial values of $\mathbf{x}$ affects the ADMM solution of the penalized non-convex optimization problem (6.47). We propose that the penalty term should not be added to the objective function until after a certain number of iterations or until the ADMM converges to a pseudocodeword, and our simulation results show that such a decoding scheme provides a substantial improvement over the penalized LP decoders whose penalty term is added before decoding starts. Such improvement is especially evident in high SNR region where adding a penalty term before decoding starts could have an error floor. Although the authors of [8] also suggested a two-stage decoder which concatenates an LP decoder with the original objective function followed by the penalized decoder, their purpose in using such a two-stage decoder is merely to provide for theoretical reasons a scheme whose decoding failures are independent of the transmitted codeword, without any intention to improve the error-rate performance of the penalized decoder. Therefore, the error-rate performance of the two-stage decoder was not discussed in [8].

Another way to further improve the error correction performance of LP decoding is to use the adaptive cut generation algorithm proposed in [7]. Although it can be directly applied after the ADMM-based LP decoding converges to a pseudocodeword, it would be interesting to find a way to include the cut generation scheme within the ADMM iterations. However, this is beyond the scope of this discussion.

6.4 Numerical Results

To demonstrate the improved performance offered by the ADMM-based LP decoding with the proposed projection algorithm and the implementation extensions introduced in the previous section, we compare their computational complexity and error-rate performance to that of other LDPC decoders such as the ADMM-based LP decoder with two-slice projection algorithm [6], adaptive LP (ALP) decoder [14], and floating-point box-plus SPA decoder [15]. We apply these decoders to several LDPC codes of various lengths, rates, and degree distributions, including a rate-0.77, (3,13)-regular LDPC codes of length 1057 [16]; a rate-$\frac{1}{3}$ irregular LDPC code of length 1920 [16]; a rate-0.82, variable-regular LDPC code of length 4095 [16]; and a rate-$\frac{1}{2}$ irregular QC-LDPC code of length 576 from the IEEE 802.16e standard. The frame error rate (FER) curves are based on Monte Carlo simulations that generated at least 100 error frames for each point in the plots.

As we mentioned earlier in Chapter 2, unlike SPA decoding, LP decoding is insensitive to the scaling of input channel LLRs. This means that the input channel messages of LP decoding need not to be scaled according to the SNR to precisely represent the LLR of each symbol. Let $t_i = (-1)^{c_i}$ be the transmitted binary symbol of the ith code bit. For the additive white Gaussian noise channel (AWGNC), the corresponding received symbol is $r_i = t_i + n_i$, where n_i is independent and identically-distributed (i.i.d.) Gaussian noise of zero-mean and variance σ^2. Then, the LLR of ith received symbol can be computed as $2r_i/\sigma^2$. If we scale the LLRs of all received symbols by multiplying by $\sigma^2/2$, the coefficient of the objective function of LP decoding becomes $\gamma_i = r_i$. This means that, with LP decoding, the information of channel SNR is not required. Similar argument also holds for the binary symmetric channel (BSC).

We know that the ADMM-based LP decoder can be viewed as a message-passing decoder, so the maximum number of allowed iterations, $T_{\max}$, affects its error-rate performance. Fig. 6.1 compares the ADMM decoder for various $T_{\max}$ with the floating-point box-plus SPA decoding and the ALP decoding with general-purpose LP solver. We applied the early termination and over-relaxation methods discussed in the previous section to the ADMM-based LP decoding to reduce the computational complexity, and we fix the penalty parameter $\rho = 1$ and the relaxation parameter $\alpha = 1.9$ throughout all iterations. If the early termination condition is not satisfied, the ADMM keeps iterating until the maximum number of iterations is reached. The error-rate of ALP decoder can be viewed as the best performance that ADMM-based LP decoders can achieve by solving the standard LP problem (6.2). It can be seen that the error-rate performance of ADMM with $T_{\max} = 100$ is already very close to that of ALP decoding, and

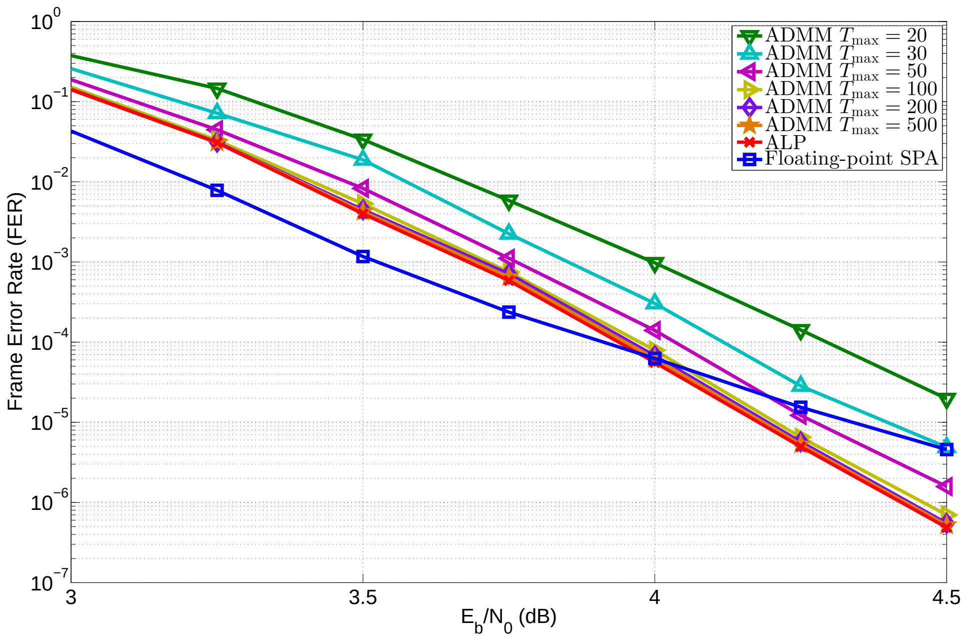
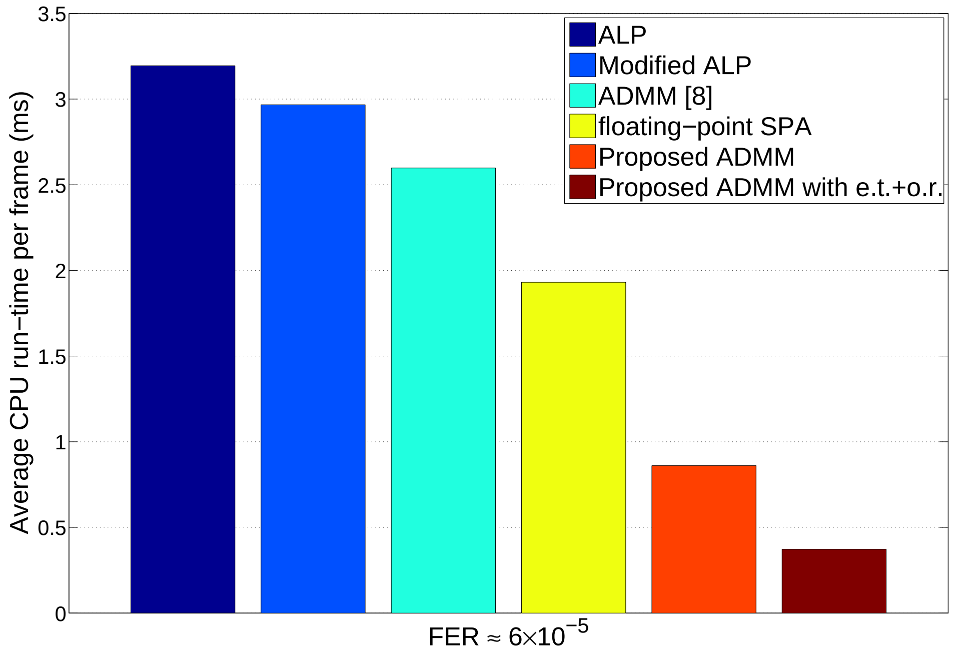


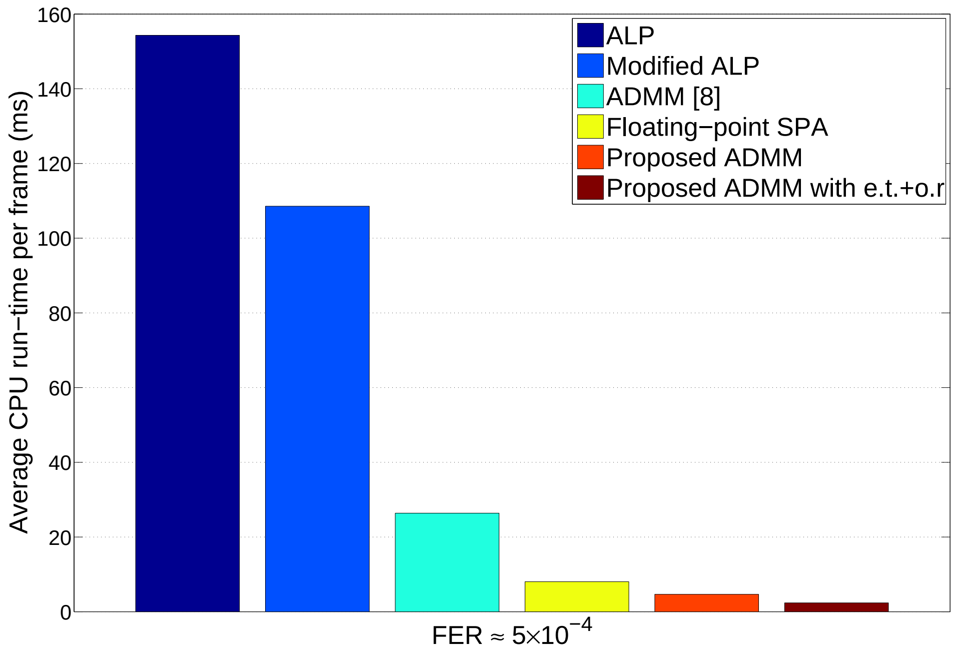
Figure 6.1: FER comparison of ADMM-base LP decoder with various maximum number of iterations for (1057,833) LDPC code on AWGNC.

the ADMM-based LP decoder with $T_{\max} = 200$ almost achieves the best error-rate performance of standard LP decoding. We can also see that all LP decoders have error-rate curves of steeper slope than the SPA decoder, and a similar phenomenon has been observed on all LDPC codes we tested [7]. We believe that LP-based decoders have better error correction performance than the SPA decoder in good channel conditions.

Fig. 6.2 presents an intuitive way of comparing the computational complexity of different decoding algorithms. It shows the average decoding time when we implement these algorithms using C++ code on a desktop PC. The SPA decoder is implemented in software with messages represented as double-precision floating-point numbers, and the pair-wise box-plus SPA is computed in the way described in [17]. The SPA decoder iterations stop as soon as a codeword is found, or when the maximum allowable number of iterations $T_{\max} = 200$ have been attempted. To avoid possible implementation variation, the C++ code of the two-slice projection algorithm [6] is obtained from one author's website [18] and plugged into our C++ platform for ADMM-based LP decoders. For ADMM-based LP decoders, the simulation parameters are set to be the same, i.e., $T_{\max} = 200$, $\rho = 1$, and $\epsilon^{\text{prim}} = 10^{-4}$, where only (6.18) is used as the stopping criterion. The simulation time is averaged over one million frames for



(a) LDPC code of rate 0.77 and length 1057.



