

OVERLAPPED FOUNTAIN CODING: DESIGN AND ANALYSIS

by

KHALED FAROUQ HAYAJNEH

A thesis submitted to the
Department of Electrical and Computer Engineering
in conformity with the requirements for
the degree of Doctor of Philosophy

Queen's University
Kingston, Ontario, Canada

August 2017

Copyright © Khaled Farouq Hayajneh, 2017

Abstract

The concept of fountain codes has gained considerable attention in the past few years due to its simplicity, reliability, and feasibility. Nowadays, fountain codes are used in many applications including, but not limited to, data storage, data broadcasting, and point-to-point communications. While traditional fountain codes achieve the channel capacity over the binary erasure channel universally and asymptotically, they offer much room for improvement over other channels, architectures, and regimes. With the development of new technologies for smart cities and the Internet-of-Things (IoT), data transmission methodologies with arguably the highest level of flexibility and adaptability are required. With these technologies, the end users have a very diverse set of capabilities in terms of memory, power, and processing. Besides, the end users are connected via a wide range of links with various qualities and capacities. As a result, the required methodologies must provide a comprehensive solution whereby voice, data, and streamed multimedia can be provided to users on an anytime-anywhere basis. Also, they must enable a dense network of nodes with very different capabilities to communicate with each other reliably and efficiently.

This thesis focuses on the design of state-of-the-art data transmission methodology with one of the highest levels of flexibility and adaptability. We present a novel fountain-based encoding technique using overlapped generations of Luby-transform

(LT) codes. The proposed overlapped LT (OLT) codes provide more degrees of freedom and better trade-offs for the rateless-coded system parameters. They are highly energy-efficient, scalable, and robust.

First, we design new LT codes that are robust to the communication system's parameters such as erasure probability ϵ as well as the source length k . Density evolution (DE), extrinsic information transfer (EXIT) chart, and code stability are used to design the parameters of fountain codes with different objectives. The main objectives are: maximizing the code rate, minimizing the bit erasure probability (BEP), and maximizing erasure threshold. Secondly, we propose OLT codes over the binary erasure channel (BEC) as well as additive white Gaussian noise (AWGN) channel to improve the performance in terms of error probability and code rate. Our analysis shows that by using OLT codes, significant gains in error probability and code rate are obtained. In addition, we show that OLT codes require a smaller number of operations to recover the source data compared with conventional LT codes.

Acknowledgments

In the Name of God, the Most Gracious, the Most Merciful. All praises and thanks be due to the God for his blessing, guidance, and help during my life. I could never have done this work without his grace and mercy.

I would like to thank my extraordinary supervisor Prof. Shahram Yousefi. There are no words that can express my deepest gratitude for his help and support. I am highly indebted to him for his patience, motivation, encouragement, guidance, and advice through my PhD studies. I am indeed a lucky PhD student to have a supervisor like him.

I would also like to take this opportunity to thank my examination committee for their time and efforts. Special thanks go to Prof. Il-Min Kim, Prof. Hossam S. Hassanein, Prof. Oussama Damen, Prof. Bahman Gharesifard, and Prof. Michael Korenberg for their constructive comments and suggestions which have significantly contributed my thesis.

Throughout my PhD studies at the Department of Electrical and Computer Engineering, I felt very fortunate to be taught by talented instructors who have increased my knowledge, and put me on the cutting edge of the research. In particular, I would like to thank Prof. Michael Korenberg, Prof. Il-Min Kim, Prof. Saeed Gazor, Prof.

Naraig Manjikian, Prof. Evelyn Morin, Prof. Geoffrey Chan, Prof. Fady Alajaji, and, certainly, my supervisor Prof. Shahram Yousefi. I am also grateful to Queen's of being given the opportunity to teach an undergraduate course in the ECE department during my PhD studies. It has been a privilege and an enjoyable experience, and I have learned a lot from this experience. I would also like to thank Yarmouk University-Jordan for giving me the opportunity and the scholarship to continue my graduate studies.

I am also grateful for the help offered by Queen's staff throughout my studies. Special thanks go to Debie who helped and supported me from the very first moment I arrived Queen's. I thank past and present members of SDAL: Mehrdad Valipour, Hossein Khonsari, Toritseju Okpotse, Yaser Esmaeili, and James Spencer. Their continual support and suggestions were helpful and added so much to my work.

I would also like to thank the wonderful friends at Queen's whom I have shared so many memories with. In particular, Abdallah Alma'aitah, Basel Alnabulsi, Ala Abu Alkheir, Ayman Sabbah, Mahmud Qutqut, Mohannad Alswailim, Imad Odat, Ashraf Alkhresheh, Muad Ghaith, Rafiq Manna, Aymen Mnasri, Khalid Elgazzar, Abdullah Nasr, Malek Karaim, Ahmed Alghamdi, Ghaith Hattab, Almounir Alkhazmi, Yaser Almtawa, Mustafa Mohamad, Ramy Atawia, Anas Mahmoud, Abdullah Abdulrahman, Phillip Oni, Hisham Farahat, Mohamed Shoeb, and Mohamed Adel. You have made my PhD journey a lot more fun and enjoyable as well. I shall miss you all. Besides, I am thankful to all my friends in Jordan (too numerous to list!) who also encouraged me in this journey.

To my parents, Farouq and Amira, I will always be thankful and grateful for what you have done to make my life better. I would not have made this work without your unflagging support, love, and prayers. I owe you any achievement I make throughout my entire life. To my brothers and sisters, thank you so much for all of encouragement and support.

Last, but definitely not least, I would like to express my sincere gratitude to my beautiful wife, Sahar, for sacrificing so much time and supporting me during this long journey with all the ups and downs. She industriously provides me the best possible environment while studying, working, and writing this dissertation. I would like to thank my two lovely sons, Farouq and Abdullah, for being a part of my PhD.

Khaled Hayajneh
Kingston, Ontario, Canada
August 2017

Contents

Abstract	i
Acknowledgments	iii
Contents	vi
List of Figures	viii
List of Abbreviations	xii
Chapter 1: Introduction	1
1.1 Advantages of Error Control Systems	2
1.2 Contributions	8
1.3 Organization of the Thesis	10
Chapter 2: Channel Coding Primer	11
2.1 Forward Error Correction Codes	13
2.2 Communication Channels	14
2.2.1 Binary Erasure Channel (BEC)	16
2.2.2 Binary Symmetric Channel (BSC)	17
2.2.3 Binary-Input Additive White Gaussian Noise (BIAWGN) Channel	18
2.3 Channel Codes	19
2.3.1 Block Codes	19
2.3.2 Convolutional Codes	21
2.3.3 Low-Density Parity-Check (LDPC) Codes	22
2.3.4 Low-Density Generator-Matrix (LDGM) Codes	26
2.3.5 Rateless Codes	26
2.4 Fountain Codes	29
2.4.1 Random Linear Fountain (RLF) Codes	30
2.4.2 Luby-Transform (LT) Codes	36
2.4.3 Raptor Codes	49

Chapter 3: Degree Distribution Design	51
3.1 Basic Notations	52
3.2 Code Analysis	55
3.2.1 Density Evolution (DE)	55
3.2.2 Extrinsic Information Transfer (EXIT) Chart	59
3.2.3 Code Stability	60
3.3 Design RSD for Finite-Length Regime	64
3.3.1 Tuning the Degree Distribution for Robustness	65
3.3.2 RSD Parameters via Monte Carlo Optimization	69
3.4 Numerical Results	70
3.5 Summary	78
Chapter 4: Overlapped Fountain Codes	79
4.1 Introduction	79
4.2 Overlapped LT (OLT) Codes	83
4.2.1 Encoding Process of OLT Codes	86
4.2.2 Decoding Process of OLT Codes	87
4.3 Performance Analysis	88
4.3.1 Conventional LT Codes	89
4.3.2 OLT Codes	90
4.4 Numerical Results	93
4.5 Summary	100
Chapter 5: OLT Codes over AWGN Channels	102
5.1 Introduction	102
5.2 System Model over AWGN Channels	103
5.2.1 Encoding Systematic LT Codes	103
5.2.2 Decoding over AWGN Channels	104
5.3 OLT Codes	107
5.3.1 Encoding OLT Codes	108
5.3.2 Decoding OLT Codes	109
5.4 Smart Overlapped LT (SOLT) Codes	111
5.5 Numerical Results	113
5.6 Summary	115
Chapter 6: Conclusions	117
6.1 Summary	118
6.2 Future Work	119
Bibliography	122

List of Figures

1.1	Venn diagram of rateless codes.	7
2.1	The generalized coding system block diagram.	13
2.2	Binary input, symmetric, discrete, and memoryless channel.	15
2.3	A binary erasure channel (BEC) with erasure probability ϵ	16
2.4	A binary symmetric channel with cross-over probability p	17
2.5	AWGN channel.	18
2.6	A convolutional encoder with memory order of $m = 3$ and code rate $R = 1/3$	22
2.7	Tanner graph of a regular LDPC code with $w_c = 2$ and $w_r = 3$. The circles represent $n = 6$ encoded bits while the squares represent $m = 4$ parity checks.	24
2.8	Upper bound on failure probability for RLF codes versus the overhead ϵ at generation sizes of $k = 128$, $k = 256$, and $k = 512$	34
2.9	Lower bound on the required number of encoded bits n in RLF codes versus generation size k at different values of c	35
2.10	Tanner graph of an LT code. Circles represents the source bits and squares represents the encoded bits	38

2.11	An example of a BP decoder over a BEC. There are $k=3$ source bits and the decoder receives $n = 4$ encoded bits.	42
2.12	Ideal soliton distribution (ISD) and robust soliton distribution (RSD) at generation size $k = 128$, $c = 0.02$, and $\delta = 0.1$	46
2.13	Robust soliton distribution (RSD) at generation size $k = 128$ and $\delta = 0.1$ with different values of parameter c	47
2.14	Raptor coding scheme: $(k + m)$ intermediate encoded bits are generated from k source bits using a pre-coding (e.g., LDPC codes). Then, LT encoder generates n encoded bits using the $(k + m)$ intermediate encoded bits.	50
3.1	An example of And-Or tree for a subgraph G_l where circles and squares represent OR-nodes and AND-nodes, respectively.	57
3.2	EXIT chart for an LT code with packet length $k = 128$ and code length $n = 256$ over different BEC's. RSD degree distribution is used with parameters $c = 0.02$ and $\delta = 0.5$	63
3.3	EXIT chart for an LT code with packet length $k = 128$ and code length $n = 256$ over a BEC of erasure probability $\epsilon = 0.2$. RSD degree distribution is used with parameters $c = 0.02$ and $\delta = 0.5$	64
3.4	Upper and lower bounds on parameter c versus packet lengths k of LT codes with fixed δ	67
3.5	Failure probability δ versus parameter c of LT codes at different packet lengths k	68
3.6	Code rate R of an LT code versus parameters c and δ at packet length $k = 128$	72

3.7	Code rate R of an LT code versus parameters c and δ at packet length $k = 512$	73
3.8	Code rate R of an LT code versus parameters c and δ at packet length $k = 1024$	74
3.9	BEP versus parameter c at packet length $k = 128$, erasure probability $\epsilon = 0.02$, and $\delta = 0.2$. The code rate R ranges from $R = 1$ (upper curve) to $R = 1/2$ (lower curve).	75
3.10	EXIT chart for an LT code with packet length $k = 128$ and code length $n = 256$ over a BEC of erasure probability $\epsilon = 0.45$. RSD degree distribution is used with parameters $c = 0.035$ and $\delta = 0.5$	76
3.11	EXIT chart for an LT code with packet length $k = 128$ and code length $n = 256$ over a BEC of erasure probability $\epsilon = 0.45$. RSD degree distribution is used with parameters $c = 0.04$ and $\delta = 0.5$	77
4.1	The required number of generations g to cover all the source data $K = 1024$ bits versus the overlapped percentage α/k at different generation sizes k	85
4.2	Broadcasting to five receivers with BEC's of erasure probabilities $\epsilon = \{0.3\ 0.2\ 0.1\ 0.01\ 0.001\}$	93
4.3	OLT codes at $K = 1024$, $k = 256$, and $\alpha = 128$	94
4.4	Code rate R of five users with different erasure probabilities using different schemes.	95
4.5	Average number of edges versus erasure probability for different schemes.	96

4.6	Analytical comparison of the number of removed edges at each decoding step between LT and OLT codes with different code rates at $K = 1024$, $k = 256$, $\alpha = 128$, and $p = 0.65$	98
4.7	Simulation comparison of the number of removed edges at each decoding step between LT and OLT codes with zero erasure probability at $K = 1024$, $k = 256$, $\alpha = 128$, and $p = 0.65$	99
4.8	Average complexity (right) and average code rate (left) of OLT codes versus selection probability p at zero erasure probability $\epsilon = 0$, $K = 1024$, $k = 256$, and $\alpha = 128$	100
4.9	Average complexity (right) and average code rate (left) of OLT codes versus selection probability p at erasure probability $\epsilon = 0.1$, $K = 1024$, $k = 256$, and $\alpha = 128$	101
5.1	Message passing from output bit u_t to its neighbor source bit v_1	106
5.2	Message passing from source bit v_t to its neighbor output bit u_1	107
5.3	Encoding OLT codes.	110
5.4	BER versus inverse code rate for LT, OLT, and SOLT codes at different SNR's.	114
5.5	BER versus inverse code rate for LT, OLT, and SOLT codes at different SNR's. These codes are generated using Shokrollahi's degree distribution.	115

List of Abbreviations

ACK	Acknowledgment
ARQ	Automatic Repeat reQuest
AWGN	Additive White Gaussian Noise
AM	Amplitude Modulation
BEC	Binary Erasure Channel
BEP	Bit Erasure Probability
BIAWGN	Binary-Input Additive Gaussian Noise
BP	Belief Propagation
BPSK	Binary Phase Shift Keying
BSC	Binary Symmetric Channel
CRC	Cyclic Redundancy Check
CSI	Channel State Information
D2D	Device-to-Device

ECC	Error-Correcting Code
ECS	Error Control System
FEC	Forward Error Correction
FM	Frequency Modulation
GF	Galois Field
ICT	Information and Communication Technology
i.i.d.	Independent and Identically Distributed
IoT	Internet-of-Things
IP	Internet Protocol
ISD	Ideal Soliton Distribution
LDD	Left Degree Distribution
LDGM	Low-Density Generator-Matrix
LDPC	Low-Density Parity-Check
LLR	Log-Likelihood Ratio
LT	Luby Transform
MAP	Maximum A Posteriori
MCO	Monte Carlo Optimization

METIS	Mobile and Wireless Communications Enablers for Twenty-Twenty (2020) Information Society
MIMO	Multiple-Input Multiple-Output
ML	Maximum-Likelihood
MLC	Multilevel Coding
mmWave	Millimeter Wave
MPA	Message Passing Algorithm
MTC	Machine-Type Communication
NAK	Negative Acknowledgment or Not Acknowledged
PCM	Pulse-Code Modulation
pdf	Probability Density Function
PET	Priority Encoding Transmission
PRNG	Pseudorandom Number Generator
QoS	Quality of Service
RLF	Random Linear Fountain
RS	Reed-Solomon
RSD	Robust Soliton Distribution
SNR	Signal-to-Noise Ratio

TCM	Trellis Coded Modulation
TCP	Transmission Control Protocol
XOR	Exclusive-OR
5G	5th-Generation Mobile Networks

Chapter 1

Introduction

Smart cities and the Internet-of-things (IoT) notions encapsulate some of the most important technological advances in progress in this so-called fourth industrial revolution [1]. The information and communication technology (ICT) sector will have a number of very interesting problems to investigate for the next few years. The full realization of IoT is one of the ICTs most uniquely identifiable challenges. How will the IoT provide a comprehensive solution whereby voice, data, and streamed multimedia can be provided to users on an anytime-anywhere basis? How will the IoT enable a dense network of nodes with very different capabilities to communicate with each other reliably and efficiently? Different quality of service (QoS) and transmission rates are demanded for applications such as wireless broadband access, multimedia messaging service, video chat, mobile TV, high-definition TV, as well as minimal services such as voice and data [2–4]. For example, according to the mobile and wireless communications enablers for twenty-twenty (2020) information society (METIS), the 5th-generation mobile networks (5G) target 1000 times higher mobile data, 10-100 times higher number of connected devices, 10-1000 higher typical user data rate, and 5 times reduced end-to-end latency [5]. The realization of such seamless connectivity

requires the synergy of a myriad of technologies.