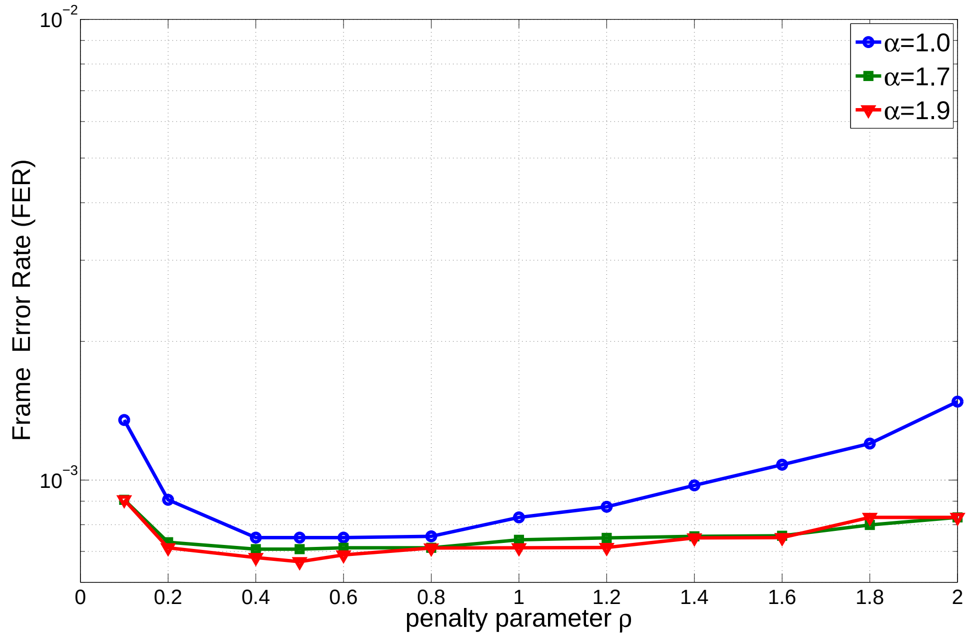
(b) LDPC code of rate 0.33 and length 1920.

Figure 6.2: Average simulation time for decoding one codeword.

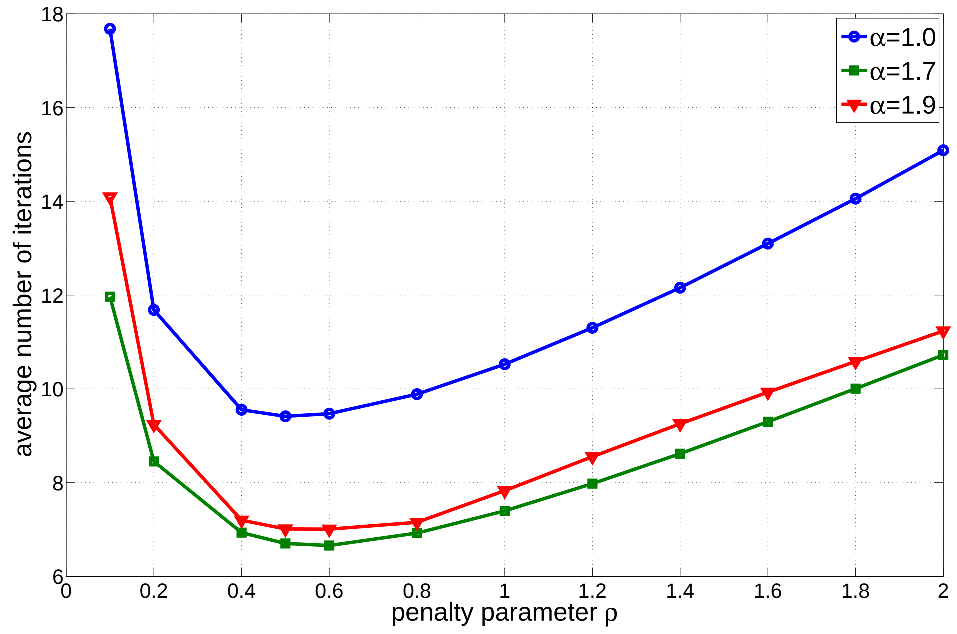
each decoder, and the channel condition is set to let all decoders achieve approximately the same frame-error rates as indicated in the figures. We can see that, compared to the two-slice projection algorithm in [6], the proposed projection algorithm significantly improves the efficiency of the ADMM-based LP decoder, especially for low-rate codes. The modified ALP (MALP) decoding algorithm [14], which is much faster than the standard LP decoder proposed by Feldman *et al.* in [**?**], uses the open-source GNU Linear Programming Kit (GLPK) [19] as its LP solver. From the simulation results, we can see that all ADMM-based LP decoders are faster than MALP, especially on low-rate codes, where the gain is more than 2 orders of magnitudes. This is because the general-purpose LP solver used in MALP does not take advantage of the sparsity in the constraints of the LP problem of decoding LDPC codes. The ADMM with early termination and over-relaxation is the most efficient LP decoding algorithm, further improving the decoding speed of the ADMM-based LP decoder by a factor of 3.

In our simulation data obtained from billions of decoded codewords, we never found any codeword satisfying the early termination condition discussed in Section 6.3.1 that was not an ML codeword, and we believe that early termination should always be applied to avoid unnecessary iterations. Hence, all ADMM-based LP decoders discussed in the remainder part of this section use early termination. We also found that the method of varying the penalty parameter discussed in Section 6.3.2 can not improve the decoding efficiency when early termination is used. However, the over-relaxation technique introduced in the same section can still substantially improve the convergence of ADMM. Hence, with the maximum number of iterations $T_{\max}$ fixed, the penalty parameter ρ and the over-relaxation parameter α can affect both the error-rate and the convergence performance of ADMM-based LP decoding, as shown by Figs. 6.3 and 6.4. Although not shown here, we found that the best choice of parameters α and ρ may vary as a function of channel SNR. It is not clear whether there is an analytical way to find the best ADMM channel parameters, and we chose the parameters empirically based on our simulation data.

Figs. 6.5 and 6.6 show the error-rate performance of ADMM-based LP decoding with penalty terms added to objective function. We compare the proposed general quadratic penalty with the so-called $\ell 1$ and $\ell 2$ penalty functions from [8]. The coefficients β_i in our general quadratic penalty are chosen according to the degree of each variable i such that $\beta_i = \alpha d_i$ where α is a constant parameter depending on codes and other ADMM parameters. The parameters of the penalty functions plotted here are chosen from a range of values tested in our simulations based on the error-rate performance. We would like to point out that, due to the limited range

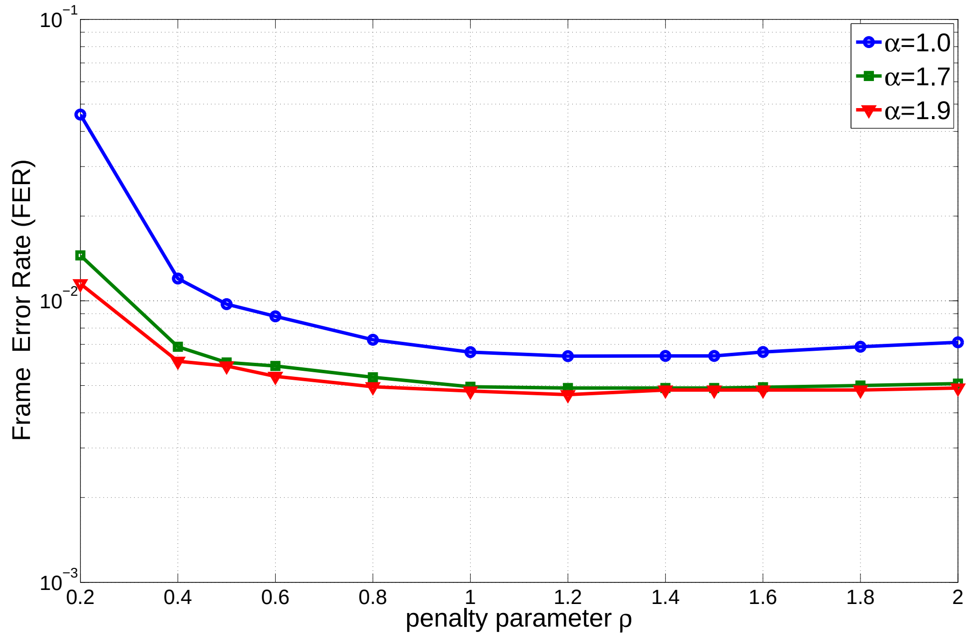


(a) FER versus penalty parameter ρ for various over-relaxation parameter α.

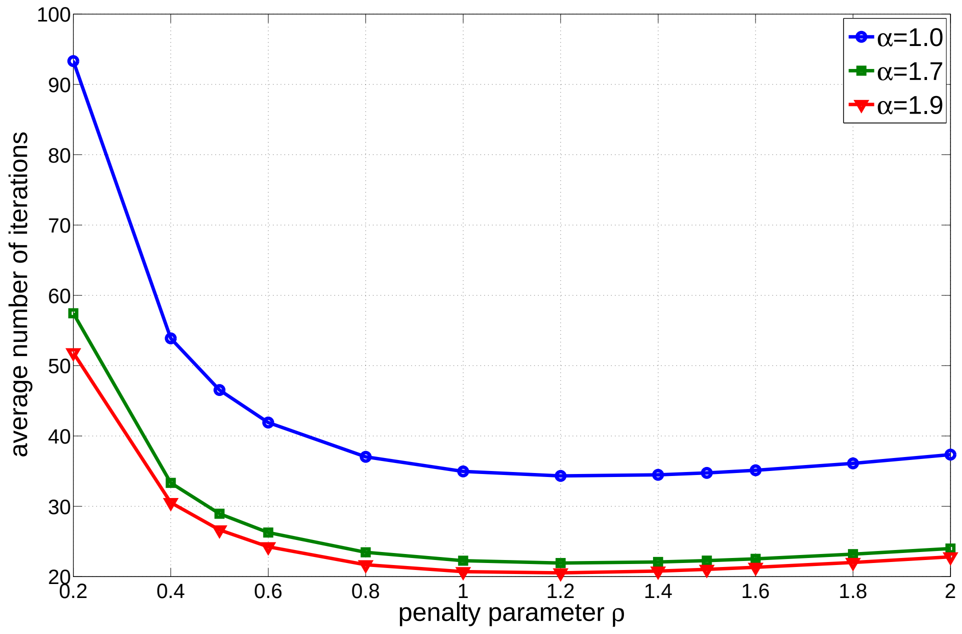


(b) Average number of iterations versus penalty parameter ρ for various over-relaxation parameter α.

Figure 6.3: ADMM parameter comparison for LDPC code of rate 0.77 and length 1057 on AWGNC, $T_{\max} = 200$ and $E_b/N_0 = 3.75$ dB.



(a) FER versus penalty parameter ρ for various over-relaxation parameter α.