Various solutions have been proposed to improve the data transmission quality and cost. These solutions include, but not limited to, backhaul integration (e.g., millimeter wave (mmWave) [6] and macro cells [7]), flexible duplex [8, 9], massive multiple-input multiple-output (MIMO) transmissions [10], machine-type communication (MTC) [11], and device-to-device (D2D) communication [12, 13]. One major and inseparable component of all advanced communication networks is the error control system (ECS).

1.1 Advantages of Error Control Systems

ECS is required to provide *reliable* communication over the above-mentioned scenarios [14, 15]. Reliable communication means that data can be sent from a transmitter to a receiver with an arbitrarily small probability of error.

In communication systems, the received data is not identical to the transmitted one with some probability thanks to phenomena such as cross-talk in a telephone system, delay, distortion, degradation, and multi-path in wireless systems, or congestion and buffer overflows in the Internet. As communication system designers, we prefer to reduce this probability (or errors induced by the channel, in other words) as much as possible. Furthermore, for some applications, we prefer to have zero error probability. Thus, the need for ECS is always there.

Mainly, there are two kinds of ECSs depending on the nature of communication channel, applications, and requirements: *automatic repeat request* (ARQ) and *forward error correction* (FEC). To understand these approaches to ECS, one must distinguish between different types of communication systems. Communication systems can be

categorized into two types: ‘one-way communications’ where only one transmitter and one receiver are involved in the system (the transmission is only in one direction). Deep space communication is an example of this type of communications. The other type is ‘two-way communications’ where the receiver and the transmitter can transmit and receive data (the transmission is in both directions). Data networks, wireless communications, satellite communications, *etc.*, are examples of this type of communications [16].

Automatic Repeat Request

When errors are detected at the receiver, a repeat request is sent to the transmitter. A familiar example of ARQ is transmission control protocol (TCP) which is the most widely used transmission protocol on the Internet [17]. There are two types of ARQ strategy: stop-and-wait ARQ and selective-repeat ARQ. In stop-and-wait ARQ, the transmitter sends a message and waits for either a positive acknowledgment (ACK) or a negative acknowledgment (NAK). If an ACK is received, the transmitter continues and sends the next message but if a NAK is received, the transmitter holds and resends the previous message. In selective-repeat ARQ, the transmitter sends the packets and receives ACK’s continuously. Once the transmitter receives a NAK for a packet, it backs and re-sends only that packet again. This type requires higher number of operations and memory, relatively [16].

Forward Error Correction

FEC strategy is used when there is no feedback channel from the receiver to verify if the data has been received correctly or not. In addition, FEC can be used in the

situations where the feedback channels are too costly. Thus, the transmitter must ensure that the receiver will get the data *reliably* even when there are errors induced by the noisy channel. In FEC, similar to ARQ, the transmitter encodes the source data by adding controlled redundancy. The redundancy allows *reliable* transmission over the channel. FEC corrects the errors induced by the channel at the receiver side without the need for any retransmissions.

Both ARQ and FEC strategies add redundancy to the source data. The former needs low redundancy to simply detect errors and then request for retransmissions when needed. However, FEC requires more redundancy to allow the receiver to correct errors. Essentially, ARQ strategy requires error detection codes with simpler encoding and decoding algorithms than the ones used in error correction codes. In addition, ARQ strategy can be used adaptively in which retransmission is requested only for those bits, symbols, or packets that considered to have errors. These are the main advantages of using ARQ strategy over FEC. In contrast, ARQ strategy needs an ideal feedback channel. Also, an extra burden is placed on the network to transmit the receiver's acknowledgments. Further, loss of a packet may take long time to be communicated to the transmitter introducing additional delay to the network.

The most significant disadvantage of ARQ strategy occurs in broadcast and multicast scenarios in which one sender is transmitting data to multiple receivers [18]. A typical example is distributing live video over the Internet. If the number of receivers is large, and each receiver loses a small fraction of the packets and requests retransmission, the transmitter may receive a prohibitively large number of retransmission requests.

Because of the above-mentioned drawbacks of ARQ, we are only interested in FEC

strategy. Different coding techniques are proposed under FEC and ARQ strategies. These coding techniques can be classified into two different types: block codes and convolutional codes. This category is based on the encoder functions. In block codes, the encoder maps a vector of k source bits into a codeword of length n (i.e., the codeword depends only on the k source bits). On the contrary, convolutional encoder maps a sequence of arbitrarily length to a codeword with arbitrary length. As a result, the encoded bits depend not only on the current k source bits but also on several consecutive past input bits. A different taxonomy of coding techniques is based on the rate design. If the rate is fixed beforehand, the coding technique is referred to as fixed-rate code. However, if the transmitter sends the encoded messages limitlessly, the coding technique is referred to as rateless codes. Other taxonomies are based on the code structure, for example linear/nonlinear codes and binary/nonbinary codes [16, 19]. One notable invention is that of fountain coding which is a type of rateless codes [20–22].

Rateless codes are considered as a hybrid between FEC and ARQ protocols where limited feedback channels can be used. Fountain codes are a specific realization of rateless codes. In fountain codes, a potentially limitless sequence of encoded messages (comparing with fixed number in fixed-rate block codes) can be generated from the k source bits such that any receiver, who wishes to decode the source bits, can recover the k source bits from any subset of encoded bits of size slightly larger than the length of source bits k . Fountain coding technique can solve drawbacks associated with ARQ such as time sensitivity. In fountain codes, the decoder starts the decoding process as soon as it receives k encoded bits. If the decoder fails to fully recover the k source bits, it keeps receiving more encoded bits until the k source bits are fully

recovered (no need for retransmission). In addition, fountain coding technique can solve drawbacks of fixed-rate codes particularly when the channel statistics are unknown to the sender. In this case, a fixed-rate code generates the encoded bits with an arbitrary rate. As a result, the rate is either too high (overestimation) and that leads to *unreliable* communications (outage) or the rate is too low (underestimation) which leads to unnecessary redundant encoded bits (i.e., waste of bandwidth). Also, another shortcoming of fixed-rate codes is found in case of point-to-multipoint communications. In this case, fountain codes ensure that each endpoint *reliably* decodes the k source bits with the reception of minimal encoded bits (i.e., each endpoint will recover the source data with a transmission rate that depends on its own channel characteristics and capability). Basically, fountain codes shift the code adaptivity from the transmitter to the receiver. Therefore, fountain codes are best fit to our target multicast/broadcast applications.

Inspired by these, this thesis focuses on the data transmission methodologies that are adaptive, scalable, and efficient in terms of both power and bandwidth requirements. Specifically, we set out to utilize the digital fountain idea to propose a novel coding strategy as an enabler of the following objectives: 1- High bandwidth efficiency 2- Low encoding complexity 3- Low decoding complexity 4- Fast decoding process 5- Low latency. Digital fountain codes were originally proposed as a solution for cases where channel state information is not available at the transmitter or when the channel variations are hard to track. Luby-transform (LT) codes and Raptor codes are the first practical realizations of fountain codes [23,24]. Figure 1.1 shows the relationship among rateless codes using Venn diagram.

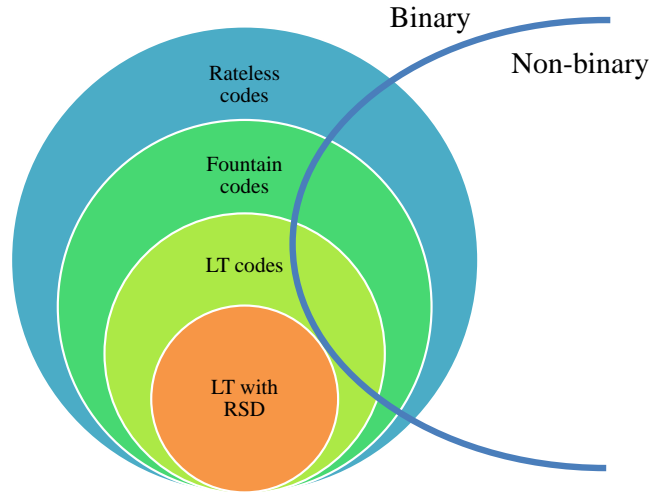


Figure 1.1: Venn diagram of rateless codes.

Originally, fountain codes were proposed for the binary erasure channel (BEC) [20]. Communication over the Internet with buffer overflows, failed cyclic redundancy check (CRC), etc., can be modeled via simple erasure channel models. Nowadays, however, fountain codes have been used successfully over a variety of *noisy* channels such as additive white Gaussian noise (AWGN) channel, binary symmetric channel (BSC) and fading channel [25–28]. They are also applied to a variety of networks and over different alphabets (binary and non-binary) [29–36].

Despite their good performance and promising applications in many areas, fountain codes exhibit some weaknesses providing much room for improvement. Such weaknesses limit the performance of the codes. For example, LT codes with robust soliton distribution (RSD) are merely asymptotically optimal (i.e., when the code length k tends to infinity) and the channel is modeled as BEC. However, a finite code length is used in practice. Thus, one can design new fountain codes that perform better than conventional fountain codes in the practical finite-length regime. We also

discuss methodologies for the design of code parameters to provide the right balance in terms of a number of the above-mentioned conflicting design objectives.

1.2 Contributions

The main contributions of this thesis are as follows:

- We design robust fountain codes in terms of the channel erasure probability and the code length k . Density evolution (DE), extrinsic information transfer (EXIT) chart, and code stability are used to optimize the parameters of fountain codes with different objectives. Three particular objectives are used: maximizing the code rate R , minimizing the bit erasure probability (BEP), and maximizing erasure threshold.
- We introduce the concept of overlapped generations for fountain codes over a BEC. Also, we focus on broadcast scenario and particularly target the improvement of latency and complexity for cases where users with a wide range of erasure rates listen to a single fountain source through BEC links. We show via analytical and simulation methods that our method can result in substantial code rate gains comparing to conventional fountain codes. In addition, our overlapped generation scheme can be used to provide better delays and complexities. Besides, we optimize the parameters of the new scheme targeted to maximize the code rate and/or minimize the complexity.
- We apply the concept of overlapped fountain codes over noisy channels such as AWGN. We show that the proposed scheme improves the code performance in terms of code rate and complexity. In addition, we design a new degree

distribution that best fits the proposed overlapped fountain codes.

The following are the publications related to the material presented in this thesis:

- **K. F. Hayajneh** and **S. Yousefi**, “Overlapped LT codes: Design and Analysis,” submitted to *IEEE Transactions on Communications*, 2017.
- **K. F. Hayajneh** and **S. Yousefi**, “Robust LT Designs in Binary Erasures,” in *Proceedings of 15th Canadian Workshop on Information Theory (CWIT), 2017*, Quebec City, Quebec, Canada, 11-14 June, 2017, pp. 1-5.
- **K. F. Hayajneh** and **S. Yousefi**, “Towards a Smart Universe: One Droplet at a Time,” in *Proceedings of 2016 International Wireless Communications and Mobile Computing Conference (IWCMC)*, Paphos, 2016, pp. 200-204.
- **K. F. Hayajneh**, **S. Yousefi** and **M. Valipour**, “Improved Finite-Length Luby-Transform Codes in the Binary Erasure Channel,” *IET Communications* 2015, vol. 9, no. 8, pp. 1122-1130, 5 21 2015.
- **K. F. Hayajneh** and **S. Yousefi**, “Overlapped Fountain Coding for Delay-Constrained Priority-Based Broadcast Applications,” in *Proceedings of 14th Canadian Workshop on Information Theory (CWIT), 2015*, St. John’s, NL, Canada, 2015, pp. 79-82.

Other contributions:

- **K. F. Hayajneh**, **J. Spencer**, and **S. Yousefi**, “Robust Quaternary Fountain Codes in AWGN Interference,” submitted to *IET Communications*, 2017.

- **K. F. Hayajneh, S. Yousefi and M. Valipour**, “Left Degree Distribution Shaping for LT Codes over the Binary Erasure Channel,” in *Proceedings of 27th Biennial Symposium on Communications (QBSC), 2014*, pp. 198-202, 1-4 June 2014.
- **K. F. Hayajneh and S. Yousefi**, “Improved Systematic Fountain Codes in AWGN Channel,” in *Proceedings of 13th Canadian Workshop on Information Theory (CWIT), 2013*, Toronto, Ontario, Canada, 2013, pp. 148-152.

1.3 Organization of the Thesis

The thesis is organised as follows: Chapter 2 gives an introduction to channel coding in general. Then, it provides more details on fountain codes and their properties. In Chapter 3, we focus on the design of robust degree distributions for fountain codes. In Chapter 4, the concept of overlapped fountain codes is proposed. The proposed overlapped fountain code is applied over a BEC. Analytical and simulation results are presented to show the improvements of the proposed scheme over conventional fountain codes. In Chapter 5, the concept of overlapped fountain codes is applied over AWGN channel. Besides, a new degree distribution is presented which outperforms the conventional one, particularly, when the idea of overlapped generations is used. Chapter 6 concludes the contributions of the thesis and presents some future extensions and directions.

Chapter 2

Channel Coding Primer

The year 1948 is considered by many as the birth year of *Information Theory* when communications and computer science pioneer Claude E. Shannon published his landmark paper [37]. Shannon stated in the paper that by proper encoding of information, errors produced by a noisy channel (or storage medium) can be decreased to any desired level as long as the information rate is less than a threshold value; this threshold is called the *capacity* of the channel [37]. Communication systems such as wireline telephone since 1876, amplitude modulation (AM) radio since early 1900's, frequency modulation (FM) radio since 1936, and pulse-code modulation (PCM) since 1937 [38] are examples of systems that predate Shannon's discovery. As a result, such communication systems were not founded on a solid theory and by design did not utilize a suitable measure of the efficiency of a communication system until then. We recall:

“The fundamental problem of communication is that of reproducing at one point either exactly or approximately a message selected at another point.”

Claude E. Shannon, 1948

In his paper, Shannon expanded on some preliminary works by Hartley [39] and discovered the fundamental laws of data compression and transmission. For instance, Shannon proved that for a noisy channel with *capacity* C , there exists a coding scheme of *transmission rate* R less than C for which the source output can be transmitted over the channel and be reconstructed with an arbitrarily small probability of error. However, Shannon did not show practical methods to encode and decode efficiently. Complexity and delay are also ignored. Many researchers have tried to find capacity-achieving codes for specific channels. For example, LT codes were shown to be asymptotically and universally optimal over the BEC [22, 23].

Definition 2.1: Code rate

For a binary code C with k source bits and a codeword of length n bits, the code rate is the ratio of the number of source bits k to the number of times that the channel has been used (i.e., n). Thus, the code rate R is given by $R = k/n$ bits/channel use.

Definition 2.2: Channel capacity

Channel capacity is the highest transmission rate at which information can be *reliably* transmitted over the channel.

Shannon's paper raised many questions such as how can we efficiently encode and decode data for *reliability*? What is the best error-correcting performance one can achieve?

In the following section, we present a primer on error control coding with an emphasis on FEC.

2.1 Forward Error Correction Codes

In FEC, adding designed redundancy into the transmitted data enables us to increase the *reliability* of transmission over a noisy channel. Since 1948, many efforts have been devoted to designing codes that achieve or approach the channel capacity in Shannon's paper.

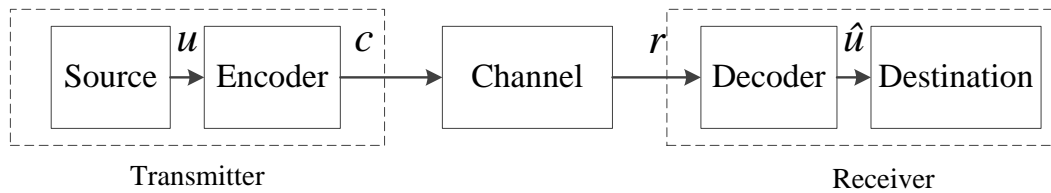


Figure 2.1: The generalized coding system block diagram.