(b) Average number of iterations versus penalty parameter ρ for various over-relaxation parameter α.

Figure 6.4: ADMM parameter comparison for LDPC code of rate 0.33 and length 1920 on AWGNC, $T_{\max} = 200$ and $E_b/N_0 = 2$ dB.

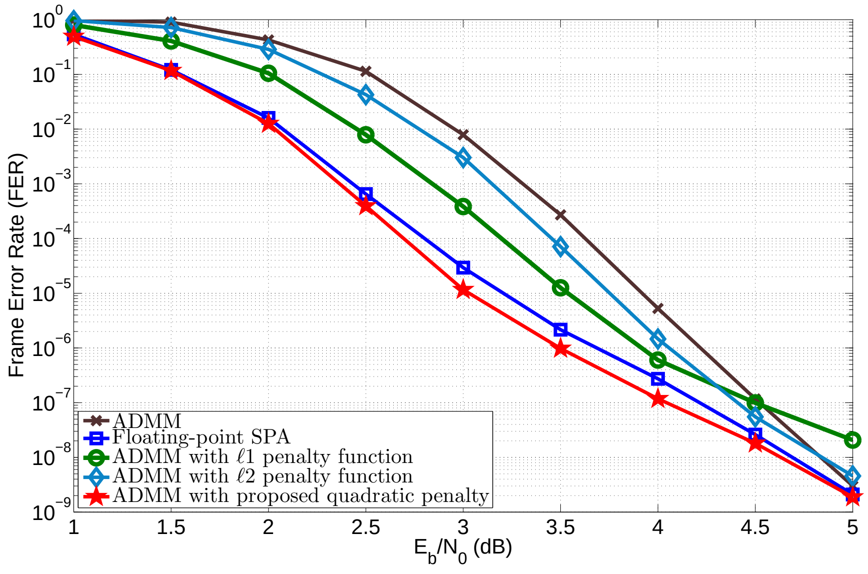


Figure 6.5: FER comparison of ADMM-base LP decoders with additional penalty terms in objective function for (576,288) LDPC code on AWGNC. $T_{\max}$ is set to 200 for all decoders. $\alpha = 0.12$ for the $\ell 1$ penalty, $\alpha = 0.1$ for the $\ell 2$ penalty, and $\beta_i = 0.78 d_i$ for the general $\ell 2$ penalty.

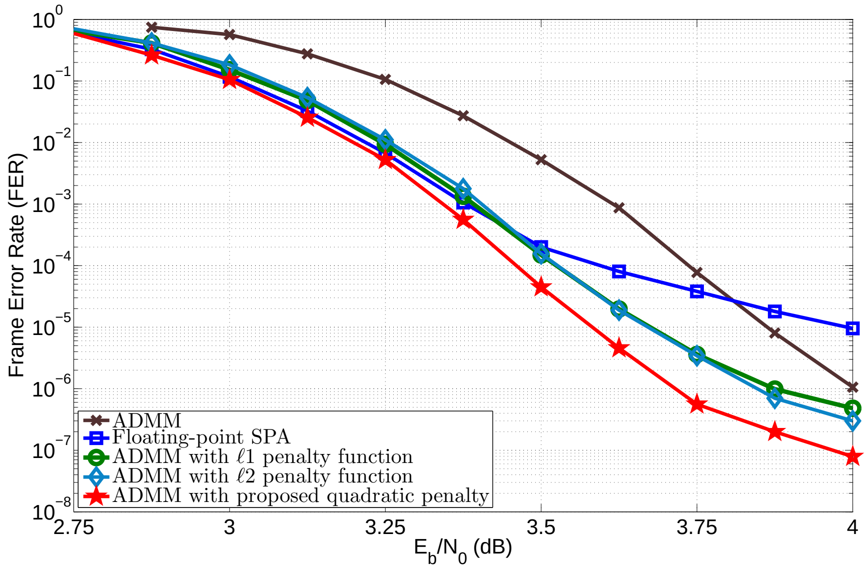


Figure 6.6: FER comparison of ADMM-base LP decoders with additional penalty terms in objective function for (4095,3358) LDPC code on AWGNC. $T_{\max}$ is set to 200 for all decoders. $\alpha = 0.08$ for the $\ell 1$ penalty, $\alpha = 0.13$ for $\ell 2$ penalty, and $\beta_i = 0.78 d_i$ for the general $\ell 2$ penalty.

of parameters we tested, there could possibly be other choices of penalty parameters that give better performance. It can be seen from these figures that, for the $\ell 1$ and $\ell 2$ penalty function, parameters giving strong penalty improve the error-rate performance in the waterfall region but introduce an error floor in the high SNR region, while weak penalties have a low error floor but inferior waterfall performance. With the proposed general quadratic penalty, which is added into the objective function after a certain number of iterations or after the ADMM has converged to a pseudocodeword, the error-rate performance in both waterfall and error-floor regions is substantially improved. To be specific, we used the original objective function for the first 100 iterations (or less if the decoder converges), and if no codeword is found, the general quadratic penalty is added for the remaining iterations by simply changing the $\mathbf{x}$-update from (6.17) to (6.48), without any complexity increase. Comparing Figs. 6.5 and 6.6, we can see that the error-rate improvement provided by the $\ell 1$ and $\ell 2$ penalty functions is smaller for the (576,288) code than for the (4095,3358) code. This is because the former code is irregular and its variable degrees vary from 2 to 7, but the $\ell 1$ or $\ell 2$ penalty remains the same for all variables.

For the irregular (576,288) IEEE 802.16 standard code of rate 0.5, our simulation data show that, when E_b/N_0 is 4 dB or larger, all decoding errors of the ADMM-base LP decoder with proposed quadratic penalty are valid codewords. By further examining these decoded codewords, we found that more than 99% of them are ML codewords. This means that the proposed decoder almost achieves ML performance for this code in the high SNR region. As shown in Fig. 6.6, the floating-point box-plus SPA decoder has an obvious error floor, which is caused by the limited number of iterations and could be improved by increasing $T_{\max}$. However, with the same $T_{\max}$ as the SPA, the ADMM-based LP decoder does not have such a high error floor and has steeper slope in its error-rate curve.

The superior error-correction performance of LP decoding makes it a promising candidate for many important applications that require extremely low error rates, such as data storage and high-speed digital communication. With the proposed efficient ADMM-based LP decoding algorithm, it becomes a potential candidate for implementation in practical applications. We conjecture that the inferior performance in the waterfall region of LP decoding with the original objective function could be caused by the chosen parity-check matrix which is often optimized for iterative belief-propagation decoding. Adding the penalty term can improve the error-rate performance, so optimizing parity-check matrices for LP decoding could also be a interesting future research topic.

6.5 Conclusion

The high complexity of general-purpose LP solvers, which are not optimized for solving LP problems with sparse constraints, makes LP decoding computationally more complex than widely used iterative message-passing decoding algorithms, especially for codes of large block size. In this paper, we propose an efficient message-passing algorithm to solve the LP decoding problem. It is based on the alternating direction method of multipliers (ADMM), a classic technique in convex optimization theory that is designed for parallel implementation. The computational complexity of ADMM-based LP decoding is largely determined by the method used to project a vector of real values to the parity polytope of a given parity check. We proposed a novel, efficient projection algorithm that can substantially improve the decoding speed of the ADMM-based LP decoder, and further improvement on decoding efficiency can be achieved by using early termination conditions, as well as optimization techniques such as over-relaxation that give faster convergence in practice. We also generalized the existing method of adding a quadratic penalty term, and our simulation results show that this generalized approach substantially improves the error-rate performance, especially for irregular LDPC codes.

Acknowledgment

This chapter is in part a reprint of the material in the paper: Xiaojie Zhang and Paul H. Siegel, “Efficient Iterative LP Decoding with Alternating Direction Method of Multipliers,” *To be submitted to IEEE Transactions on Information Theory*. The dissertation author was the primary investigator and author of this paper.

Bibliography

[1] D. Gabay and B. Mercier, “A dual algorithm for the solution of nonlinear variational problems via finite element approximations,”*Computers and Mathematics with Applications*, vol. 2, pp. 17–40, 1976.

[2] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, “Distributed optimization and statistical learning via the alternating direction method of multipliers,” *Foundations and Trends in Machine Learning*, vol. 3, no. 1, pp. 1–122, 2011.

[3] D. Bertsekas and J. Tsitsiklis, *Parallel and distributed computation: numerical methods*. Belmont, MA: Athena Scientific, 1997.

[4] P. O. Vontobel and R. Koetter, “Graph-cover decoding and finite-length analysis of message-passing iterative decoding of LDPC codes,” CoRR, arxiv.org/abs/cs.IT/0512078.

[5] S. Barman, X. Liu, S. Draper, and B. Recht, “Decomposition methods for large scale LP decoding,” *Proc. 46th Allerton Conf. Commun., Control, Computing*, Monticello, IL, Sep. 2011, pp. 253–260.

[6] S. Barman, X. Liu, S. Draper, and B. Recht, “Decomposition methods for large scale LP decoding,” arXiv:1204.0556 [cs.IT], 2012.

[7] X. Zhang and P. Siegel, “Adaptive cut generation algorithm for improved linear programming decoding of binary linear codes,” *IEEE Trans. Inform. Theory*, vol. 58, no. 10, pp. 6581–6594, Oct. 2012.

[8] X. Liu, S. Draper, and B. Recht, “Suppressing pseudocodewords by penalizing the objective of LP decoding,” in *Proc. IEEE Information Theory Workshop (ITW)*, Lausanne, Switzerland, Sep. 2012.

[9] M. Yannakakis, “Expressing combinatorial optimization problems by linear programs,” *J. Computer and System Sciences*, vol. 43, no. 3, pp. 441–466, 1991.

[10] A. Marshall, I. Olkin, and B. Arnold, *Inequalities: theory of majorization and its applications*. New York: Springer, 2010.

[11] M. H. Taghavi and P. H. Siegel, “Adaptive methods for linear programming decoding,” *IEEE Trans. Inform. Theory*, vol. 54, no. 12, pp. 5396–5410, Dec. 2008.

[12] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge, UK: Cambridge University Press, 2004.

[13] B. He, H. Yang, and S. Wang, “Alternating direction method with selfadaptive penalty parameters for monotone variational inequalities,” *Journal of Optimization Theory and Applications*, vol. 106, no. 2, pp. 337–356, 2000.

[14] M. H. Taghavi, A. Shokrollahi, and P. H. Siegel, “Efficient implementation of linear programming decoding,” *IEEE Trans. Inform. Theory*, vol. 57, no. 9, pp. 5960–5982, Sep. 2011.

[15] X. Zhang and P. Siegel, “Will the real error floor please stand up?” in *Proc. IEEE International Conference on Signal Processing and Communication (SPCOM)*, Bangalore, India, July 22–25, 2012, pp. 1–5.

[16] D. J. C. MacKay, *Encyclopedia of Sparse Graph Codes*. [Online]. Available: http://www.inference.phy.cam.ac.uk/mackay/codes/data.html

[17] X. Hu, E. Eleftheriou, D. Arnold, and A. Dholakia, ”Efficient implementations of the sum-product algorithm for decoding LDPC codes,” in *Proc. IEEE Global Telecommun. Conf.*, vol. 2, San Antonio, Nov. 2001, pp. 1036–1036E.

[18] C++ ADMM Decoder by X. Liu. [Online]. Available: https://sites.google.com/site/xishuoliu/codes

[19] GNU Linear Programming Kit, [Online]. Available: http://www.gnu.org/software/glpk

Chapter 7

Reliable Video Streaming over Distributed Cognitive Radio Networks

7.1 Introduction

Distributed wireless networks are characterized by scarce radio spectrum, unreliable propagation channels, strong interference, and user mobility. An important application over these networks is bandwidth-intense video streaming, e.g., transmitting surveillance video among mobile units in a battlefield. Such applications may have strict delay constraints, and may require a cross-layer mechanism that can effectively allocate its resources to each user in time. For the radio resource, we want the cognitive radio framework to be able to provide high data rates to accommodate video streams and to adaptively distribute radio resources according to users' requirements and channel conditions. For video streams, the video codec should have high scalability to adjust the coding rates according to required quality and available data rates. In this paper, we study the problem of dynamic, distributed, cross-layer resource allocation across multiple users transmitting real-time video over a multicarrier-based wireless cognitive radio network in a reliable way.

Radio resource allocation, i.e., subcarrier and power allocation, for multiple users has been widely studied in the literature [1–4]. By taking advantage of the time-varying property of wireless channels, knowledge of channel state information (CSI) can be used to exploit the multicarrier and multiuser diversity [1]. Multiuser resource allocation based on multicarrier modulation such as orthogonal frequency division multiplexing (OFDM) attracted extensive attention [2–4]. However, the results of this research are limited, since they are typically intended for systems where the available frequency spectrum has been given and multiple users synchronize to a central controller such as a base station which coordinates the cooperation between different users and optimally distributes the radio resources according to system requirements. Moreover, due to the large peak-to-average power ratio (PAPR) and its vulnerability to interference and jamming, OFDM is not as suitable as multicarrier direct sequence code-division multiple access (multicarrier DS-CDMA) [5] for applications such as military wireless communication systems. In challenging scenarios that have high demand for reliability of real-time communication, multi-hop relays might introduce unacceptable delay, so that users might have to transmit directly to their destination. Hence, a sophisticated distributed subcarrier and power allocation algorithm is needed. The conventional distributed power control algorithms [6,7] allocate power levels for multiple users on one single subcarrier. In [8–10], multiuser power control algorithms are proposed for frequency-selective channels, which can jointly allocate subcarriers and power levels. However, all these algorithms have target signal-to-interference-plus-noise ratio (SINR) or rate requirements which have been carefully designed based on the network conditions, that were assumed to be known a priori.

The cross-layer resource allocation algorithm proposed in this paper can adaptively control the joint subcarrier and power allocation according to the physical layer conditions and the video contents. To improve the overall performance for video streaming over a wireless network, joint source and channel coding has appeared in several recent works and has been shown to be an effective approach [11–14]. However, these papers typically focus on the downlink of an OFDM system, where a central controller is in charge of coordinating the cooperation between the users and optimally (or sub-optimally) distributing the system resources. Moreover, they assume that users have spectral utilization information via the aid of a base station, which is not realistic in scenarios such as distributed networks, where infrastructure is not available.

In this paper, we propose a reliable video transmission framework based on distributed cognitive radio ad-hoc networks. To deal with a severe near-far problem in such a framework, and in order to transmit video streams reliably, we propose a distributed cross-layer resource allocation algorithm that jointly allocates channel resources and video encoding rates to minimize the distortion of all transmitted video streams. The conventional rate-sum or utility-sum maximization algorithms sacrifice the performance of some users in order to increase the average rate or utility.

The rest of the paper is organized as follows. In Section II, we describe our wireless channel and network model, as well as the video distortion model, and formulate the cross-

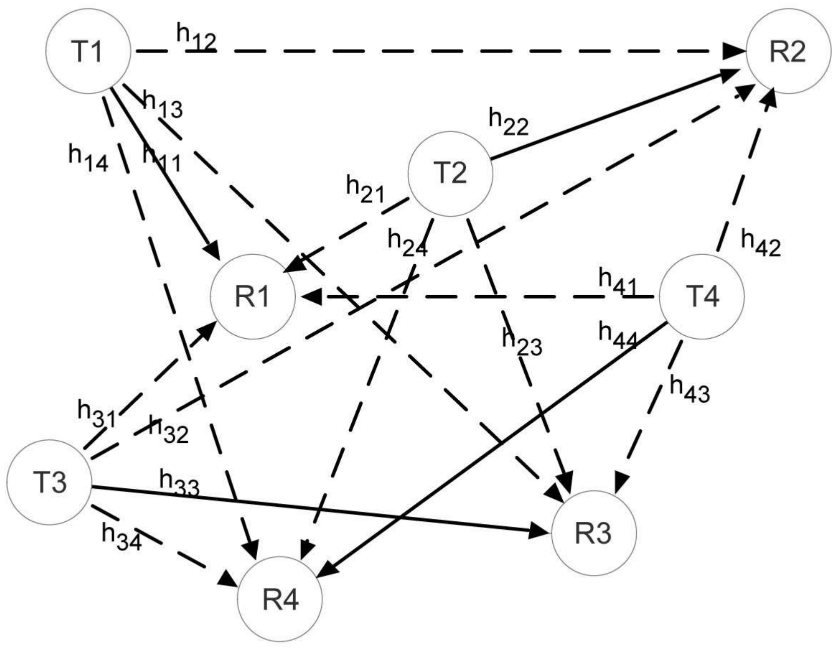


Figure 7.1: A cognitive radio network with 8 users (4 pairs).

layer resource allocation problem. The distributed cross-layer resource allocation algorithm is proposed in Section III. Section IV presents our simulation scenario and results. Finally, conclusions are provided in Section V.

7.2 System Description

7.2.1 Cognitive Radio Network Description

Let us consider a distributed ad-hoc wireless network with its users randomly deployed in a region. Each user is either a transmitter or a receiver. One transmitter and one receiver form a one-to-one pair and communicate with each other in a single-hop manner. Therefore, there may exist severe interference (near-far problem) among different user-pairs. Fig. 7.1 shows an example of such a system with 8 users (4 pairs).

The solid arrows in Fig. 7.1 are video signals to desired receivers, and the dashed arrows represent the interference to other receivers. The channel gain of user-pair i is denoted as h_{ii}, and the interference channel from transmitter i to receiver j ($i \neq j$) is denoted as h_{ij}. Each user-pair is only able to estimate the channel gain between its transmitter and receiver, i.e., user-pair i can only know h_{ii}.

For simplicity, we assume that the available frequency spectrum is divided into an integer number of subcarriers with equi-width frequency bands. We also assume that each subcarrier experiences flat fading, as in [5], by an appropriate choice of subcarrier width. In addition, band limited multicarrier waveforms are utilized in order to eliminate interference between subcarriers.

Transmission time is divided into equal-length time-frames, as shown in Fig. 7.2. We assume channels change sufficiently slowly such that the channel gain can be considered fixed during each time-frame, but channels may vary from frame to frame. At the beginning of each time-frame, every user performs spectrum sensing to detect the available subcarriers. Each receiver measures the total noise-plus-interference power it experiences, and feeds back the measured SINR to its own transmitter. It is assumed that this feedback is performed error free. All users cooperate with each other during spectrum sensing [15]. The sensing algorithm will be explained in Section 7.3. The proposed cross-layer resource allocation algorithm is performed during the power/rate control time slot, and this is followed by the actual video transmission.

The distributed network employs multicarrier DS-CDMA. Any user-pair can stream video content over any number of available subcarriers. Assume that at a given time instant t, there are K subcarriers and L user-pairs. On the k-th subcarrier, the received signal of user-pair l can be expressed as

$$\begin{aligned}\mathbf{y}_l^{(k)}(t) &= \sqrt{p_l^{(k)}(t)\, h_{ll}^{(k)}(t)}\, b_l^{(k)}(t)\, \mathbf{s}_l(t) \\ &+ \sum_{i \neq l, i=1}^{L} \sqrt{p_i^{(k)}(t+\Delta t_{il})\, h_{il}^{(k)}(t+\Delta t_{il})}\, b_i^{(k)}(t+\Delta t_{il})\, \mathbf{s}_i(t+\Delta t_{il}) + \mathbf{n}_l^{(k)}(t)\end{aligned} \tag{7.1}$$

where $b_l^{(k)}(t)$ and $p_l^{(k)}(t)$ are user-pair l's transmitted information symbol and transmission power on the k-th subcarrier at time t, respectively. The information symbol can be modulated with M-ary quadrature amplitude modulation (MQAM) as in this paper, but other modulation schemes are also applicable. The parameter $h_{il}^{(k)}(t)$ is the channel power gain between the transmitter of user-pair i and the receiver of user-pair l; $\mathbf{s}_l(t)$ is the spreading sequence for user-pair l; Δt_{il} is the time offset between user-pairs i and l; and $\mathbf{n}_l^{(k)}(t)$ denotes the noise vector for user-pair l on the k-th subcarrier which is assumed to be zero-mean additive white Gaussian noise (AWGN) with two-sided power spectral density of $\eta_0/2$ such that the noise power within each subcarrier bandwidth is σ^2. In (7.1), the received signal consists of three parts: desired signal, interference from other transmitters, and noise. We assume that each transmitter has a maximum transmit power:

$$\sum_{k=1}^{K} p_l^{(k)} \leqslant p_l^{\max} \tag{7.2}$$

7.2.2 Video Distortion Model

Operational Rate-Distortion (R-D) models [22, 23] describe the achievable quality of a video codec as a function of data rate. Studies have shown that partitioning video packets into different priority classes and adjusting rates for each class correspondingly can significantly improve overall received quality for given rates [24, 25]. In this paper, we will use a widely adopted parametric model for video distortion as a function of the output rate of the encoder [23]. For each user i, we associate a mean square error (MSE) distortion D_i when its video stream is encoded at rate r_i as

$$D_i\left(r_i\right) = D_{0i} + \frac{\theta_i}{r_i + R_{0i}} \tag{7.3}$$

where D_{0i}, R_{0i}, and θ_i are parameters which depend on the video sequence characteristics and the operational encoder-selected parameters. Note that the rate r_i can be zero, which means the user does not transmit at all in a time-frame. If that occurs, then a part of a new frame, or the whole frame, is missing at the receiver side, and the video decoder can take the corresponding part, or frame, from the last successfully decoded video frame to replace the missing new one. Hence, the distortion at zero rate is calculated at the transmitter side by comparing the new video frame with the one that the decoder should be able to rebuild with previously transmitted data. For example, if the video scene is rather static, i.e., the contents of successive video frames are almost the same, the decoder can rebuild the new frame with fairly low distortion from previous transmitted frames even if the whole new frame is missing (for example, if rate r_i is zero). However, if there are high-motion contents or a scene-cut in the new frame, the loss of even a part of the new frame will cause significant distortion when decoding.