In general, a communication system with channel coding can be simply described by Figure 2.1. Essentially, any communication system contains these five parts which are relevant to this treatise:

1. The data source which produces a sequence of messages/bits u to be delivered to the other terminal. We assume that the source generates independent and identically distributed (i.i.d) messages.
2. An encoder which performs operations on the generated messages from the data source to produce a codeword c to be transmitted over a noisy channel to improve communication *reliability*.
3. The noisy channel modeling the medium in which the encoded messages (code-words) from the transmitter are communicated to the receiver.

4. A decoder which performs the inverse operations sequence done by the encoder such to attempt to reconstruct the sequence of messages u .
5. The destination is a sink where the sequence of messages u is intended to be delivered.

Ideally, the decoded messages will be exactly the same as the source messages (i.e., $\hat{u} = u$).

Definition 2.3: Source bits

Source, information, message, or input bits represent those of a data source which are to be coded for transmission.

Definition 2.4: Encoded bits

Encoded, codeword, coded, or output bits represent those of the encoder output sent through the channel.

In the next section, various communication channels relevant to this thesis, their probability density functions, and their capacities are discussed.

2.2 Communication Channels

In general, communication channels can be categorized differently. Examples include continuous/discrete, memory/memoryless and symmetric/non-symmetric channels. In this thesis, we consider binary-input, *discrete*, *memoryless*, and *symmetric* channels. These channels can be completely characterized by their input $X = (x_1, x_2, x_3, \dots, x_n)$ and output $Y = (y_1, y_2, y_3, \dots, y_n)$ alphabets for n bits as well as their conditional probability density function (pdf) $p(Y|X)$. Because the channel is

a memoryless channel, we can describe it by single time instant of input x , output y , and conditional probability $p(y|x)$ as shown in Figure 2.2. The output codeword alphabet Y is represented either as the binary field $F_2 = \{0, 1\}$ or as $\{-1, +1\}$ if antipodal signaling is used.

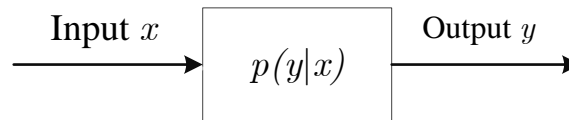


Figure 2.2: Binary input, symmetric, discrete, and memoryless channel.

Definition 2.5: Discrete channel

A discrete channel consists of an input alphabet X , output alphabet Y , and probability transition matrix $p(y|x)$.

Definition 2.6: Memoryless channel

A channel is memoryless channel if the output of the channel at any time instant depends only on the input at that time instance and is conditionally independent of previous channel inputs and outputs.

Definition 2.7: Symmetric channel

A binary memoryless channel is a symmetric channel if and only if $p(y = 1|x = 1) = p(y = 0|x = 0)$ and $p(y = 1|x = 0) = p(y = 0|x = 1)$.

Although we have contributions on non-binary codes [40], we are only interested in binary channels in this thesis. These include BEC, BSC, and binary-input AWGN (BIAWGN) channel.

2.2.1 Binary Erasure Channel (BEC)

The BEC, which is a discrete binary-input memoryless channel, is the simplest communication channel where the receiver either receives the transmitted data/message perfectly or does not receive it at all (the data is erased). The BEC was introduced by Elias in 1955 [41]; it can be characterized by its symmetric erasure probability ϵ and it is commonly denoted by $\text{BEC}(\epsilon)$ as shown in Figure 2.3. The inputs to the channel are the binary bits $\{1, 0\}$, and the output alphabet is ternary $\{1, e, 0\}$ where e represents an erasure. Therefore, a transmitted bit is either received perfectly with probability $(1 - \epsilon)$, or it is erased with probability ϵ . Elias found the capacity of the BEC as [41, 42]:

$$C = 1 - \epsilon \text{ bits/channel use.} \quad (2.1)$$

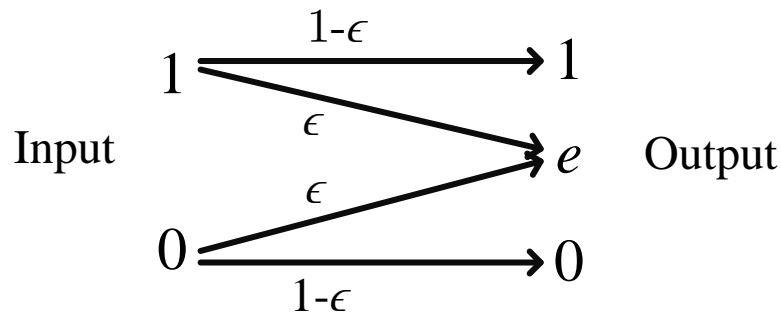


Figure 2.3: A binary erasure channel (BEC) with erasure probability ϵ .

Despite its simplicity, many real communication scenarios can be modeled by the BEC. For instance, on the Internet, files are transmitted in multiple small packets which are either received correctly or declared erased. Representing packets by bits for simplicity, we can resort to a BEC model. The erasure declaration is due to

two main reasons: buffer overflows occurring in the network or one or more detected bit errors at the receiver which cannot be corrected due to lack of FEC capability. Another example can be seen in data storage problems where corrupt data can be considered erased [43].

2.2.2 Binary Symmetric Channel (BSC)

The second basic channel is the BSC where the transmitted bit is either received correctly with probability $(1 - p)$ or flipped with probability p as shown in Figure 2.4. The capacity of this type of channel is completely characterized by its cross-over probability p and it is given by [42]:

$$C = 1 - p \log_2(p) - (1 - p) \log_2(1 - p) \text{ bits/channel use.} \quad (2.2)$$

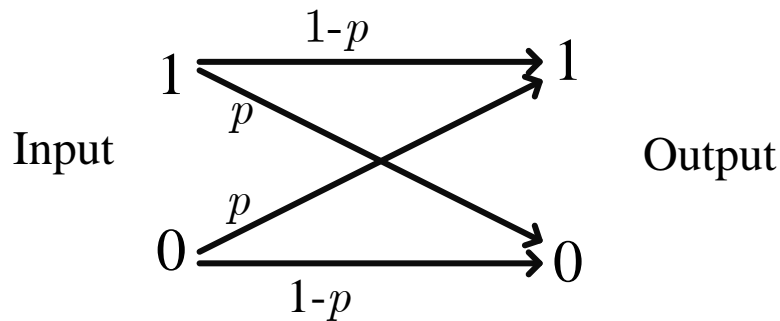


Figure 2.4: A binary symmetric channel with cross-over probability p .

2.2.3 Binary-Input Additive White Gaussian Noise (BIAWGN) Channel

AWGN channel is the most studied channel model. BIAWGN channel is an AWGN channel with a binary input $\{+1, -1\}$ and the set of real numbers as an output (continuous output). An instant transmission through a BIAWGN channel can be expressed as

$$Y = X + Z, \quad (2.3)$$

where Y , X are output and input sequences, respectively, and Z is a sequence of i.i.d Gaussian noise samples with zero mean, and variance $\sigma^2 = \frac{N_0}{2}$, where N_0 is the double-sided power spectral density.

Owing to the fact that the channel is memoryless, we can express it by single time instant as shown in Figure 2.5. In addition, the conditional pdf $p(y|x)$ can be written as

$$p(y|x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y-x)^2}{2\sigma^2}}. \quad (2.4)$$

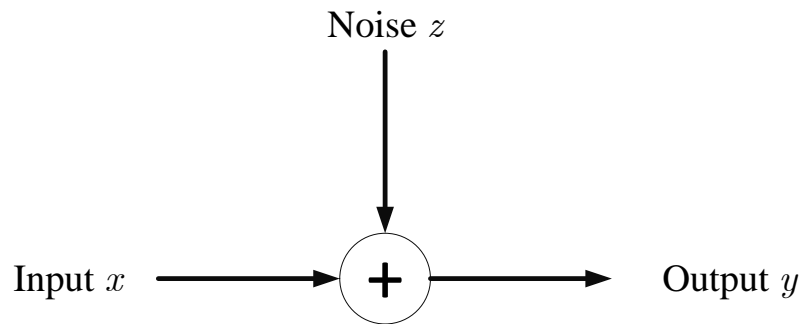


Figure 2.5: AWGN channel.

The capacity of an AWGN channel with power constraint S and noise variance σ^2 is given by

$$C = \frac{1}{2} \log_2(1 + S/\sigma^2) \text{ bits/channel use}, \quad (2.5)$$

where S/σ^2 is the signal-to-noise ratio (SNR) [14].

2.3 Channel Codes

In this thesis, our focus is on linear binary codes. Such codes can be block or convolutional type. Algebraic codes have received a great deal of attention since their inception in the works of Richard Hamming in the middle of last century [44]. Our attention here will rather be on graph-based codes. The explosive popularity of this family of codes is primarily owed to the invention of Turbo codes and turbo decoding concept [45]. That, in turn, resurrected an elegant family of codes referred to as LDPC codes invented previously by Robert Gallager [46].

In what follows, we briefly introduce block and convolutional codes followed by LDPC and closely related low-density generator matrix (LDGM) and rateless codes.

2.3.1 Block Codes

Block codes are FEC codes used to both detect and correct errors. They accept a block of k input bits and generate a block of n output bits at the transmitter. Block codes are also sometimes referred to as (n, k) codes. For k (input) source bits, there are a total of 2^k distinct messages (in binary case). After encoding, each of those messages will have length n bits, which will be referred to as codewords in the rest of this thesis. Thus, block codes add $(n - k)$ redundant bits to the k input stream to produce an n output bitstream. The code rate of a block code is simply the ratio

$$R = k/n.$$

Various *linear block codes* have been developed over the years, some of which include repetition codes, Hamming codes [44], Bose-Chaudhuri-Hocquenghem (BCH) codes [47, 48], Reed-Solomon (RS) codes [49], and low-density parity-check (LDPC) codes [46].

Definition 2.8: Linear block codes

A block code C of length n and 2^k codewords is a linear (n, k) code if and only if the sum of two codewords is also a codeword; i.e., a linear code C must satisfy the following condition:

$$\forall c_1 \in C \text{ and } \forall c_2 \in C \Rightarrow c_1 + c_2 \in C.$$

Generally, linear block codes are commonly represented using either *matrix algebra* or *graph theory*. In matrix representation, a linear block code can be defined by its generator matrix (G) and/or its parity-check matrix (H). The generator matrix $G_{k \times n}$ is a $k \times n$ matrix which carries information about the relationship between the input (k source bits) and the output (n encoded bits). The G matrix is utilized in the encoding process. The parity check matrix $H_{(n-k) \times n}$ is an $(n - k) \times n$ matrix which represents the relationship between the output (n encoded bits) and parity check bits ($n - k$ bits). The H matrix is typically used in the decoding process.

Alternatively, a graphical representation is possible using the *bipartite/Tanner graph* [16, 50] which can be associated with the generator matrix G or the party-check matrix H of the code.

Definition 2.9: A graph

A graph \mathbf{G} consists of the node/vertex set \mathbf{V} and edge set \mathbf{E} such that each edge in \mathbf{E} can be identified with a pair of nodes from \mathbf{V} .

Definition 2.10: Bipartite graph

A graph \mathbf{G} is called bipartite/Tanner graph if its nodes \mathbf{V} can be grouped into two disjoint types \mathbf{V}_1 and \mathbf{V}_2 such that each edge in \mathbf{E} connects a node in \mathbf{V}_1 with a node in \mathbf{V}_2 .

2.3.2 Convolutional Codes

In 1955, Elias introduced convolutional codes by adding memory to the encoding process [41, 51]. Convolutional codes are one of the most widely used channel codes in practical communication systems. These codes convert the entire data stream into one single codeword. The encoded bits depend not only on the current k input bits but also on several consecutive past input bits [16, 52].

Commonly, convolutional codes are specified by three parameters (k, n, m) which represent the number of input bits, the number of output bits, and the number of memory registers, respectively. Figure 2.6 shows an example of a rate $R = 1/3$ convolutional encoder with memory order of $m = 3$.

In the literature, we find several extensions to convolutional codes. Amongst the popular extensions are Trellis coded modulation (TCM) [53–55] and Turbo codes [45]. From its name, TCM adds redundancy by combining coding and modulation. The main advantage of using TCM is that the data rate is not decreased for a fixed channel bandwidth. Turbo codes use two or more convolutional encoders working in parallel to generate the encoded sequence. The two encoders are separated by

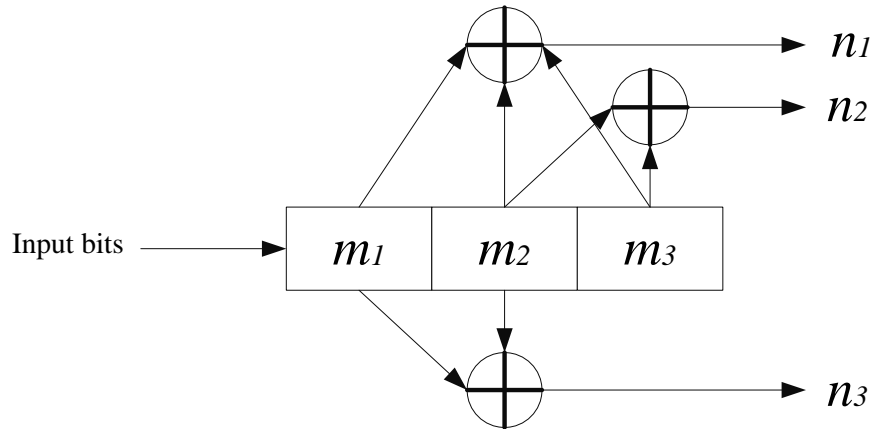


Figure 2.6: A convolutional encoder with memory order of $m = 3$ and code rate $R = 1/3$.

a random interleaver which changes the order of the source bits such that the two encoders work on different source sequences, one of which is the permutation of the other. Turbo code performance was shown by Berrou et al. to approach the channel capacity of an AWGN channel [45].

However, a major disadvantage of convolutional codes is that the decoding computational complexity increases exponentially as the length of the code increases linearly. In addition, convolutional codes have relatively high latency, which may not be suitable for some applications.

2.3.3 Low-Density Parity-Check (LDPC) Codes

LDPC codes were originally invented by Gallager in his PhD thesis in the early 1960's [46, 56]. The codes were not popular or practically feasible due to their high

computational complexity. In the middle of 1990's, with the availability of high-performance processors, the LDPC codes resurged by the independent efforts of Sipser and Spielman [57, 58], Wiberg et al. [59, 60], and MacKay and Neal [61–63]. LDPC codes are *sparse* codes that can be specified by a parity-check matrix H which contains mostly 0's and relatively few 1's (*low density*). The sparsity of the H matrix is an essential property that allows for the algorithmic efficiency of LDPC codes.

Definition 2.11: Low-density matrix

A $k \times n$ matrix is a low-density if kn is tending to infinity and the number of nonzero elements (i.e., ones) is less than $\max(k, n)$. Also, it is referred to as a sparse matrix.

There are two types of LDPC codes: regular and irregular LDPC codes. An LDPC code is regular if the columns and rows of H matrix have uniform weight; i.e., the number of 1's in each column w_c is the same and the number of 1's in each row w_r is the same. Irregular LDPC code relaxes these conditions; i.e., w_c and w_r are different for different columns and rows, respectively.

Like any other block codes, LDPC codes can be represented using a *Tanner graph*. The Tanner graph of LDPC codes contains n variable nodes representing encoded bits and m check nodes representing the parity checks. Figure 2.7 shows a Tanner graph of an example of regular LDPC code. The circles represent $n = 6$ encoded bits while the squares represent $m = 4$ parity checks.

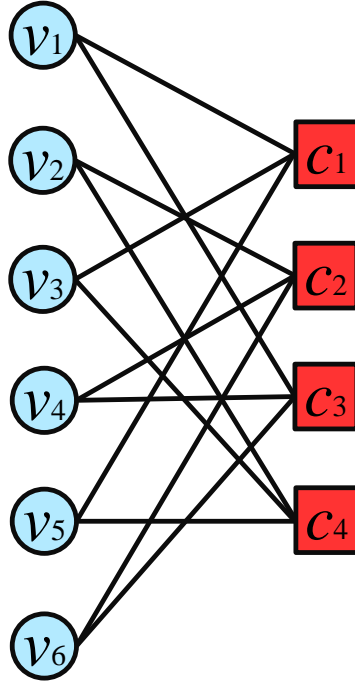


Figure 2.7: Tanner graph of a regular LDPC code with $w_c = 2$ and $w_r = 3$. The circles represent $n = 6$ encoded bits while the squares represent $m = 4$ parity checks.

The parity check matrix H corresponding to the Tanner graph in Figure 2.7 is given by:

$$H = \begin{pmatrix} 0 & 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix}. \quad (2.6)$$

It is noted that the H matrix in Equation (2.6) is regular with $w_c = 2$ and $w_r = 3$.

To generate a codeword using LDPC codes, one must obtain the generator matrix G associated with H . Firstly, $H_{m \times n}$ matrix should be written in the form

$[P_{m \times (n-m)}^T \mid I_m]$ using Gaussian elimination, where I_m is the identity matrix of order m . Then, $G_{k \times n}$ matrix can be calculated as $G = [I_{(n-m)} \mid P_{(n-m) \times m}]$, where $k = n - m$ is the number of source bits. The number of operations sufficient for performing the encoding depends on the *Hamming weights* of the vectors in the G matrix. Since the G matrix is not sparse in general, the cost of the encoding process is proportional to kn . For fixed-rate codes, the cost is proportional to n^2 resulting in high complexity and high latency as well, particularly, as the code length increases. As a result, LDPC codes are not suitable for delay-sensitive and/or real-time applications [43].

Definition 2.12: Hamming weight

The Hamming weight of a vector \mathbf{v} of length k is the number of nonzero components of \mathbf{v} .

Based on the channel type and application requirements, several decoding methods can be used to reconstruct the source data. These methods include, but not limited to, majority vote decoding [64], bit flipping decoding [16], and *belief propagation* (BP) decoding (also known as peeling decoder or sum-product algorithm) [65–67].

Definition 2.13: Belief propagation (BP)

The BP algorithm is an iterative decoding technique that runs on the Tanner graph of the code. In the BP, the messages are passed from encoded bits to source bits and from source bits to encoded bits. The BP decoder can be used over the BEC as well as noisy channels such as AWGN channel.

2.3.4 Low-Density Generator-Matrix (LDGM) Codes

In contrast to LDPC codes, LDGM codes are constructed using sparse generator matrix G [68, 69]. One particular examples of LDGM codes are concatenated single parity check codes [70]. Since the generator matrix G is a sparse matrix, the LDGM encoder has low complexity. However, the decoding complexity is high compared with decoding complexity of LDPC codes. To reduce the decoding complexity of LDGM codes, systematic LDGM codes are used in which the generator matrix G is given by $G_{k \times n} = [I_k \mid P_{k \times (n-k)}]$, where P is a sparse matrix. As a result, the parity-check matrix of LDGM codes, $H_{(n-k) \times n} = [P_{(n-k) \times (k)}^T \mid I_{(n-k)}]$, is also sparse.

The decoding process of systematic LDGM codes is equivalent to that of LDPC codes. Thus, the decoding complexity of systematic LDGM codes is low. The reduction of the encoding and decoding complexities in LDGM codes comes at the cost of error correction capability. In fact, MacKay showed that LDGM codes are asymptotically *bad codes* [63].

Definition 2.14: Bad codes

Bad codes are a family of codes that can achieve arbitrarily small probability of error only by decreasing the code rate to zero.

2.3.5 Rateless Codes

There has been considerable amount efforts in the literature to enhance the performance of LDPC codes, and linear block codes in general. For example, the concept of irregular LDPC codes was introduced to perform at rates extremely close to the

capacity of an AWGN channel [71, 72]. Also, different decoders were designed to improve the decoding performance. For instance, BP decoder [65], min-sum algorithm (also known as BP-based algorithm) [73], linear programming (LP) decoder [74], etc. Although these works improve the performance of regular LDPC code, they are not as efficient as rateless codes for broadcast applications.

In LDPC codes, the code rate must be fixed at the design stage. For time-varying channel, the code rate is either too high or too low. If the code rate is low, excessive redundant information will be received. Thus, the channel bandwidth is wasted. If the code rate is high, *reliable* communication between the transmitter and the receiver is not guaranteed. In addition, in broadcast applications, each receiver is connected to the transmitter with different channel characteristics. Thus, the transmitter either uses LDPC codes with different code rates (which is not broadcasting) or uses LDPC codes that are designed for the worst channel. In both cases, the solution is not good enough, and the transmitter can do better. Another drawback of using LDPC codes is that in some applications, the feedback channels are too costly or infeasible. Thus, the transmitter cannot predict *channel state information (CSI)*. In this case, the transmitter tries to guess which code rate is good. To solve the drawbacks/shortcomings of LDPC codes (and fixed-rate codes in general), rateless codes have been introduced.

Definition 2.15: Channel state information (CSI)

Channel state information is information which represents the state of a communication link between the transmitter to the receiver, namely channel gain. The information characterizes how the signal propagates from the transmitter to the receiver.

Unlike fixed-rate codes, rateless codes can adapt their rate on-the-fly. They are rateless in the sense that a theoretically endless stream of output messages can be generated from the source messages. There are two basic approaches of rateless codes, puncturing a mother code and fountain coding. The first idea of puncturing a mother code was proposed by Mandelbaum in 1974 [75]. In his work, Mandelbaum transmitted a punctured codeword initially using RS codes. If the receiver cannot decode the source message, it asks the transmitter to send another increment of redundancy. This process is continued until the source messages are fully decoded. This approach is also referred to as *rate-compatible* approach. The rate-compatible approach has been applied to different channel codes. For example, rate-compatible LDPC codes [76, 77], rate-compatible punctured convolutional codes [78, 79] and rate-compatible Turbo codes [80, 81].

However, a puncturing approach has mainly three shortcomings:

1. The encoding and decoding complexities are high. For example, if the channel is *perfect*, the encoder generates a low rate code while the decoder decodes the same low rate code.
2. Only limited set of code rates are allowed depending on the size of the incremental redundancy.
3. The performance of this approach mainly depends on the design of the mother code. If the mother code is suboptimal, the punctured high rate code may have very bad performance.

Definition 2.16: Perfect channel

A channel is a perfect (also known as noiseless) channel if and only if its output Y equals to its input X so that $p(Y|X) = 1$. Therefore, the perfect channel has no noise and no uncertainty.

The second approach is the concept of fountain coding [20, 21, 65, 66]. Fountain codes are considered as a type of dynamic LDGM codes. A full review on fountain codes is given in the next section.

2.4 Fountain Codes

In the fountain codes, the encoder looks like a (digital) fountain that generates a limitless sequence of encoded bits (or water drops) from the source bits. Each encoded bit has a partial information about the original source bits. Any receiver that wishes to receive the source data holds a bucket under the fountain and collects drops of water (encoded bits) until the bucket is full. The size of the bucket must be a little larger than the number of source bits in order to have a full recovery. The receiver can start recovering the source bits once the bucket becomes full. It does not matter how far the receiver from the fountain (the channel characteristics) or which specific encoded bits have been collected (i.e., the order of encoded bits does not matter). Any subset of encoded bits with a size slightly larger than the size of the original source bits provides enough information for the receiver to decode the whole source bits.

In general, the fountain encoder chops the source data K into g generations, each generation is of size k packets, and each packet is of size l bits. For simplicity, we assume that each packet contains only one bit. Therefore, packets and bits will be

used interchangeably throughout the thesis.

Many works have been done as first steps towards realizing practically achievable fountain codes [82–85] which generally suffer from a few drawbacks such as:

1. The encoding and decoding times are slow at large block sizes, effectively limiting k to small values for practical applications.
2. As the number of receivers increases or the rates of packet loss increases, the number of retransmissions increases as well.
3. A receiver must receive the encoded packets in the same order as they have been generated. Thus, the receiver cannot start decoding process anytime.

Next, some practical realizations of fountain codes are introduced such as LT codes and Raptor codes. Before we discuss such codes, we are going to show a category of codes that is considered as the first step towards LT codes. This category is referred to as random linear fountain (RLF) codes.

2.4.1 Random Linear Fountain (RLF) Codes

RLF codes are the simplest example of fountain codes [22, 64]. For a generation of size k and source bits $\{u_1, u_2, \dots, u_k\}$, an encoded bit $c_i, i = 1, 2, \dots, n$, where n is the total number of generated encoded bits, is generated using the following equation:

$$c_i = \sum_{j=1}^k u_j \cdot G_{ji}, \quad (2.7)$$

where the sum is the bitwise sum modulo-2 (exclusive-or, XOR) of the source bits for which $G_{ji} = 1$. The matrix $G_{k \times n}$ is $k \times n$ generator matrix containing 0's and 1's,

uniformly at random. For example, a 4×5 generator matrix G is randomly generated as

$$G_{4 \times 5} = \begin{pmatrix} 1 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 \end{pmatrix}.$$

Note that the number of columns can be extended to as many as needed. The decoder can reconstruct the generator matrix $G_{k \times n}$ in different ways. For example, the encoder and decoder can be synchronized to have an identical pseudorandom number generator, or the information (number of ones and their positions) of each encoded bit/packet can be carried in its header.

Depending on the total number of encoded bits n , two cases can be identified:

1. The total number of encoded bits is less than the size of the generation (i.e., $n < k$). In this case, the receiver cannot recover the k source bit because there is not enough information.
2. The total number of encoded bits is greater than or equal to the size of the generation (i.e., $n \geq k$). In this case, if the generator matrix $G_{k \times n}$ contains an invertible $k \times k$ matrix, the receiver can recover the k source bits by computing $G_{k \times k}^{-1}$ and then using the following equation

$$u_j = \sum_{i=1}^n c_i \cdot G_{ij}^{-1}. \quad (2.8)$$

The probability that a random $G_{k \times n}$ is invertible depends on the linear independence of its columns. If there are k columns in the $G_{k \times n}$ that are *linearly independent*,

then the matrix is an invertible matrix. Firstly, let's assume $n = k$. Thus, the probability that the first column is all-zero vector is $(\frac{1}{2})^k$ which means that the probability that the column is different from the all-zero vector is $(1 - (\frac{1}{2})^k)$. Then, the probability that the second column is different from the first column as well as the all-zero vector is given by $(1 - (\frac{1}{2})^{k-1})$. Next, the probability that the third column is different from the first two columns as well as the all-zero vector is $(1 - (\frac{1}{2})^{k-2})$ and so forth. Thus, the probability that $G_{k \times k}$ is invertible, P_{invert} , is equal to

$$P_{invert} = \left(1 - \left(\frac{1}{2}\right)^k\right) \times \left(1 - \left(\frac{1}{2}\right)^{k-1}\right) \times \left(1 - \left(\frac{1}{2}\right)^{k-2}\right) \times \cdots \times \left(1 - \frac{1}{4}\right) \times \left(1 - \frac{1}{2}\right). \quad (2.9)$$

For generation size k greater than 10, $P_{invert} \approx 0.289$ which is very low. This is equivalent to the probability of successful decoding $p = 1 - \delta$, where δ is the failure probability.

Definition 2.17: Linearly independent

A finite set $\{c_1, c_2, \dots, c_k\}$ of vectors is said to be linearly independent if there are no scalars $s_1, s_2, s_3, \dots, s_k$ not all of which are 0, such that $s_1c_1 + s_2c_2 + \dots + s_kc_k = 0$. In other words, any one of the vector c 's cannot be written as a linear combination of the other vectors.

If n is greater than k ; i.e., $n = k(1 + \varepsilon)$, where ε is the *overhead*, the probability that the random matrix $G_{k \times n}$ contains an invertible $G_{k \times k}$ is given by $P_{invert} = 1 - \delta$. Therefore, the failure probability δ is upper bounded by

$$\delta(\varepsilon) \leq 2^{-k\varepsilon}. \quad (2.10)$$

Definition 2.18: Code overhead

Code overhead (or simply overhead) is defined as the ratio between the extra encoded bits required to recover the original source bits and the length of source bits k . The overhead is given by

$$\varepsilon = \frac{n - k}{k} = R^{-1} - 1, \quad (2.11)$$

where n is the total number of encoded bits and $R^{-1} = 1/R$ is the inverse code rate.

Figure. 2.8 shows the upper bound on the failure probability versus the overhead at generation sizes of $k = 128$, $k = 256$, and $k = 512$. As can be seen in the figure, the failure probability is decreased (almost to zero) as the overhead increases. For instance, at overhead of $\varepsilon = 0.1$ (code rate $R = 0.91$), an RLF code of generation size $k = 128$ has an upper bound of $\delta = 1.4 \times 10^{-4}$. However, the upper bound is decreased to $\delta = 2.75 \times 10^{-12}$ when the overhead increased to $\varepsilon = 0.3$ (code rate $R = 0.77$).

Another parameter that affects the performance of the code is the probability that a source bit is not covered by any encoded bit in the encoding process. This is equivalent to having a row of all zeros in the generator matrix $G_{k \times n}$ after generating n encoded bits. The probability is given by

$$P_{zero-row} = \left(1 - \frac{1}{k}\right)^n \simeq e^{-n/k}. \quad (2.12)$$

Therefore, the expectation of having a source bit without any connection is $ke^{-n/k}$. To have a *reliable* decoder, $P_{zero-row}$ must converge to zero. Therefore, the error

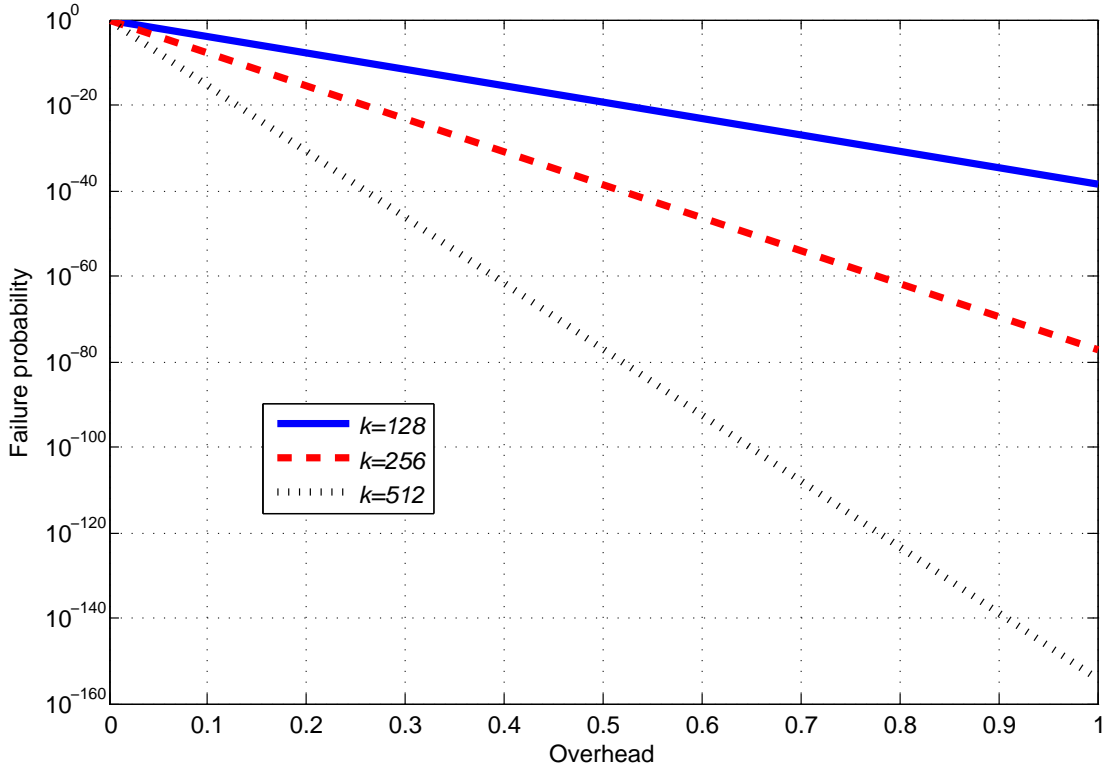


Figure 2.8: Upper bound on failure probability for RLF codes versus the overhead ε at generation sizes of $k = 128$, $k = 256$, and $k = 512$.

probability must be at most $1/k^c$; i.e., $P_{zero-row} \leq 1/k^c$ for some constant $c \geq 1$.