Typically, video quality is measured by Peak Signal-to-Noise Ratio (PSNR), which is related to the MSE distortion as

$$PSNR = 10\log_{10}\frac{255^2}{MSE} = 10\log_{10}\frac{255^2}{D(r_i)} \tag{7.4}$$

Substituting (7.3) into (7.4), and letting $Q_i(r_i)$ denote the PSNR of user-pair i as a function of data rate r_i, the Quality-Rate (Q-R) model can be written as

$$Q_i\left(r_i\right) = 10\log_{10}\frac{255^2\left(r_i + R_{0i}\right)}{D_{0i}\left(r_i + R_{0i}\right) + \theta_i} \tag{7.5}$$

In practical video codecs, the Q-R model is generally discrete. The continuity of our Q-R model can be considered as an ideal case for fine granularity. Note that, as shown in Section III-C, our proposed resource allocation algorithm will work with any Q-R model.

7.2.3 Problem Formulation

For reliable video transmission where all users are of the same importance, we consider how to achieve fair video quality among all users and formulate this problem as a max-min problem. Assume that there are $2L$ users paired into L pairs, and that the total bandwidth is divided into K subcarriers. The resource allocation algorithm wants to maximize the minimum received video quality among all user-pairs as follows:

$$\begin{aligned} \max_{\mathbf{p}} \quad & \left\{ \min_{i=1,\ldots,L} Q_i(r_i) \right\} \\ \text{subject to:} \quad & \sum_{k=1}^{K} p_i^{(k)} \leqslant p_i^{\max}, \forall i; \\ & r_i = \sum_{k=1}^{K} r_i^{(k)}, \forall i; \end{aligned} \tag{7.6}$$

where $\mathbf{p}$ is the power assignment matrix whose ik-th element is $p_i^{(k)}$, and $r_i^{(k)}$ is the bit rate of user-pair i on the k-th subcarrier. Note that each user only has the knowledge of the Q-R function of its own video stream.

This problem has both continuous and integer parameters, and non-convex constraints; therefore, it is NP-hard [13]. Even with a genie who knows every parameter of every user, it is still extremely difficult to find the optimum. In the distributed ad-hoc wireless network considered in this paper, each user only has the information about its own pair, such as CSI and SINR of each available subcarrier, and the Q-R function of its video stream, which are not shared with other users. Hence, finding a practical solution close to the optimum in a limited time is of more interest. A framework for cross-layer resource allocation in a distributed cognitive ad-hoc network is needed and will be described in the next section.

7.3 Proposed Cross-layer Resource Allocation Framework

The framework is designed from a cognitive radio perspective, where each time-frame starts with a channel sensing interval, as shown in Fig. 7.2. Passive channel sensing of the RF stimuli for each subcarrier band is intended to determine if a subcarrier is occupied. After deciding the set of available subcarriers, the noise power and channel gain at each available subcarrier can be measured at the receiver side. With all necessary information fed back via a reliable channel, the transmitter performs the cross-layer resource allocation to allocate appropriate spectral bands with associated power levels and to determine the bit rate for the video encoder. In this section, we first describe a cognitive-radio-based mechanism for channel sensing and estimation, then a distributed-spectrum and power-allocation algorithm will be presented, and finally, the cross-layer resource allocation algorithm will be proposed.

7.3.1 Detection and Estimation of Physical Channel Parameters

To detect the availability of a subcarrier, the multitaper spectrum estimation method (MTM) is employed with singular value decomposition (SVD) in order to have accurate and near-optimal performance [15, 17]. Each user first uses MTM to obtain the expansion coefficients [18], which are defined as follows:

$$y_m(f) = \sum_{n=0}^{N-1} x(n)\, \nu_n^{(m)} e^{-j2\pi f n}, \quad m = 0, 1, \ldots, M-1 \tag{7.7}$$

where N is the number of time series samples, $x(n)$ is the n-th sample of the time series, $\upsilon_n^{(k)}$ is the n-th sample of the k-th Slepian sequence with parameters N and resolution bandwidth 2ω (which is a parameter used to control the estimation variance [18]), and $M = 2\omega N$ is the number of Slepian sequences. So, the expansion coefficients in (7.7) are obtained by windowing the samples with a Slepian sequence and then Fourier transforming them.

The computed expansion coefficients contain both the energy of the background noise and the signal if present. To detect the signal in the presence of the noise, and to account for variations of spectrum at different spatial locations, and thus to improve the detection reliability, cognitive radio users exchange the computed expansion coefficients with their neighbors [15,19]. Assuming one user receives $N-1$ sets of expansion coefficients from its neighbors, $y_m^{(n)}(f)$, $n = 1, \ldots, N-1$, it can form a spatial-temporal $N \times M$ matrix [17] as follows:

$$\mathbf{A}(f) = \begin{bmatrix} y_0^{(1)}(f) & y_1^{(1)}(f) & \cdots & y_{M-1}^{(1)}(f) \\ y_0^{(2)}(f) & y_1^{(2)}(f) & \cdots & y_{M-1}^{(2)}(f) \\ \vdots & \vdots & \ddots & \vdots \\ y_0^{(N)}(f) & y_1^{(N)}(f) & \cdots & y_{M-1}^{(N)}(f) \end{bmatrix} \tag{7.8}$$

where row entries are from different spatial points and column entries are obtained with different Slepian sequences. By performing SVD upon $\mathbf{A}(f)$ and keeping the largest singular-value $\eta_0(f)$, we can significantly reduce the background noise while retaining most of the signal power [20]. The detection statistic D can be calculated as

$$D = \int_B |\eta_0(f)|^2 \, df \tag{7.9}$$

where B is the bandwidth of the subcarrier. Then, by comparing the detection statistic D with a predefined threshold, cognitive radio users can decide whether a particular subcarrier is occupied by primary users.

7.3.2 Distributed Spectrum Management and Power Control Algorithm

In a cognitive-radio-based multiuser ad-hoc network where real-time video transmission has very strict delay constraints, multi-hop relays may not be able to meet the delay and reliability requirement. To avoid this problem, users may have to transmit their video streams via a single hop, which may induce a severe near-far problem when employing multicarrier DS-CDMA modulation. Hence, dynamic spectrum management and power control plays a central role in interference mitigation. To overcome the lack of central coordination, we will present a distributed algorithm, named adaptive distributed water-filling (ADWF), which adjusts the spectrum and power allocation in an iterative way. The distributed water-filling algorithm was first proposed in [8] for digital subscriber lines (DSL), and then extended to wireless Gaussian interference channels in [9,16]. However, these algorithms all have to know the pre-determined target signal-to-interference-plus-noise ratio (SINR) or rate requirements, which have to be carefully chosen based on full knowledge of all channel and interference coefficients [8] in order to let all distributed users be able to achieve their targets. In this section, we modify the conventional distributed water-filling algorithm such that it is suitable for cross-layer resource allocation where target rates are not pre-fixed, and there is both a total power constraint on each user as well as a rate constraint on each subcarrier due to the limited modulation alphabet size.

The distributed power control algorithms work in an iterative way. Each user updates its subcarrier and power profile in every iteration based on the latest local measured information such as interference-plus-noise power level on each subcarrier. Assume the total number of subcarriers is K, and $\mathcal{K}_i$ is the set of subcarriers considered to be available to user-pair i after the sensing interval. The achievable bit rate of user-pair i on the k-th subcarrier is given by

$$r_i^{(k)} = \log_2 \left(1 + \frac{h_{ii}^{(k)} p_i^{(k)}}{\Gamma_i^{(k)} \left(I_i^{(k)} + N_i^{(k)} \right)} \right) \tag{7.10}$$

where $h_{ii}^{(k)}$, $I_i^{(k)}$, and $N_i^{(k)}$ are the associated channel gain, interference power and noise power, respectively. $\Gamma_i^{(k)}$ is the SNR-gap of user-pair i on the k-th subcarrier defining the gap between the achievable rate and the channel capacity, which depends on the symbol error rate (SER) requirement and modulation technique used [27]. For example, if MQAM is used as in this paper, the SNR-gap can be defined as follows [15]:

$$\Gamma_i^{(k)} = \frac{1}{3}\left[Q^{-1}\left(\frac{SER_i^{(k)}}{4}\right)\right]^2 \tag{7.11}$$

where $Q(\cdot)$ is the Gaussian tail function.

Let us define $\nu_i^{(k)}$ as the indicator of the channel quality of the k-th subcarrier of user-pair i:

$$\nu_i^{(k)} = \frac{h_{ii}^{(k)}}{\Gamma_i^{(k)}\left(I_i^{(k)} + N_i^{(k)}\right)} \tag{7.12}$$

Denote the target rate of user-pair i and the rate constraint on each subcarrier as r_i^{target} and $r^{\max}$, respectively. Then each transmitter solves the following optimization problem based on its local information, which is, for user-pair i, the available subcarrier set $\mathcal{K}_i$ and channel quality indicator $\nu_i^{(k)}$:

$$\begin{aligned} \min_{p_i^{(k)}, k\in\mathcal{K}_i} \quad & \sum_{k\in\mathcal{K}_i} p_i^{(k)} \\ \text{subject to:} \quad & r_i^{\text{target}} = \sum_{k=1}^{K} r_i^{(k)}; \\ & r_i^{(k)} = \log_2\left(1 + \nu_i^{(k)} p_i^{(k)}\right), \forall k \in \mathcal{K}_i; \\ & r_i^{(k)} \leqslant r^{\max}, \forall k \in \mathcal{K}_i; \\ & \sum_{k\in\mathcal{K}_i} p_i^{(k)} \leqslant p_i^{\max} \end{aligned} \tag{7.13}$$

The solution to the above problem is the well-known water-filling solution with rate constraints [28], where the effective interference-plus-noise power on each subcarrier is $\frac{1}{\nu_i^{(k)}}, \forall k \in \mathcal{K}_i$. An example is illustrated in Fig. 7.3. The water level is indicated by λ, which is chosen such that the target rate can be achieved while the power constraints are satisfied. Subcarrier 2 is saturated, i.e., it reaches its maximum supported bit rate due to the bits-per-symbol limit of the modulation scheme. Subcarriers 7 and 8 are not utilized by this user since the interference-plus-noise power exceeds the water level.

For user-pair i, denote the set of all saturated subcarriers as $\mathcal{S}_i$, and the set of all utilized subcarriers that are not saturated as $\mathcal{N}_i$. Note that both sets are subsets of the available subcarrier set of user-pair i, i.e., $\mathcal{S}_i \subseteq \mathcal{K}_i$ and $\mathcal{N}_i \subseteq \mathcal{K}_i$. For any saturated subcarrier $k \in \mathcal{S}_i$, since it reaches its maximum supported rate, i.e, $r^{\max} = r_i^{(k)} = \log_2\left(1 + \nu_i^{(k)} p_i^{(k)}\right)$, the signal power $p_i^{(k)}$ can be computed as

$$p_i^{(k)} = \frac{1}{\nu_i}\left(2^{r^{\max}-1}\right) \tag{7.14}$$

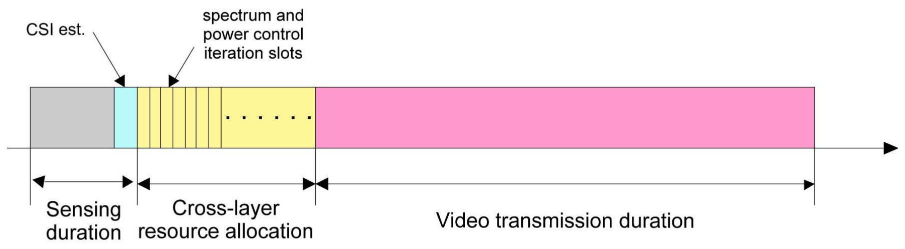

CSI est.
spectrum and power control iteration slots
Sensing duration
Cross-layer resource allocation
Video transmission duration

Figure 7.2: Transmission time-frame structure.

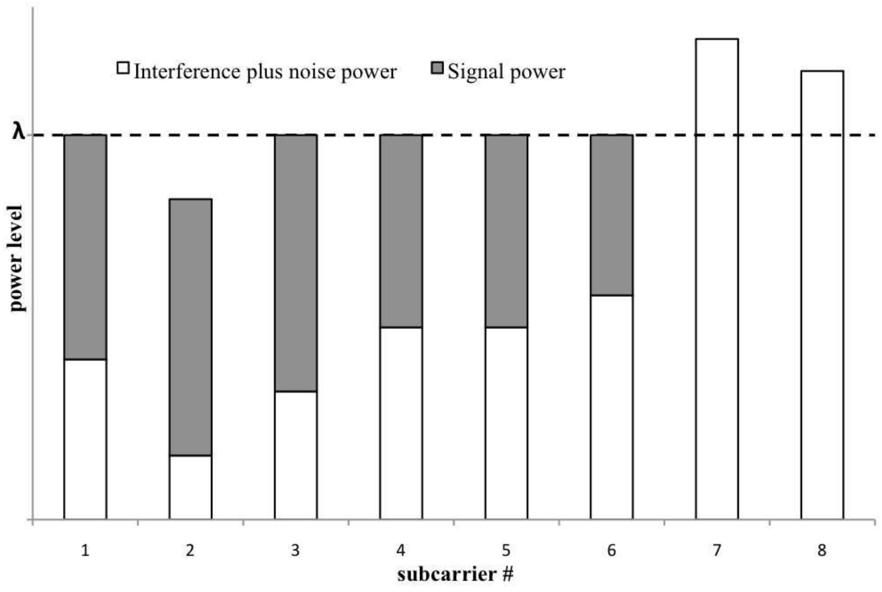


Figure 7.3: An example of water-filling solution for 8 subcarriers.

Algorithm 7.1 ADWF Algorithm

```
FOR $n = 1$ : number_of_iterations
    1) Receiver of user-pair $i$ estimates the channel quality $\nu_i^{(k)}$, $\forall k \in \mathcal{K}_i$, for every available subcarrier and feeds back this information to its transmitter.
    2) Transmitter of user-pair $i$ calculates the power level $p_i^{(k)}$ for every subcarrier $k \in \mathcal{K}_i$, as in (7.13).
    3) If the transmit power exceeds the power constraint, i.e., $\sum_{k \in \mathcal{K}_i} p_i^{(k)} > p_i^{\max}$, reduce the water level $\lambda_i$ such that $\sum_{k \in \mathcal{K}_i} p_i^{(k)} = p_i^{\max}$.
    4) Begin transmitting training sequence with updated power profile $\{p_i^{(k)}\}$.
END
```

For the unsaturated subcarriers $k \notin \mathcal{S}_i$, the signal power is found by water-filling [28]

$$p_i^{(k)} = \left(\lambda_i - \frac{1}{\nu_i^{(k)}} \right)^+ \tag{7.15}$$

where $(x)^+$ denotes $\max(x, 0)$.

The solution can be written as follows:

$$p_i^{(k)} = \begin{cases} \lambda_i - \frac{1}{\nu_i^{(k)}} & \text{for } k \in \mathcal{N}_i \\ \frac{1}{\nu_i} \left(2^{r^{\max} - 1} \right) & \text{for } k \in \mathcal{S}_i \\ 0 & \text{otherwise} \end{cases} \tag{7.16}$$

with the water level λ_i chosen to satisfy the target rate r_i^{target}.

The proposed ADWF algorithm for spectrum management and power control is described in Algorithm 7.1 . Each user-pair employs ADWF simultaneously, such that in each iteration, they update their spectrum and power allocation and transmit training sequences at their target rates in synchronized iteration time slots. We assume that the time-slot synchronization offsets between different user-pairs are much smaller than the length of an iteration time slot, so they do not affect the estimation of interference-plus-noise power at the receiver side [7]. At the end of each iteration time slot, all transmitters update their spectrum and power profile according to the interference-plus-noise power information estimated and fed back from their receivers. However, such information can be outdated in the next iteration since other transmitters probably also change their transmission spectrum and power profile, so the actual interference at receivers would differ from the last iteration, and the target rates might no longer be reached if the interference power increases.

Now, the question is whether every user-pair is able to converge to their target rates while satisfying the rate and power constraints. Sufficient conditions were found in [9, 10], but finding the necessary conditions is still an open problem. Even the sufficient conditions are based on global information, which is not available to any user in the distributed ad-hoc network. So, distributed users will not know whether their target rates are achievable until they finish the fixed number of iterations. At the end of the last iteration of the ADWF algorithm, if the SINR of the received training sequence is larger than or equal to the minimum SINR required for the allowed bit error rate (BER), the user-pair achieves its target rate; otherwise, it will broadcast a distress signal [7] to inform others that not all user-pairs have reached their target rates. Then, some or all of the users would reduce their target or temporarily stop transmission according to the cross-layer resource allocation algorithm that will be discussed in the next section.

7.3.3 Distributed Cross-layer Resource Allocation Algorithm

As illustrated by the flowchart in Fig. 7.4, we propose a distributed cross-layer resource allocation algorithm that aims to maximize the minimum PSNR among all user-pairs. We define μ_0 to be a PSNR value below which the video quality is not acceptable. Each user-pair sets the initial target PSNR to μ_0, which is taken to be 25 dB in our simulation in Section 7.4.

With the inverse function of the Q-R model in (7.5), the target PSNR of user-pair i is mapped into data rate r_i^{target} that will be used as the target rate in the ADWF algorithm. After each user-pair individually employs the ADWF algorithm for the same number of iterations, if all user-pairs converge to their target rates, i.e., no distress signal is detected, they will increase their target PSNR by $\Delta\mu$ and repeat the outer loop until at least one user cannot achieve its target rate and broadcasts a distress signal. Then every user-pair employs the previously converged-to spectrum/power profile and video coding parameter (i.e., target PSNR) to begin video transmission. It is worth pointing out that, for user-pairs whose PSNRs at zero rate are larger than the initial target PSNR μ_0 (which may occur for very low motion scenes), they will not start transmission until the target PSNR rises above their PSNRs at zero rate.

When there are too many users competing for the limited system resource so that they cannot even converge at the end of the first outer loop, i.e., some user-pairs are not able to achieve target rates after the fixed number of ADWF iterations, each user-pair i will continue the ADWF algorithm for an additional Δn_i iterations before turning off until the end of the current time-frame. The parameter Δn_i is a function of the video quality at zero rate, given by $Q_i(0)$ in (7.5). Note that for the video decoder, if there is no incoming data, it will hold the whole or part of the

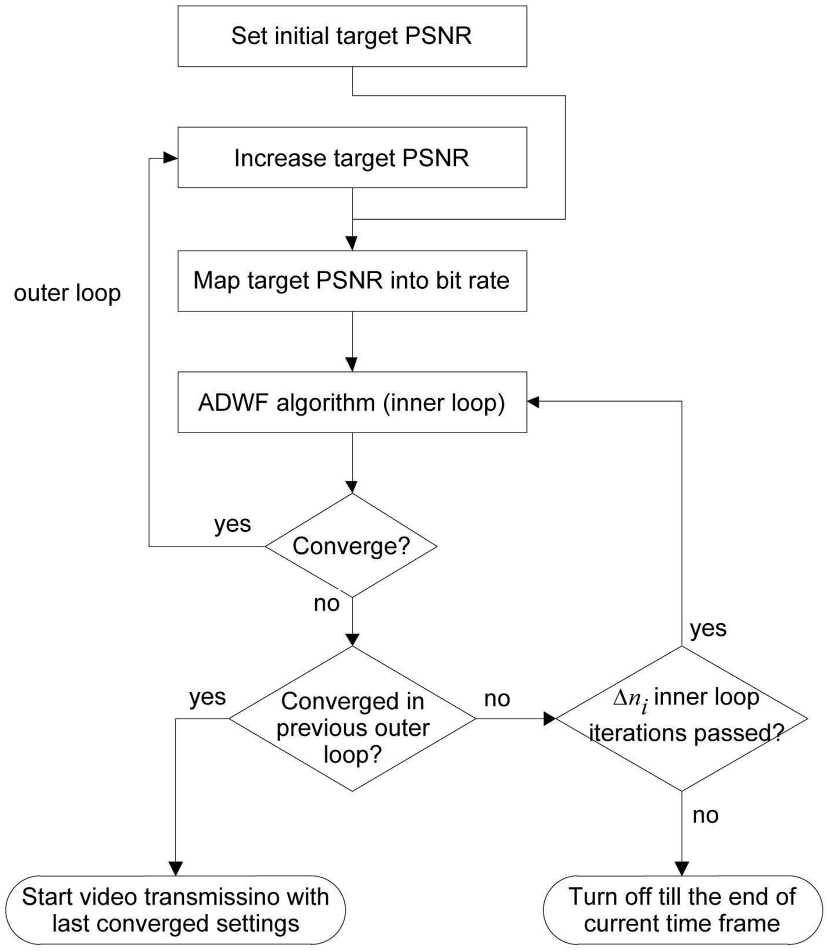

Set initial target PSNR
Increase target PSNR
outer loop
Map target PSNR into bit rate
ADWF algorithm (inner loop)
yes
Converge?
no
yes
yes
Converged in previous outer loop?
no
Δn_i inner loop iterations passed?
no
Start video transmissino with last converged settings
Turn off till the end of current time frame

Figure 7.4: Flowchart of distributed cross-layer resource allocation algorithm.

last successfully decoded video frame as the corresponding new one that is missed due to the temporary transmission stop of its transmitter. Hence, the smaller that $Q_i(0)$ is, the worse the decoded video quality will be if the transmitter turns off. So, we define Δn_i as follows:

$$\Delta n_i = \left\lfloor \alpha \left(\mu_0 - Q_i(0)\right)^+ \right\rfloor \tag{7.17}$$

where $\alpha \geqslant 0$, and $\lfloor \cdot \rfloor$ denotes the integer part of its argument. Note that this function is decreasing in its argument, capturing the intuition that the smaller that user-pair i's PSNR is at zero rate (i.e., the smaller the $Q_i(0)$ is), the longer user-pair i should continue ADWF iterations before turning off until the end of the current time-frame. For user-pairs which have turned off for more than one time-frame, their PSNR at zero rate is calculated by comparing the last transmitted video frame and the current one. Generally, if a user has not transmitted for a long time, that user's PSNR at zero rate will be small because the video contents will tend to continue diverging away from the last transmitted frame. Therefore, that user would continue the ADWF iterations and attempt to converge for a longer time before being required to turn off, as defined in (7.17). If one user has stopped transmission for a very long time, its PSNR at zero rate would probably be the smallest among all users such that it could still be on while others have turned off; therefore, it would have the opportunity to transmit in the new time-frame. In other words, no user would be kept from transmission forever.