This shows that

$$n \geq ck \ln(k). \quad (2.13)$$

Figure. 2.9 illustrates the lower bound on the required number of encoded bits n versus generation size k at different c values. As can be seen, smaller values of c have better performance (less number of encoded bits). For instance, at generation size $k = 128$ and $c = 1$, the number of encoded bits must be at least $n = 621$ (code rate $R = 0.21$). However, at least $n = 1863$ encoded bits ($R = 0.07$) are needed if $c = 3$.

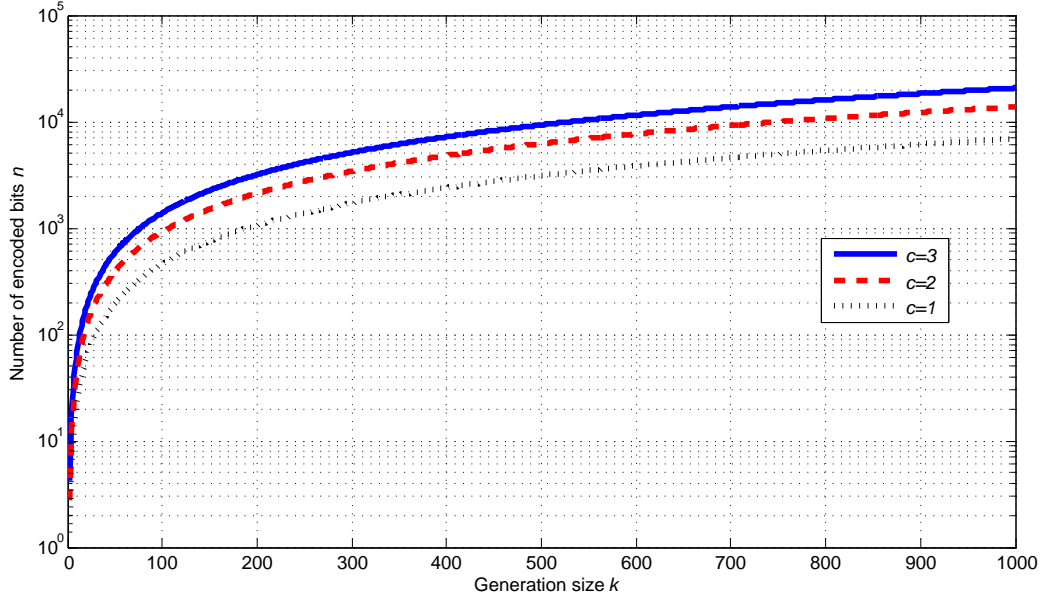


Figure 2.9: Lower bound on the required number of encoded bits n in RLF codes versus generation size k at different values of c .

Consequently, we can set an upper bound on the code rate R using Equation (2.13)

as

$$R \leq \frac{1}{c \ln(k)}. \quad (2.14)$$

That means as the generation size k increases, the code rate R decreases.

In conclusion, if the overhead of the RLF codes increases, the probability of successful decoding increases rapidly; this is the main advantage of RLF codes. In contrast, the encoding and decoding complexities of RLF codes are very high. Namely, the encoding process requires $k/2$ bit operations per encoded bit (because it is random with probability $1/2$ for choosing bit 1) whereas the decoding process requires a total of $k^3 + k^2/2$ bit operations; k^3 for finding the inverse matrix and $k^2/2$ is the cost of multiplying the number of encoded bits by the inverse matrix.

2.4.2 Luby-Transform (LT) Codes

To reduce the encoding and decoding complexities of RLF codes, Luby introduced LT codes [23]. LT codes were the first practical realization of fountain codes over the BEC. In LT codes, the encoder can generate as many encoded bits as needed from the k source bits. The LT encoder uses a *degree distribution* to decide how many ones are required in each column in the generator matrix. Therefore, RLF codes are a special type of LT codes in which the degree distribution is a uniform distribution.

By using a designed degree distribution, the generator matrix of an LT code can be a *low-density matrix*. Therefore, the encoder achieves very low encoding complexity. In addition, the decoder does not need to invert the generator matrix; different simpler methods can be used. Some of these methods will be discussed later in this section.

Definition 2.19: Degree

In the graphical representation of codes, the degree of a node is the number of edges/connections to other nodes. For example, a source bit degree is the number of edges that connect this source bit to the encoded bits. Also, the encoded bit degree is the number of edges that connects this encoded bit to the source bits. In the matrix representation of codes, degrees of source bits are the Hamming weight of rows of the generator matrix G whereas degrees of encoded bits are the Hamming weight of columns of G .

Definition 2.20: Degree distribution

Over a Tanner graph corresponding to a generator matrix G , the degree distribution of source (encoded) bits is given by

$$\gamma(x) = \sum_{i=1}^{d_{max}} \gamma_i x^i,$$

where γ_i and d_{max} are the fraction of source (encoded) bits of degree i and the maximum source (encoded) degrees, respectively.

Encoding and decoding processes of LT codes are very simple and will be described in the following two subsections.

I. Encoding Process

The encoding process is very simple and straightforward. For a generation of size k and source bits $\mathbf{u} = (u_1, u_2, \dots, u_k)$, an encoded bit c_i , $i \geq 1$ is generated using Algorithm 1.

Algorithm 1 LT Encoding Process

1. Choose a degree d from a predesigned degree distribution.
2. Choose, uniformly at random, d source bits.
3. Set c_i to the XOR of the chosen d source bits.

By repeating Algorithm 1, the encoder generates potentially a limitless number of encoded bits. Algorithm 1 is tantamount to defining a graph (or generator matrix) connecting the source bits to the encoded bits. Another way to define the encoded bits is by using the generator matrix \mathbf{G} . For example, the encoded bits $\mathbf{c} = (c_1, c_2, \dots, c_n)$,

where n is the total number of encoded bits, is given by

$$\mathbf{c} = \mathbf{u} \cdot \mathbf{G}, \quad (2.15)$$

where $\mathbf{G}_{k \times n}$ is a $k \times n$ generator matrix.

LT encoder can be best described using the Tanner graph as shown in Figure 2.10 in which left nodes (circles) show the source bits while right nodes (squares) show the encoded bits.

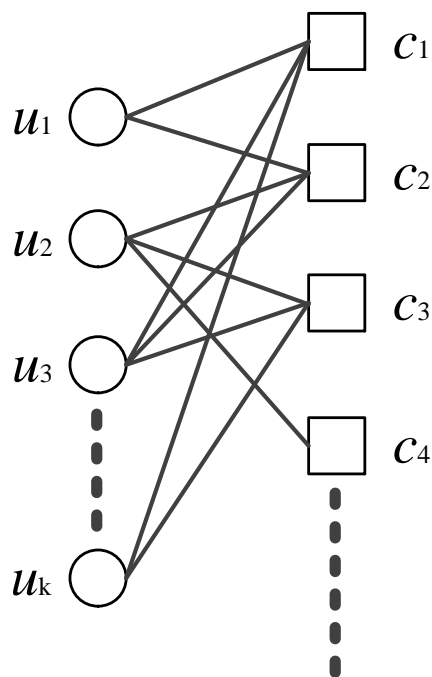


Figure 2.10: Tanner graph of an LT code. Circles represents the source bits and squares represents the encoded bits

The generator matrix corresponding to the Tanner graph in Figure 2.10 is given

by

$$G_{k \times \dots} = \begin{bmatrix} 1 & 1 & 0 & 0 & \dots \\ 0 & 1 & 1 & 1 & \dots \\ 1 & 1 & 1 & 0 & \dots \\ \vdots & \vdots & \vdots & \vdots & \ddots \\ 1 & 0 & 1 & 0 & \dots \end{bmatrix}.$$

It is noted that the positions of element 1 in a column j indicate the source bits that participate in generating the encoded bit c_j . In other words, when $G_{ij} = 1$, the source bit u_i participates to generate the encoded bit c_j .

II. Decoding Process over the BEC

Over BEC channels, any receiver who wishes to recover the source data, starts collecting the encoded bits. Once a receiver collects a sufficient number of encoded bits n , where n is slightly larger than k , it can recover the k source bits. In general, the receiver can extract the inverse matrix of \mathbf{G} for decoding process. However, this method has high decoding complexity especially for large generation size k as we show in the case of RLF codes in Section 2.4.1. There are many other methods that benefit from the sparsity of the graph. These methods fit under the umbrella of *message passing algorithms* (MPA). One of the most well-known MPA decoders is the BP decoder.

The BP decoding algorithm over the BEC is fully described in Algorithm 2. Note that steps 1 through 3 are referred to as full *decoding iteration*.

Algorithm 2 BP Decoding Process over a BEC

1. Identify all degree-one encoded bits and add them to storage referred to as *ripple*. If the ripple becomes empty before all the k source bits are recovered, the decoder decides decoding failure and it requires more encoded bits.
2. Choose one encoded bit c_j from the ripple, and then its neighbor source bit u_i is immediately recovered (i.e., $u_i = c_j$). Then, remove the edge that connects u_i to c_j .
3. Update all the neighbor encoded bits of the recovered source bit u_i in step (2); i.e., for each neighbor encoded bit c_l , the current value is XORed with u_i , $c_l = c_l \oplus u_i$. Then, remove all edges of the recovered source bit u_i .
4. Repeat (1)-(3) until all k source bits are recovered.

Definition 2.21: Ripple

There are two relevant definitions of the ripple in the literature: 1) Ripple is the set of encoded bits each of which is connected to only one source bit [24]. 2) Ripple is the set of source bits each of which is connected to at least one encoded bit of degree one [23]. Throughout the thesis, we use the first definition.

Definition 2.22: Decoding iteration

Each decoding iteration (also known as decoding step) constitutes any number of messages passing from encoded bits to source bits and any number of messages sent back to the encoded bits.

An example of a BP decoder is given in Figure 2.11. In the example, there are $k = 4$ source bits and the decoder receives $n = 4$ encoded bits with $\{c_1, c_2, c_3, c_4\} = \{1, 0, 1, 0\}$. In Figure 2.11 (a), the decoder reconstructs the graph which means that the decoder knows the components of each encoded bit as we discussed in Section 2.4.1. Then, the decoder finds degree-one encoded bit which is c_1 in this example as shown in Figure 2.11 (b). The source bit which is connected to c_1 is recovered (i.e.,

$u_1 = c_1 = 1$), and the edge between c_1 and u_1 is removed as shown in Figure 2.11 (c), this is referred to as a forward iteration. In the backward iteration, c_2 is updated (i.e., $c_2 = c_2 \oplus u_1 = 0 \oplus 1 = 1$), and edge between c_2 and u_1 is removed as shown in Figure 2.11 (d). The first decoding iteration/step is done.

Again, the decoder searches for degree-one encoded bits. Now, c_2 is degree-one encoded bit; therefore, its neighbor source bit u_2 is recovered ($u_2 = c_2$) as shown in Figure 2.11 (e). In the backward iteration, neighbor encoded bits of u_2 (i.e., c_3 and c_4) are updated and the edges of u_2 are removed as shown in Figure 2.11 (f). Finally, c_3 and c_4 are now degree-one encoded bits (Figure 2.11 (g)). Consequently, u_3 and u_4 , which are the neighbor source bits of c_3 and c_4 , respectively, are recovered (Figure 2.11 (h)). Finally, the edges of u_3 and u_4 are removed as shown in Figure 2.11 (i). Thus, the decoder succeeds to recover the source bits $\{u_1, u_2, u_3, u_4\} = \{1, 1, 0, 1\}$.

At any decoding step, if the decoder fails to find degree-one encoded bits (empty ripple) before it recovers all the k source bits, it decides a decoding failure and needs more overhead (encoded bits). The decoding complexity is in order of $O(\ln(k))$ per encoded bit.

As shown in the encoding and decoding processes of LT codes, the encoding and decoding complexities are determined by the edge complexity. Therefore, one can decrease/increase the complexity by changing the connectivity (degree distribution) of the graph. This comes at the cost of probability of successful decoding.

In conclusion, designing an excellent degree distribution is tantamount to finding a better complexity-error probability trade-off. In the next subsection, we discuss the degree distribution in details.

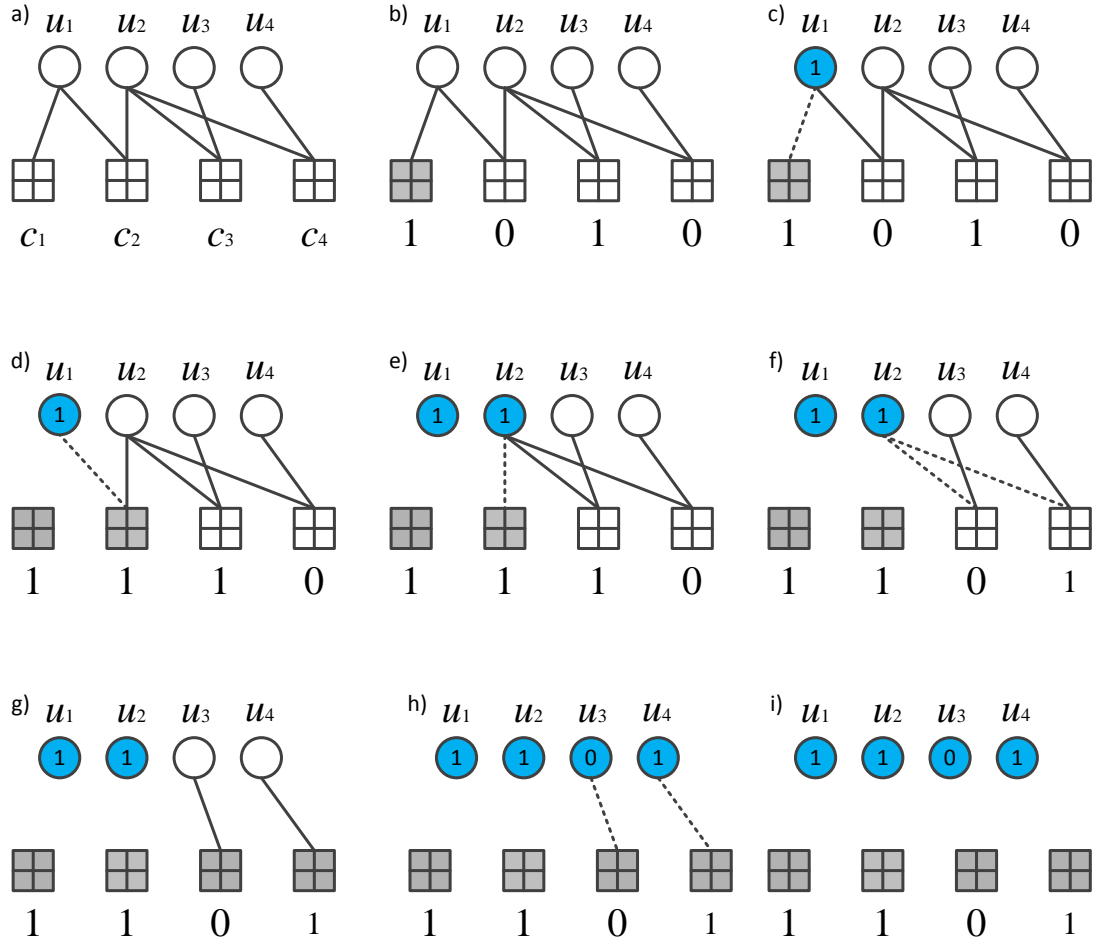


Figure 2.11: An example of a BP decoder over a BEC. There are $k=3$ source bits and the decoder receives $n = 4$ encoded bits.

III. Degree Distribution

The degree distribution design is the most important part in designing LT codes. One can think designing the degree distribution is equivalent to designing a good ripple size. On the one hand, ripple size must not be zero during the decoding process; otherwise the decoder fails to recover the whole source bits. On the other hand, large

ripple size means high redundant encoded bits (i.e., waste of bandwidth).