By using the additional number of iterations defined in (7.17) to govern when to turn off, the user-pair with the largest PSNR at zero rate, i.e., user-pair $j = \arg \max\limits_{i:Q_i(0)<\mu_0} Q_i(0)$, will be the first to turn off its transmission until the end of the current time-frame, while the other active user-pairs continue the ADWF iterations and attempt to reach convergence. If they still cannot converge, the user-pair with the second largest PSNR at zero rate would turn itself off. This procedure continues until all active user-pairs are able to reach their target rate. Thus, by using the proposed distributed cross-layer resource allocation algorithm, each user-pair can independently adjust its spectrum usage and power level, and perform distributed video coding rate control with only its local SINR measurements, and therefore, the video distortion of the worst-case user is improved.

This algorithm could experience a highly stressful situation if one user-pair has a very poor channel, for example, with the transmitter located very far from the receiver, and the user-pair also has unusually complex video so their video contents require high data rate to reach even minimally acceptable quality. In such a situation, this user could take most of the resources and force many other users to have very low quality for protracted periods of time. The possibility of this type of outcome is a result of our goal of helping the quality of the worst-case user.

Although it is not within the scope of this paper, one could avoid this type of outcome by partially abandoning the goal of helping the worst-case user, forcing certain very unfavorable users to back off for a while and give other users an opportunity to transmit.

7.4 Simulation Results

The simulations are set up as follows. The distributed network has its users randomly located in an area of size 500×500 m^2. Each subcarrier experiences independent, flat Rayleigh fading. The processing gain for the multicarrier DS-CDMA system is 64. The thermal noise power density of each carrier is -145 dBm/Hz, while the maximum transmit power of each user is set to be 100 mW [26]. For simplicity, the number of subcarriers available to each cognitive user is assumed to be the same, and every available subcarrier has the same bandwidth. The distributed water-filling algorithm generally converges in a small number of iterations when the target rates are achievable [10]. Based upon the results of many runs of the simulation, we chose the fixed number of iterations of the ADWF algorithm to be 10. The parameter α in (7.17) is taken to be 10 as well. The maximum supported data per subcarrier is usually smaller than the preferred data rate of most users, thus a user-pair typically employs multiple subcarriers for its transmission. MQAM is used here, and the number of bits that can be transmitted by user-pair l over subcarrier k is [15]

$$b_l^{(k)} = \min\left\{ b_{\max}, \left\lceil \log_2\left(1 + \frac{\zeta_l^{(k)}}{\Gamma_l^{(k)}}\right)\right\rceil \right\} \tag{7.18}$$

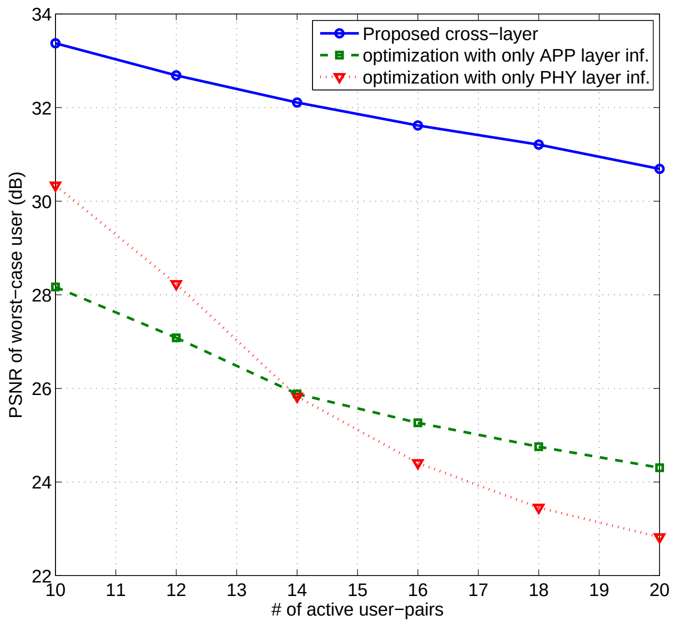
where $b_{\max}$ is determined by the maximum allowable signal constellation which is set to be 7; $\zeta_l^{(k)}$ denotes the output SINR for the k-th subcarrier of user-pair l, $\lceil x \rceil$ denotes the minimum integer larger than or equal to x, and $\Gamma_l^{(k)}$ is the SNR-gap defined in (7.11).

The video sequences used for generating the R-D curves were taken from travel documentaries at a resolution of 352×240 pixels (SIF) and at 30 frames per second. These video sequences contained various types of scenes, including high motion and low motion scenes. For each group of pictures (GOP), one R-D curve, i.e., one set of the R-D model coefficients D_0, R_0, and θ in (7.3), is generated by curve-fitting with the unconstrained nonlinear minimization approach [23]. The GOP size is 15 frames, and the frames inside are encoded using H.264 rate control. There are 60 sequences, and each sequence is 50 seconds in length. The ranges of video quality of R-D curves are within 20 to 40 dB in terms of PSNR. In the simulation, every user-pair randomly picks one video sequence to generate its R-D curves and the maximum data rate that each user-pair can require is 400kbps.

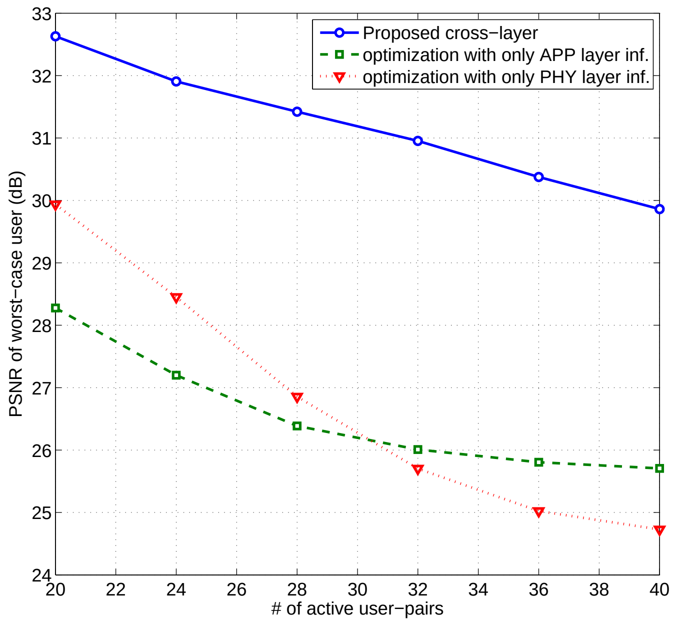
We compare the proposed cross-layer resource allocation algorithm with two non-cross-layer allocation algorithms. For optimization of the resource allocation using only application layer information, each user is randomly allocated the same number of subcarriers, and is not allowed to switch to other subcarriers or add subcarriers. Then, every user performs the ADWF algorithm within their assigned subcarriers according to the R-D curves of their video streams similar to the cross-layer algorithm. The difference between the application layer optimization and the proposed cross-layer algorithm is that in application layer optimization users are not allowed to change the number of allocated subcarriers or switch to other unassigned subcarriers. For optimization using only physical layer information, each user is allowed to use any subset of subcarriers, but without the knowledge of the video contents, i.e., the R-D curves. Each user starts with a pre-defined target transmission rate which is usually set to be the highest rate needed for typical video streaming. Then, according to the initial target rate and the CSIs of subcarriers, each user selects a subset of subcarrier and performs the distributed power allocation algorithm in [7] to obtain an achievable transmission rate which is then mapped into PSNR value for comparison.

In Fig. 7.5, we compare the PSNR of the worst-case user-pair for 16 and 32 available subcarriers. As expected, the worst-case PSNR decreases as the number of user-pairs increases, because more users are competing for the limited resources. For the application layer optimization algorithm, each user is randomly assigned the same number of subcarriers which is set to be 4 here. For the physical layer optimization algorithm, each user's initial target rate is 400kbps. It is evident from the simulation results that the video quality of the worst-case user is greatly improved with our proposed cross-layer resource allocation algorithm. There is about a 3 to 6 dB gain in terms of PSNR. Note that the PSNR values from the simulation highly depend on the transmitted video contents. The gain of the cross-layer algorithm may vary according to different types of video content. So during the simulation, we randomly pick the video content for each user from the test videos and repeat this procedure 10000 times in order to average the differences in PSNR caused by different videos.

An interesting observation is that, when the number of user-pairs is relatively small, the physical layer optimization performs better than the application layer one, but the application layer optimization outperforms the physical layer one when the number of user-pairs is large. In order to explain this, we first show a typical Q-R curve in Fig. 7.6. In this Q-R curve, the PSNR is about 22 dB when the rate is zero. As discussed in Section II-B, the video decoder at the receiver side will hold over the previous video frame when transmission stops temporally. So if



(a) 16 available subcarriers



(b) 32 available subcarriers

Figure 7.5: Worst-case distortion comparison for 16 and 32 available subcarriers with various number of active user-pairs.

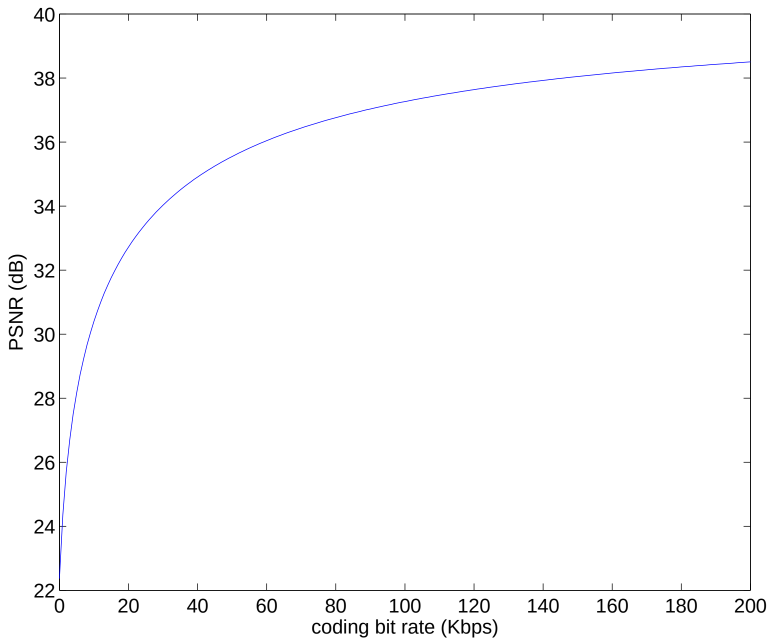
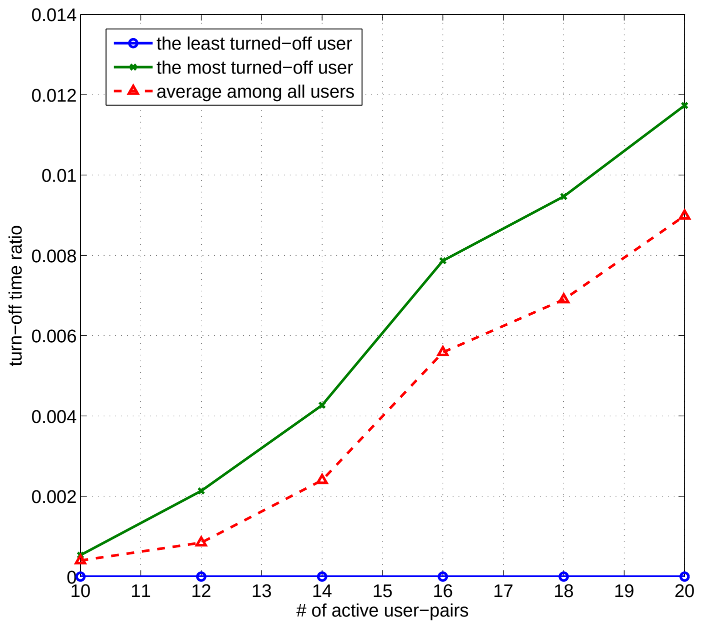


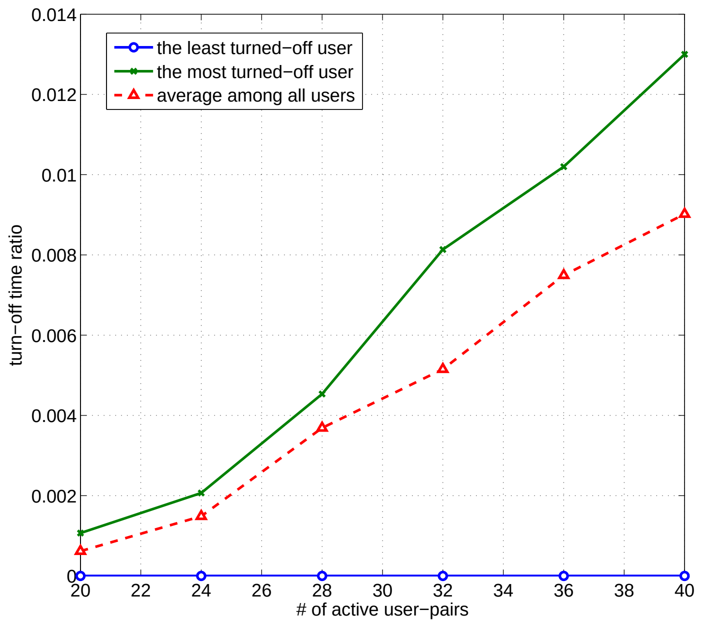
Figure 7.6: Video quality versus coding bit rate.

the video content were a more static scene, the PSNR by holding the previous video frame could still be relatively high. In a typical R-D curve, we can see that in the low-rate region, with even a slight increase of rate, the PSNR can have a significant improvement; however, for the high rate region, the increase of rate does not have as much impact on video quality. When there are fewer users in the cognitive radio ad-hoc network, each user can achieve relatively high transmission data rate (i.e., function on the relatively flat part of the Q-R curve), so the knowledge of video contents (Q-R information) is not of much value for improving the PSNR of the worst-case user, while joint spectrum and power allocation can significantly increase the data rate. However, when the number of user-pairs is large, each user can only transmit with a relatively low data rate, and the Q-R information is of great importance, because users with steeper curves deserve more resources, since even a small increase in data rate can significantly improve the PSNR.

In Fig. 7.7, the average turned-off time percentage during the total transmission period using the proposed cross-layer algorithm for 16 and 32 available subcarriers are plotted. As expected, the least turned-off user never turns off for the range of active user-pairs shown. We can see that when the number of active user-pairs is relatively small compared to the available number of available subcarriers, all the users can transmit almost all the time without having to temporally turn off. The user-pair that is forced to turn off most often only has to turn off about



(a) 16 available subcarriers



(b) 32 available subcarriers

Figure 7.7: Percentage of turned-off time. Note that the least turned-off user never turns off for the range of active user-pairs shown here.

0.1% time, i.e., only one video frame out of a thousand frames. However, as the number of active user-pairs increases, some users tend to temporally turn off more often than others. This is because these users have more static videos than do the others, so when these users temporally turn off their transmission, they still have relatively good PSNRs, and so other users can have better channel quality and thus achieve higher throughput and better PSNRs.

7.5 Conclusion

In this paper, we presented a cognitive radio based cross-layer resource allocation framework for video streaming over distributed networks using multicarrier DS-CDMA modulation with a frequency selective fading channel. In particular, we considered multiple video streams in the distributed network and performed joint resource allocation over both the application layer and the physical layer. The objective is to minimize the maximum distortion of all transmitted video streams. Our simulation results showed a significant performance gain over schemes that allocate subcarriers and video rates individually, improving the video quality of the worst-case user by 3 to 6 dB in PSNR.

Acknowledgment

This chapter is on a project when the author was working with Prof. Pamela Cosman and Prof. Laurence Milstein, and the research was supported in part by the Office of Naval Research under Grant no. N000140810081.

Bibliography

[1] R. Knopp and P. A. Humblet, “Information capacity and power control in single-cell multiuser communications,” in *Proc. of International Conference on Communications (ICC)*, Jun. 1995, vol. 1, pp. 331–335.

[2] C. Y. Wong, R. S. Cheng, K. B. Letaief, and R. D. Murch, “Multiuser OFDM with adaptive subcarrier, bit, and power allocation,” *IEEE J. Select. Areas Commun.*, vol. 17, pp. 1747–1758, Oct. 1999.

[3] Z. Han, Z. Ji, and K. Liu, “Fair multiuser channel allocation for OFDMA network using Nash bargaining solutions and coalitions,” *IEEE Trans. Commun.*, vol. 53, no. 8, pp. 1366–1376, Aug. 2005.

[4] C. Ng and C. Sung, "Low complexity subcarrier and power allocation for utility maximization in uplink OFDMA systems," *IEEE Trans. Wireless Commun.*, vol. 7, no. 5, pp. 1667–1675, May 2008.

[5] S. Kondo and L. B. Milstein, "Performance of multicarrier DS CDMA systems," *IEEE Trans. Commun.*, vol. 44, no. 2, pp. 238–246, Feb. 1996.

[6] G. J. Foschini and Z. Miljanic, "A simple distributed autonomous power control algorithm and its convergence," *IEEE Trans. Veh. Technol.*, vol. 42, pp. 641–646, Nov. 1993.

[7] N. Bambos, S. C. Chen, and G. J. Pottie, "Channel access algorithm with active link protection for wireless communication networks with power control," *IEEE/ACM Trans. Networking*, vol. 8, pp. 583–597, Oct. 2000.

[8] W. Yu, G. Ginis, and J. M. Cioffi, "Distributed multiuser power control for digital subscriber lines," *IEEE J. Select. Areas Commun.*, vol. 20, pp. 1105–1115, Jun. 2002.

[9] O. Popesce, D. C. Popescu, and C. Rose, "Simultaneous water filling in mutually interfering systems," *IEEE Trans. Wireless Commun.*, vol. 6, pp. 1102–1113, Mar. 2007.

[10] J. Pang, G. Scutari, F. Facchinei, and C. Wang, "Distributed power allocation with rate constraints in Gaussian parallel interference channels," *IEEE Trans. Inf. Theory*,vol. 54, pp. 3471–3489, Aug. 2008.

[11] Y. Su and M. der van Schaar, "Multiuser multimedia resource allocation over multicarrier wireless networks," *IEEE Trans. Signal Process.*, vol. 56, pp. 2102–2116, May 2008.

[12] C. Shen, M. der van Schaar, "Optimal resource allocation for multimedia applications over multi-access fading channels," *IEEE Trans. Wireless Commun.*, vol. 7, pp. 3546–3557, Sep. 2008.

[13] G. Su, Z. Han, M. Wu, and K. J. R. Liu, "A scalable ultiuser framework for video over OFDM networks: fairness and efficiency," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 16, pp. 1217–1231, Oct. 2006.

[14] S. Adlakha, X. Zhu, B. Girod, and A. Goldsmith, "Joint capacity, flow and rate allocation for multiuser video streaming over wireless ad-hoc networks," in *Proc. of International Conference on Communications (ICC)*, Jun. 2007, pp. 1747–1753.

[15] Q. Qu, L. B. Milstein, and D. R. Vaman, "Cognitive radio based multi-user resource allocation in mobile ad-hoc networks using multi-carrier CDMA modulation," *IEEE J. Select. Areas Commun.*, vol. 26, pp. 70–81, Jan. 2008.

[16] G. Scutari, D. P. Palomar, and S. Barbarossa, "Asyschronous iterative water-filling for Gaussian frequency-selective interference channels", *IEEE Trans. Inf. Theory*, vol. 54, pp. 2868–2878, Jul. 2008.

[17] M. E. Mann and J. Park, "Oscillatory spatialtemporal signal detection in climate studies: a multiple-taper spectral domain approach," *Advance in Geophysics*, New York Academic, vol. 41, 1999.

[18] D. J. Thomson, “Spectrum estimation and harmonic analysis,” *Proc. IEEE*, vol. 70, pp. 1055–1096, Sep. 1982.

[19] S. M. Mishra, A. Sahai, and R. W. Broderson, “Cooperative sensing among cognitive radios,” in *Proc. of International Conference on Communications (ICC)*, Jun. 2006, vol. 4, pp. 1658–1663.

[20] L. Cohen, *Time-Frequency Analysis*. Prentice Hall, 1995.

[21] F. Meshkati, M. Chiang, H. V. Poor, and S. C. Schwartz, “A game-theoretic approach to energy-efficient power control in multicarrier CDMA systems,” *IEEE J. Select. Areas Commun.*, vol. 24, pp. 1115–1129, Jun. 2006.

[22] A. Ortega and K. Ramchandran, “Rate-distortion methods for image and video compression,” *IEEE Signal Processing Mag.*, vol. 15, pp. 23–50, Nov. 1998.

[23] K. Stuhlmüller, N. Färber, M. Link, and B. Girod, “Analysis of video transmission over lossy channels,” *IEEE J. Select. Areas Commun.*, vol. 18, pp. 1012–1032, Jun. 2000.

[24] M. van der Schaar and D. S. Turaga, “Cross-layer packetization and retransmission strategies for delay-sensitive wireless multimedia transmission,” *IEEE Trans. Multimedia*, vol. 9, pp. 185–197, Jan. 2007.

[25] M. Wang and M. van der Schaar, “Operational rate-distortion modeling for wavelet video coders,” *IEEE Trans. Signal Process.*, vol. 54, no. 9, pp. 3505–3517, Sep. 2006.

[26] A. Stranne, O. Edfors, and B. A. Molin, “Energy-based interference analysis of heterogeneous packet radio networks,” *IEEE Trans. Commun.*, vol. 54, pp. 1299–1309, July 2006

[27] M. H. Ahmed, H. Yamikomeroglu, and S. Mahmoud, “Fairness enhancement of link adaptation techniques in wireless networks,” in *IEEE Vehic. Technol. Conf. (VTC)*,Sep. 2003, vol. 4, pp. 1154–1157.

[28] T. M. Cover and J. A. Thomas, *Elements of Information Theory*. New York: Wiley, 1991.