In addition, for connectivity purposes, some encoded bits must have high degrees to ensure that no single source bit is left without a connection whereas many other encoded bits must have low degrees so that the decoding process can get started and keep going. Also, the low degrees are needed to ensure the encoding and decoding complexities are small [22]. In the following, some of the well-known degree distributions are presented.

All-at-Once Distribution

The first distribution we can think about is the all-at-once distribution, where each encoded bit is of degree one (i.e., each encoded bit is connected to only one source bit). In other words, all-at-once distribution is a rateless random repetition scenario.

The distribution coefficients ρ_d are given as

$$\rho_d = \begin{cases} 1, & \text{if } d = 1, \\ 0, & \text{if } d = 2, 3, \dots, k. \end{cases} \quad (2.16)$$

The distribution is used in theory to design better distributions despite being impractical. In fact, it requires a high number of encoded bits to cover all the source bits (the code rate is low). According to the bins and balls problem, it needs at least $k \cdot \ln(k)$ encoded bits to ensure that all the source bits are covered once, at least.

Ideal Soliton Distribution

Theoretically, the decoding process needs only one degree-one encoded bit at each iteration; which means that at each decoding step, the current degree-one encoded

bit is processed while another degree-one encoded bit is added to the ripple. To do so, Luby introduced ideal soliton distribution (ISD) in his original paper [23]. The ISD needs only k encoded bits to recover the k source bits (optimal in terms of code rate). The ISD is given by

$$\rho_d = \begin{cases} \frac{1}{k}, & \text{if } d = 1, \\ \frac{1}{d(d-1)}, & \text{if } d = 2, 3, \dots, k. \end{cases} \quad (2.17)$$

In ISD, the expected number of degree-one encoded bits (i.e., *ripple* size) at each decoding step is exactly one. Although this distribution is theoretically optimal, it works poorly in practice due to the fluctuation of the ripple size during the decoding process.

Robust Soliton Distribution

Luby introduced another degree distribution called robust degree distribution (RSD). The RSD increases the expected number of degree-one from 1 (in ISD) to

$$S = c \cdot \ln(k/\delta) \cdot \sqrt{k}, \quad (2.18)$$

where c and δ are positive constants, and parameter δ denotes an upper bound on failure probability [23].

To introduce the RSD, first define

$$\tau_d = \begin{cases} \frac{S}{k} \frac{1}{d}, & \text{if } d = 1, 2, \dots, (\lfloor \frac{k}{S} \rfloor - 1) \\ \frac{S}{k} \ln(S/\delta), & \text{if } d = \lfloor \frac{k}{S} \rfloor \\ 0, & \text{if } d > \lfloor \frac{k}{S} \rfloor, \end{cases} \quad (2.19)$$

then add the ρ_d from the ISD (Equation 2.17) to τ_d and normalize to obtain the RSD, Ω_d :

$$\Omega_d = \frac{\rho_d + \tau_d}{\beta}, \quad (2.20)$$

where $\beta = \sum_{d=1}^k \{\rho_d + \tau_d\}$. RSD can be written as a polynomial function

$$\Omega(x) = \sum_{d=1}^k \Omega_d x^d. \quad (2.21)$$

Using RSD, the receiver can recover the k source bits from any $n = k + O(\sqrt{k} \cdot \ln^2(k/\delta))$ encoded bits with probability at least $(1 - \delta)$. Parameters c and δ constitute the tunable parameters of the distribution [23]. For example, c maintains the average degree distribution in a reasonable range.

Figure 2.12 illustrates the ISD (dash-dotted curve) and RSD (solid curve) at generation size $k = 128$, $c = 0.02$, and $\delta = 0.1$. They both have the highest probability at degree $d = 2$. The probability of degree $d = 1$ is higher in RSD. In addition, the main difference between the two distributions is that the RSD has a *spike* degree at $d = \lfloor \frac{k}{S} \rfloor = 79$.

Figure 2.13 shows the position of the spike degree at generation size $k = 128$ and $\delta = 0.1$ with different values of parameter c . As can be seen in the figure, the position

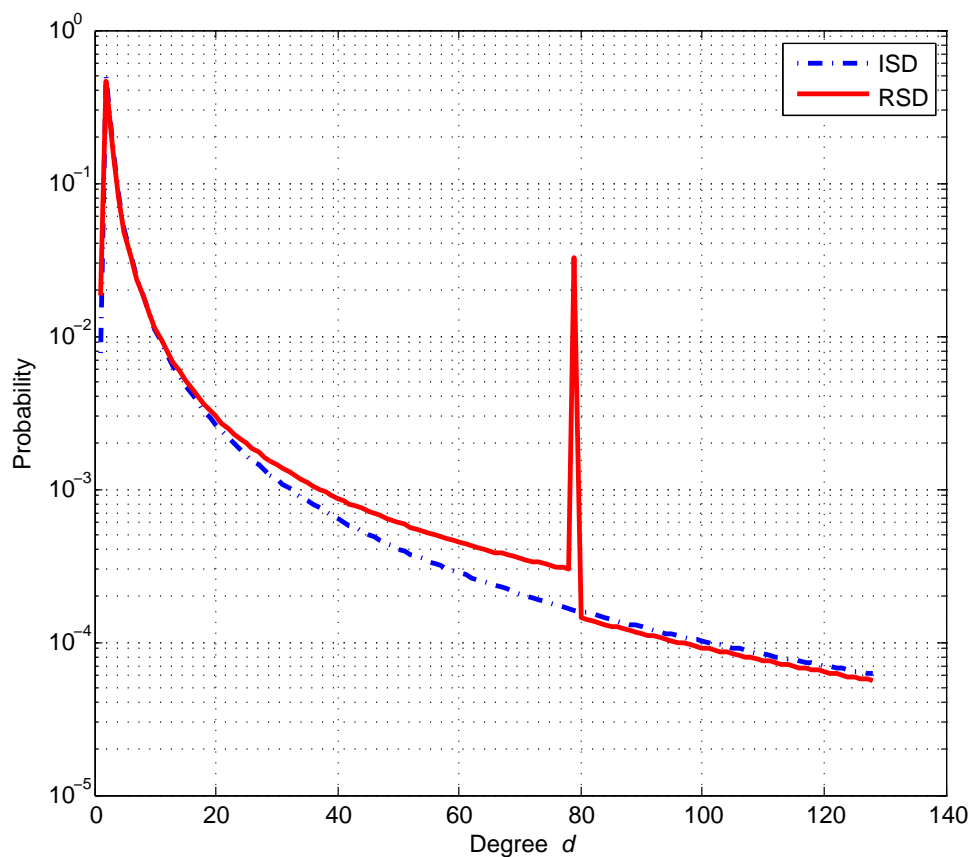


Figure 2.12: Ideal soliton distribution (ISD) and robust soliton distribution (RSD) at generation size $k = 128$, $c = 0.02$, and $\delta = 0.1$.

of the spike degree increases (moves to the right) as the c value decreases. However, very small c values push the spike away from the range of valid degrees (i.e., from $d = 2$ to $d = k$). Thus, one must choose the parameters c and δ precisely.

LT codes with RSD are asymptotically *optimal* and *universal* codes over the BEC in terms of maximizing the code rate.

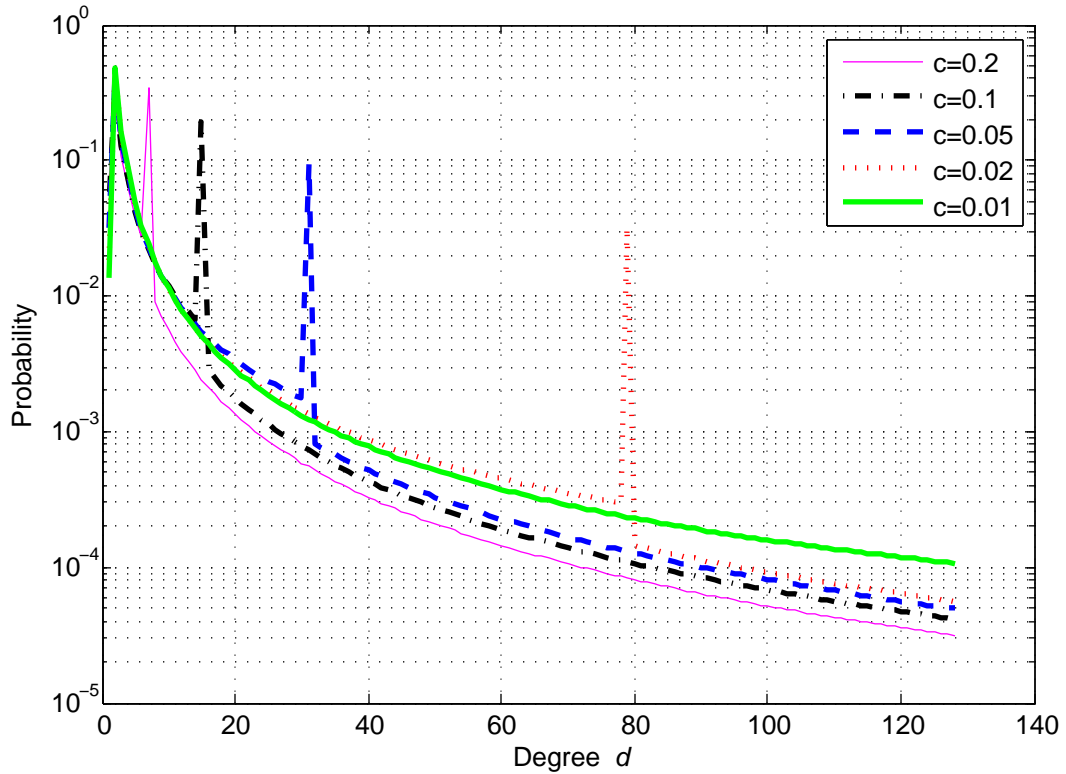


Figure 2.13: Robust soliton distribution (RSD) at generation size $k = 128$ and $\delta = 0.1$ with different values of parameter c .

Definition 2.23: Universal-optimal code

A universal-optimal code over a family of channels is a code that achieves the channel capacity under any channel characteristics from the same family.

Another way to represent LT codes is using *edge-perspective degree distributions* instead of using node degree distributions (or simply, degree distributions).

Definition 2.24: Edge-perspective degree distribution

Over a Tanner graph corresponding to an LT code with a generator matrix G , the edge-perspective degree distribution of source (encoded) bits is given by

$$\lambda(x) = \sum_{j=1} \lambda_j x^j,$$

where λ_j is the fraction of edges that are connected to a source (encoded) nodes of degree j .

The average degree of RSD is given by

$$\Omega'(1) = \sum_d d\Omega_d, \quad (2.22)$$

where $\Omega'(x)$ is the derivative of $\Omega(x)$ with respect to x . By generating $n = k(1 + \varepsilon)$ encoded bits, where ε is the overhead percentage, there is a probability $p_{uncovered}$ that a source bit is not covered. The probability is given by

$$p_{uncovered} = \left(1 - \frac{\Omega'(1)}{k}\right)^n. \quad (2.23)$$

Consequently, this probability which depends on the average degree $\Omega'(1)$, generation size k , and number of encoded bits n , is considered as a lower bound on the successful decoding. The lower bound makes the performance of LT codes exhibit *error floor* as the overhead increases.

Definition 2.25: Error floor

In fountain codes, the performance (particular bit error rate, BER) curves are well-known as waterfall curves. At some point while the overhead (or SNR) increases, the curves do not fall as quickly as previous points (the performance in this region becomes almost flat). This phenomenon is referred to as error floor.

2.4.3 Raptor Codes

Raptor codes were introduced by Shokrollahi in 2004 as an extension to LT codes [24, 86]. They are obtained using LT codes as an inner code serially concatenated with high-rate LDPC codes as an outer code. Mainly, Raptor codes were proposed to reduce the encoding and decoding complexities of LT codes. In Raptor codes, each encoded bit is produced using $O(\ln(1/\varepsilon))$ bit operations, where ε is the overhead. The decoder requires $O(k \ln(1/\varepsilon))$ bit operations to recover the k source bits.

The main reason behind the concatenation is to relax the full recovery requirement of LT decoder. In other words, the problem of error floor will diminish significantly by relaxing the recovering condition of the LT decoder. In Raptor codes, the LT decoder does not have to recover the entire source bits, instead, only a constant fraction of source bits is enough to be recovered by LT decoder. The remains unrecovered source bits will be decoded using the outer decoder (i.e., the LDPC decoder). Therefore, the LT degree distribution should be modified accordingly.

Figure 2.14 shows the structure of the Raptor codes. The pre-coding is done using systematic fixed-length LDPC codes where $(k + m)$ intermediate encoded bits are generated from the k source bits. Then, n encoded bits are generated using the LT

encoder from the $(k+m)$ intermediate encoded bits. As can be seen in the figure, the intermediate encoded bit y_{k+2} (as an example) is not covered in LT coding. However, it is not a problem in Raptor codes because this uncovered bit y_{k+2} can be decoded using the LDPC decoder.

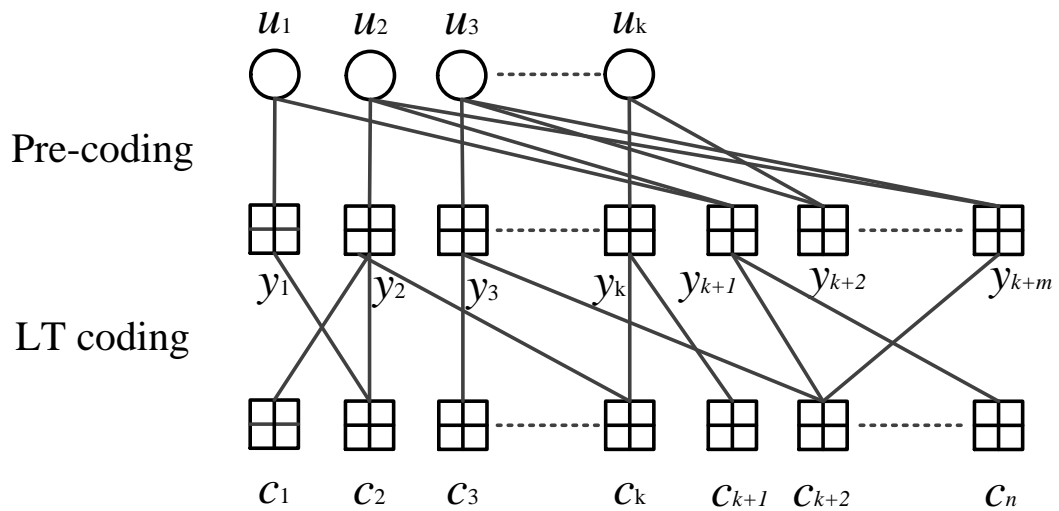


Figure 2.14: Raptor coding scheme: $(k+m)$ intermediate encoded bits are generated from k source bits using a pre-coding (e.g., LDPC codes). Then, LT encoder generates n encoded bits using the $(k+m)$ intermediate encoded bits.

Chapter 3

Degree Distribution Design

As we showed in Chapter 2, LT codes with RSD are asymptotically *optimal* and *universal* over a BEC in the sense that they achieve the capacity over any erasure probability. Despite the good performance of LT codes, universality and optimality do not exist in the finite-length regime. In fact, one can easily design new degree distributions that outperform the RSD. In addition, the parameters of the RSD can be carefully chosen to provide the right balance in terms of a number of conflicting objectives. Our contribution in this chapter is to design new LT codes that are robust to the communication system's parameters such as erasure probability ϵ as well as the source length k .

We employ density evolution (DE) and extrinsic information transfer (EXIT) chart together with Monte Carlo optimization to design robust LT codes offering some of the attractive properties of *universally optimal* codes. Different objectives are used in the analysis such as minimizing the erasure probability and maximizing the code rate. Besides, we use code stability method to design a new LT codes with the highest stability over different BECs.

We start this chapter by defining the terminologies used in the chapter and throughout the thesis.

3.1 Basic Notations

We start by analytically reviewing the encoded and source degree distributions of LT codes.

Definition 3.1: Encoded degree distribution

For any block codes, the encoded (also referred to as right or output) degree distribution is given by

$$\Omega(x) = \sum_{d=1}^k \Omega_d x^d, \quad (3.1)$$

where k and Ω_d are the source length and the probability of an encoded node having a degree d , respectively.

In LT codes, the probability Ω_d is obtained from the RSD, Equation (2.20). As such, the average degree of the encoded bits $\Omega_{avg} = \Omega'(1)$ is given by Equation (2.22). In addition, the encoded edge degree distribution is given as

$$\rho(x) = \frac{\Omega'(x)}{\Omega'(1)} = \sum_{d=1}^k \rho_d x^{d-1}. \quad (3.2)$$

In the encoding process, while the d source bits are chosen uniformly at random, the source (or left) degree distribution $\Lambda(x)$ follows a binomial distribution:

$$\Lambda(x) = \sum_{i=0}^n \Lambda_i x^i, \quad (3.3)$$

where

$$\Lambda_i = \binom{n}{i} \left(\frac{\Omega_{avg}}{k} \right)^i \left(1 - \frac{\Omega_{avg}}{k} \right)^{n-i}, \quad (3.4)$$

and n is the number of received encoded bits at a realized code rate $R = k/n$. A binomial distribution converges to the Poisson distribution asymptotically with $\eta = \Omega_{avg}n/k = \Omega_{avg}/R = \Omega_{avg}(1 + \varepsilon)$ as the average rate. Therefore,

$$\Lambda_i = \frac{e^{-\eta} \cdot \eta^i}{i!}. \quad (3.5)$$

Hence, Equation (3.3) can be asymptotically approximated to

$$\Lambda(x) = e^{\eta(x-1)}. \quad (3.6)$$

Equivalent to the average degree of the encoded bits in Equation (2.22), the average degree of the source bits is equal to:

$$\Lambda_{avg} = \sum_{i=0}^n i \Lambda_i = \Lambda'(1), \quad (3.7)$$

where $\Lambda'(x)$ is the derivative of the source degree distribution in Equation (3.3) with respect to x .

Note that, the total number of edges E in the graph is given by

$$E = \Omega_{avg} \times n = \Lambda_{avg} \times k. \quad (3.8)$$

Finally, the source edge degree distribution is determined as

$$\lambda(x) = \frac{\Lambda'(x)}{\Lambda'(1)} = \sum_{i=1}^n \lambda_i x^{i-1}. \quad (3.9)$$

Next, we use the following lemma to calculate the code rate of an LT code.

Lemma 3.1

For an LT code with $\lambda(x)$ and $\rho(x)$ as source and encoded edge degree distributions, respectively, the code rate R is given as

$$R = \frac{\int_{x=0}^1 \lambda(x) dx}{\int_{y=0}^1 \rho(y) dy}. \quad (3.10)$$

Proof. Using Equation (3.8), we can write the code rate as

$$R = \frac{k}{n} = \frac{\Omega_{avg}}{\Lambda_{avg}}, \quad (3.11)$$

then by using Equation (3.2), we can write $\rho_d = d \times \Omega_d / \Omega_{avg}$ for $d = 1, \dots, k$. Now, we divide both sides by d and take the summation over all d 's to get

$$\Omega_{avg} = \frac{1}{\sum_{d=1}^k \frac{\rho_d}{d}}. \quad (3.12)$$

Similarly by using Equation (3.9), we get

$$\Lambda_{avg} = \frac{1}{\sum_{i=1}^n \frac{\lambda_i}{i}}. \quad (3.13)$$

Using Equation (3.11), the code rate can be written as

$$\begin{aligned} R &= \frac{\sum_{i=1}^n \frac{\lambda_i}{i}}{\sum_{d=1}^k \frac{\rho_d}{d}} \\ &\approx \frac{\int_{x=0}^1 \lambda(x) dx}{\int_{y=0}^1 \rho(y) dy}. \end{aligned} \quad (3.14)$$

where the last equation is valid for large k and n . □

3.2 Code Analysis

In this section, we focus on three analysis methods that can be used to design, evaluate, and compare graph codes. These methods are DE, EXIT chart, and code stability.

3.2.1 Density Evolution (DE)

DE is an analytical method that predicts the convergence behavior of iterative decoders by tracking the probability distributions of extrinsic log-likelihood ratios (LLRs) for a given code ensemble [71, 87]. In fact, DE determines exact expressions for the average probability density functions of the messages passing between the nodes in

each iteration. Besides, DE gives a tool that can be used in designing families of codes with iterative decoders. The underlying assumption is that the graph describing the particular code is a *cycle-free* graph. For noisy channels, DE tracks the evolution of entire probability density functions [71, 88]. However, for erasure channels, DE tracks only the density distribution of the erasure probability being passed from one side of the decoder to another [65, 66]. Then, the bit erasure probability (BEP) associated with the density can be determined. Note that this DE process is only used to construct the code *offline*, and thus only needs to be performed once during code design.

Definition 3.2: Cycle-free graph

A graph is a cycle-free graph (also referred to as a tree graph) if and only if any two of its nodes are connected by one path only.

In fact, DE originally was used for LDPC codes [71] and Turbo codes [89]. However, a modified version of DE of graph code over a BEC is developed independently by Luby et. al [90, 91]. It is referred to as And-Or tree analysis. And-Or tree analysis is used to design the degree distributions of graph codes over a BEC such as LT codes [23, 92], Raptor codes [24, 25] and online codes [93].

And-Or Tree Analysis

And-Or tree analysis can be described as follows: For LT codes with generator matrix G , consider, uniformly at random, an edge connecting a source node u_i , $1 \leq i \leq k$ with encoded node c_j , $1 \leq j \leq n$. Then, a subgraph G_l that corresponds to u_i can be obtained. The subgraph G_l has a maximum depth $2l$ with the root at depth 0 and the leaves at depth $2l$. The source nodes are labeled as “OR-nodes” at depths

0, 2, 4, ..., (2l - 2) whereas the encoded nodes are labeled as “AND-nodes” at depths 1, 3, 5, ..., (2l - 1). Figure 3.1 shows an example of the And-Or tree corresponding to subgraph G_l . The assumption here is that for large n , the subgraph G_l converges to a tree [93].

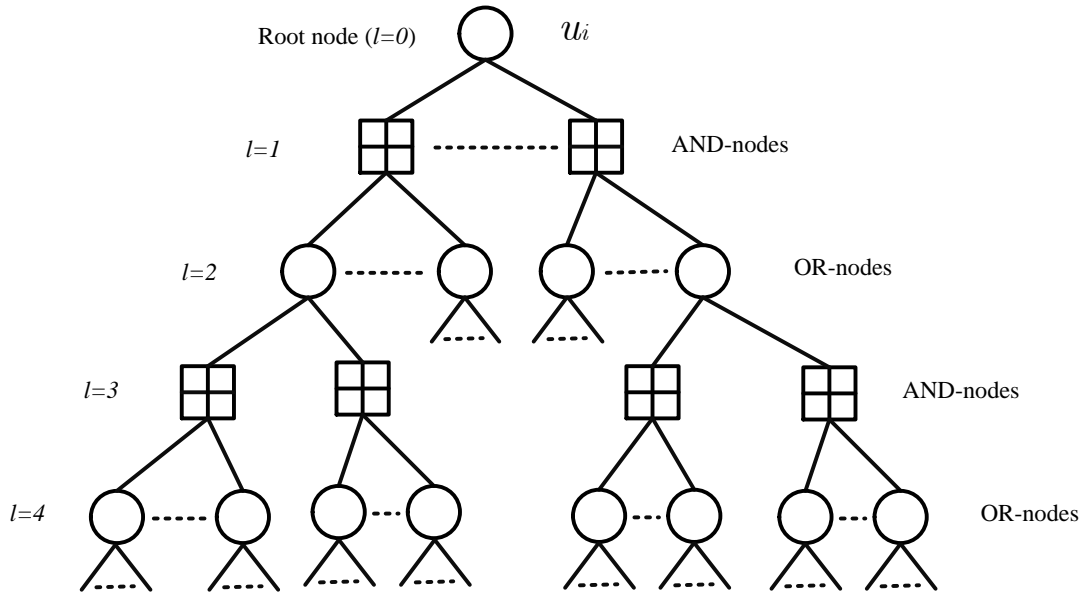


Figure 3.1: An example of And-Or tree for a subgraph G_l where circles and squares represent OR-nodes and AND-nodes, respectively.

Each “OR-node” is connected to a number of children depending on a specific distribution related to the input degree distribution $\Lambda(x)$; that is the source edge degree distribution $\lambda(x)$. Therefore, each “OR-node” is connected to j children with probability λ_j , $0 \leq j \leq \lambda_{\max}$, where λ_{\max} is the maximum degree of the distribution. Also, each “AND-node” is connected to a number of children depending on a distribution related to the encoded degree distribution $\Omega(x)$; that is the encoded edge degree distribution $\rho(x)$. Therefore, each “AND-node” is connected to i children with

probability ρ_i , $0 \leq i \leq \rho_{\max}$, where ρ_{\max} is the maximum degree of the distribution.

The leaves are independently assigned to 0 with probability y_0 or 1 with probability y_1 . We are interested in finding the probability y_l which evaluates the root of G_l to 0. Aforementioned can be done recursively using the probability y_{l-1} which evaluates the root of G_{l-1} to 0 and so on. The probability is calculated as follows:

The outgoing message $M_{s \rightarrow e}^j$ of a source node s at step j to an encoded node e is an erasure with probability:

$$M_{s \rightarrow e}^j = \sum_{d=0}^{n-1} \lambda_d (M_{e \rightarrow s}^j)^d = \lambda(M_{e \rightarrow s}^j). \quad (3.15)$$

Note that, the source bit cannot be decoded if and only if it receives an erasure from all of its neighbor encoded bits. In addition, the outgoing message from the source bit is an erasure if at least one of incoming messages is an erasure. Otherwise, the outgoing message is the XOR of the incoming messages.

The outgoing message $M_{e \rightarrow s}^j$ of an encoded node e at step j to a source node s is an erasure with probability:

$$\begin{aligned} M_{e \rightarrow s}^j &= \sum_{d=0}^{k-1} \rho_d \left[1 - (1 - M_{s \rightarrow e}^{j-1})^d (1 - \epsilon) \right] \\ &= 1 - \rho (1 - M_{s \rightarrow e}^{j-1}) (1 - \epsilon). \end{aligned} \quad (3.16)$$

Note that, an encoded bit cannot be decoded either if it is erased from the channel or it receives an erasure from one of its neighboring source bits. In addition, the outgoing message from the encoded bit is an erasure if all the incoming messages are erasures. Otherwise, the outgoing message is based on the majority vote of the incoming non-erasure messages. The initial message from an encoded bit to a source

bit is the erasure probability from the channel ϵ (i.e., $M_{e \rightarrow s}^1 = \epsilon$).

In conclusion, the probability y_l that the root node of a G_l And-Or tree evaluates to 0 is given by the Lemma (3.2) [90].

Lemma 3.2: The And-Or Tree Lemma

The probability y_l that the root node of a G_l And-Or tree evaluates to 0 is $y_l = f(y_{l-1})$, where y_{l-1} is the probability that the root node of a G_{l-1} And-Or tree evaluates to 0, and

$$f(x) = \lambda(1 - (1 - \epsilon)\rho(1 - x)), \text{ for} \quad (3.17)$$

$$\lambda(x) = \sum_{i=0}^{\lambda_{\max}} \lambda_i x^i \quad \text{and} \quad \rho(x) = \sum_{j=0}^{\rho_{\max}} \rho_j x^j.$$

Proof. The proof is simple as shown above. For more details, please refer to [90]. \square

3.2.2 Extrinsic Information Transfer (EXIT) Chart

EXIT chart was proposed by Stephan ten Brink in 1999 as a powerful method for analyzing iteratively decoded codes such as Turbo codes [94, 95]. Basically, EXIT chart visualizes the convergence behavior of decoder of Turbo codes. In addition, it can be used to search for good codes or compare between codes in terms of decoding speed. EXIT chart, as well as DE, reveals a decoding threshold value; the code cannot recover the source messages correctly with high probability for channels worst than the threshold [15]. The main advantage of using EXIT chart over DE is that EXIT chart tracks only one number per iteration such as LLRs on their mean, variance, fidelity, an error probability, or mutual information [87, 96] while DE tracks all of the density

functions. As a result, DE is often difficult and complicated to be analyzed for most channels except the BEC where only the channel erasure probability is considered. However, EXIT chart is not accurate as DE. EXIT is also successfully used to analyze and design other codes such as LDPC codes [88, 97, 98], fountain codes [99].

In LT codes over noisy channels, the decoder is divided into two decoders exchanging extrinsic mutual information. The first decoder is on the side of source nodes, and the other one is on the side of encoded nodes. Spacial case for the BEC, EXIT chart is equivalent to DE because only one variable is tracked [87].

3.2.3 Code Stability

We develop a stability condition for LT codes over the BEC. The stability condition is presented in Lemma (3.3).

Lemma 3.3: Stability Condition

For an LT code with edge degree destitution pair (λ, ρ) and erasure probability ϵ , the stability condition is given by:

$$\eta \cdot e^{-\eta} \cdot \rho'(1) < 1, \quad (3.18)$$

where $\eta = \Omega_{avg} n/k = \Lambda_{avg} = \Lambda'(1)$ is the average source degree.

Proof. Another version of the DE is given by the following equation [90]:

$$y_l = \lambda(1 - \rho(1 - y_{l-1})), \quad (3.19)$$

where the erasure probability ϵ is implicitly given in the degree distributions $\lambda(x)$ and $\rho(x)$. Next, we expand the Equation (3.19) into Taylor series around zero, we get

$$y_l = \lambda(0) + [\lambda'(0)\rho'(1)]y_{l-1} + O(y_{l-1}^2), \quad (3.20)$$

where $\lambda'(x)$ and $\rho'(x)$ are the derivatives of $\lambda(x)$ and $\rho(x)$ with respect to x , respectively. If y_l is sufficiently small, $O(y_{l-1}^2)$ term approaches zero. Thus, we get

$$y_l = \lambda(0) + [\lambda'(0)\rho'(1)]y_{l-1}. \quad (3.21)$$

As a result, the convergence of any LT code depends on the $\lambda'(0)\rho'(1)$ value. This value must be less than 1 as a sufficient condition for the LT decoder to converge (or equivalently, to achieve zero BEP in an infinite number of LT decoder iterations). That means

$$\lambda'(0)\rho'(1) < 1. \quad (3.22)$$

Next, using Equations (3.6) and (3.9), the source edge degree distribution $\lambda(x)$ can be written as

$$\lambda(x) = e^{\eta(x-1)}, \quad (3.23)$$

and its derivative as

$$\lambda'(x) = \eta \cdot e^{\eta(x-1)}. \quad (3.24)$$

Thus, substituting $\lambda'(0)$ from Equation (3.24) into Equation (3.22) and doing some manipulations proves the lemma.

□

Lemma (3.3) is used to give an upper bound (threshold) on the erasure probability ϵ such that for any channel with erasure probability below the threshold, the LT code with this edge degree distribution pair $(\lambda(x), \rho(x))$ can be *reliably* transmitted over the channel. Besides, the lemma can be used as a lower bound on the average degree $\Omega'(1)$ for the *reliable* decoder.

As in the case of LDPC codes [100], for a stable LT code and using Equation (3.19), the convergence condition is given by:

$$\lambda(1 - \rho(1 - x)) < x, \quad (3.25)$$

for $\forall x \in (0, 1)$. Then, by taking the inverse function of $\lambda(x)$ for both sides (it is noted that $\lambda(x)$ has only non-negative coefficients), we get

$$c(x) = 1 - \rho(1 - x) < \lambda^{-1}(x) = v^{-1}(x). \quad (3.26)$$

As a result, for a stable code, $c(x)$ must be below $v^{-1}(x)$. That means the code cannot be transmitted and received reliably with an erasure probability above ϵ^* . Note that this threshold erasure probability is different and tighter than Shannon limit for the code (i.e., $\epsilon_{Sh} = 1 - R$). One should minimize the gap between these two thresholds by increasing ϵ^* . Figure 3.2 shows $c(x)$ and $v^{-1}(x)$ of an LT code of packet length $k = 128$ and code length $n = 256$ (i.e., the code rate $R = 1/2$) with RSD ($c = 0.02$ and $\delta = 0.5$) over BEC with different erasure probabilities ϵ . The figure shows that this code cannot be reliably transmitted over the BEC with an erasure probability more than $\epsilon = 0.48$.

In order to show how the decoder works using EXIT chart, we use the previous

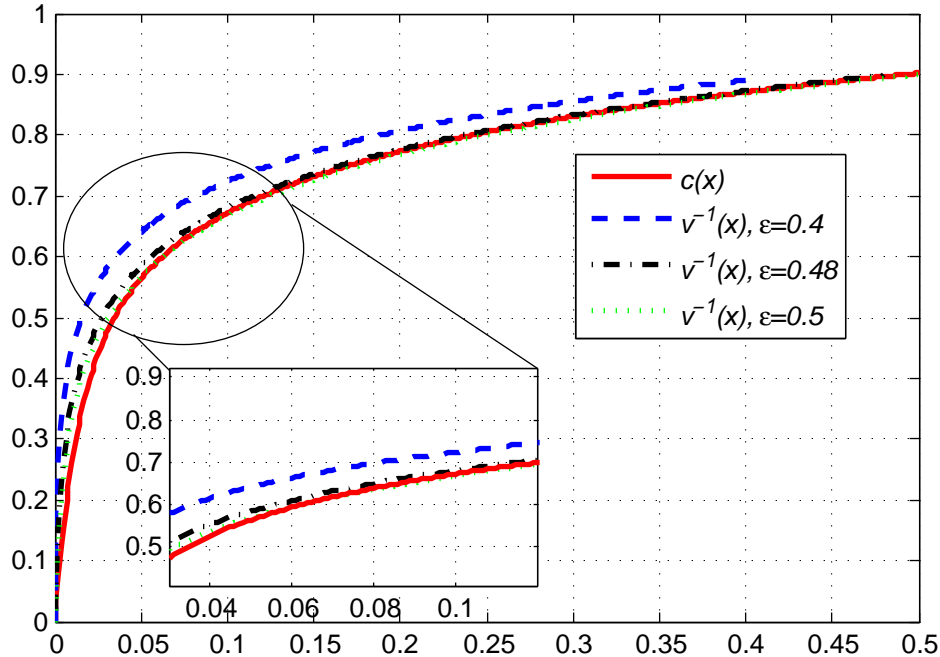


Figure 3.2: EXIT chart for an LT code with packet length $k = 128$ and code length $n = 256$ over different BEC's. RSD degree distribution is used with parameters $c = 0.02$ and $\delta = 0.5$

LT code at channel erasure probability $\epsilon = 0.2$. Figure 3.3 shows $c(x)$ and $v^{-1}(x)$ for this code. The decoder starts using the erasure probability $\epsilon = 0.2$, $\epsilon = v(x) = 0.2$ is the initial messages erased by the encoded nodes. Then, the source nodes send erasure messages to the encoded node with $c(0.2) = 0.7731$. At the second iteration, the encoded nodes send erasure messages to the source nodes with $x = v^{-1}(0.7731) = 0.081$ and then the source nodes send erasure messages with $c(0.081) = 0.6402$. This process continues until both curves reach zero as shown by the dotted lines in the figure.

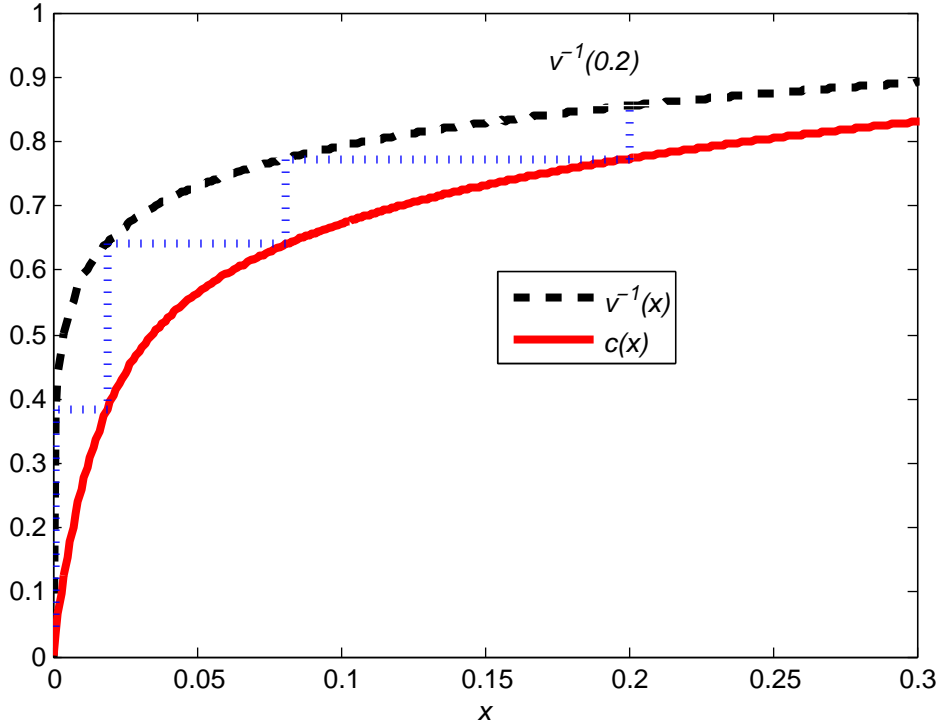


Figure 3.3: EXIT chart for an LT code with packet length $k = 128$ and code length $n = 256$ over a BEC of erasure probability $\epsilon = 0.2$. RSD degree distribution is used with parameters $c = 0.02$ and $\delta = 0.5$

3.3 Design RSD for Finite-Length Regime

In this section, we design robust LT codes in terms of the erasure probability ϵ and the code length k . DE and EXIT chart together with Monte Carlo optimization are used to optimize the parameters of the degree distribution with different objectives. Two objectives are used in the optimization, namely: maximizing the code rate R and minimizing the BEP.

3.3.1 Tuning the Degree Distribution for Robustness

Much research has been done in the area of LT codes using different values of c and δ parameters without much attention to important properties such as robustness. In the absence of universality, robust designs are particularly important in networks with many diverse users and changing channels conditions. In what follows, we aim to design robust LT codes given rate and erasure objective functions.

We start by using the equations from Section 3.2.1. After a significant number of iterations, the two probabilities in Equations (3.15) and (3.16) converge. Thus, the BEP equals to the probability that a source bit cannot be recovered which is given by

$$\text{BEP} = \sum_d \Lambda_d (M_{e \rightarrow s}^{j \rightarrow \infty})^d. \quad (3.27)$$

Next, we set upper and lower bounds on parameter c of the RSD (Equation 2.20) using Lemma (3.4).

Lemma 3.4

Given an LT code whose length is k with RSD, and parameters c and δ , then c can be bounded such that

$$\frac{1}{\sqrt{k} \ln(k/\delta)} \leq c \leq \frac{1}{2} \frac{\sqrt{k}}{\ln(k/\delta)}. \quad (3.28)$$

Proof. As can be seen from the RSD distribution, there is a spike degree at $d = k/S$ [23]. The benefit of having this spike is to ensure that all the source bits are covered in the encoding process. The position of the spike should be a valid degree; i.e., between 2 (if the lowest degree is 1, the RSD distribution becomes all-at-once

distribution) and k . Thus,

$$\begin{aligned} 2 &\leq \frac{k}{S} \leq k, \\ \Rightarrow 1 &\leq S \leq \frac{k}{2}. \end{aligned}$$

By setting $S = c \ln(k/\delta) \sqrt{k}$ from Equation (2.18) and do some manipulations, we get an upper bound on c as

$$c \leq \frac{1}{2} \frac{\sqrt{k}}{\ln(k/\delta)}, \quad (3.29)$$

and a lower bound as

$$c \geq \frac{1}{\sqrt{k} \ln(k/\delta)}. \quad (3.30)$$

□

Figure 3.4 illustrates the upper and lower bounds on c from Lemma (3.4) with changing the packet length k at fixed δ . As can be seen from the figure, as the packet length increases, the gap between the upper and lower bounds increases as well. In addition, the lower bound changes slightly at high packet length ($k > 50$). Also, increasing δ does not affect the bounds heavily.

By using Lemma (3.4) from another perspective, we can find bounds on the error probability δ at fixed parameter c . Figure 3.5 shows the upper and lower bounds on the failure probability as a function of c at different packet lengths k . From the figure, we can tell that the smaller the c the better the performance. For example, at a packet length $k = 1024$ and $c = 0.2$ the minimum failure probability is 1.84×10^{-32} . However, if parameter c is decreased to 0.1, the minimum failure probability will be decreased to 3.3×10^{-67} .

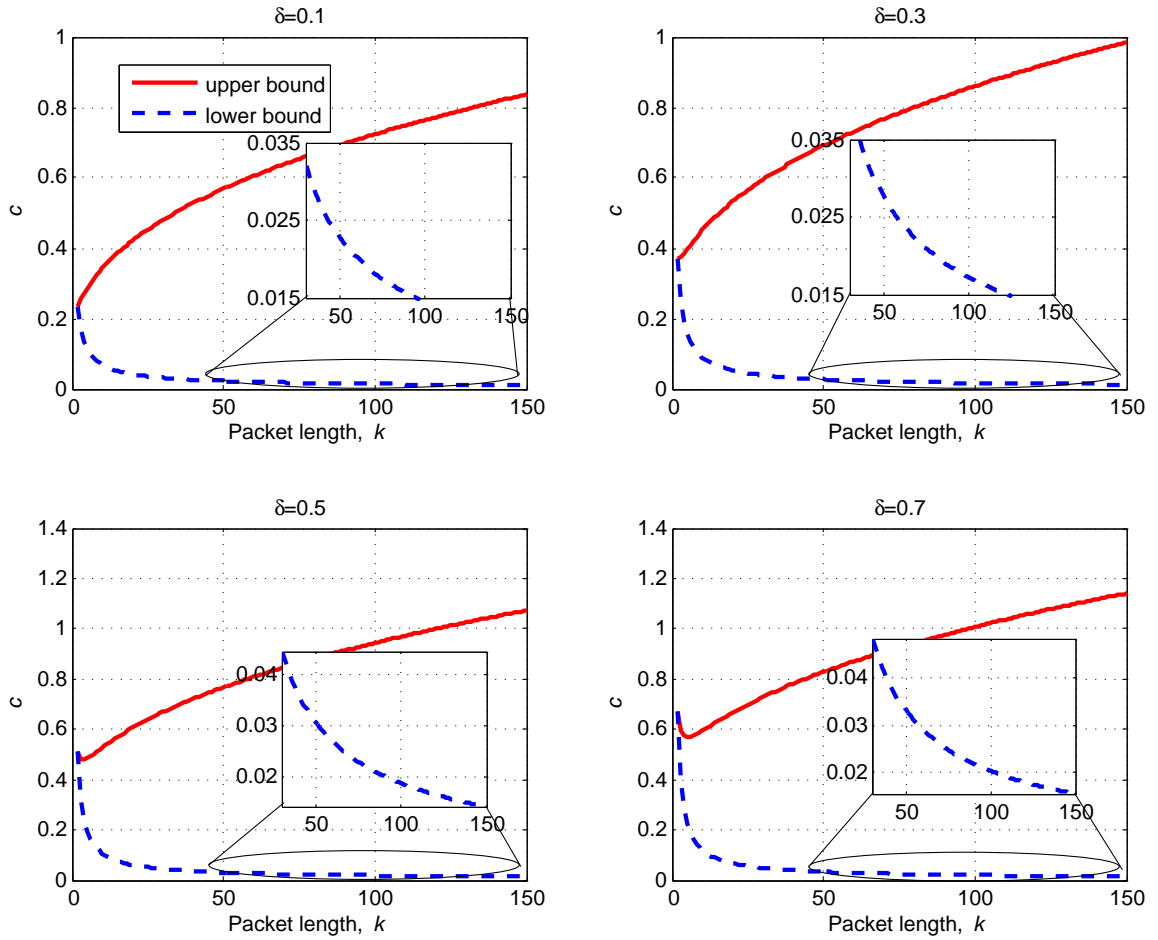


Figure 3.4: Upper and lower bounds on parameter c versus packet lengths k of LT codes with fixed δ .

Lemma 3.5

Given an LT code whose length is k and code rate R with RSD of parameters c and δ , then for a successful decoding with error probability no more than δ , the average degree distribution Ω_{avg} must be at least $c \cdot R \cdot \ln(k)$

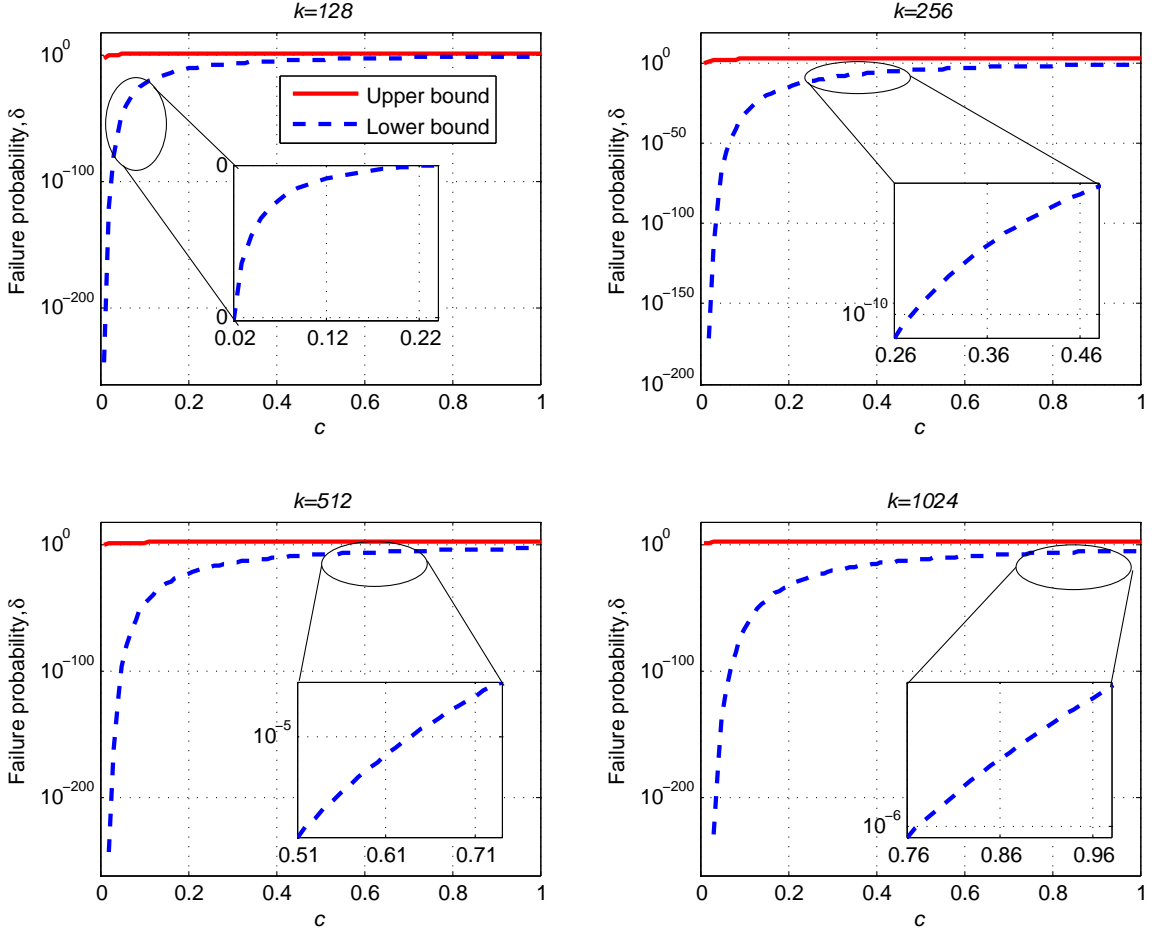


Figure 3.5: Failure probability δ versus parameter c of LT codes at different packet lengths k .

Proof. The probability that a source bit is not covered in the encoding process is given by

$$\left(1 - \frac{\Omega_{avg}}{k}\right)^n,$$

where n is the total number of encoded bits. This probability must be decreased as the code length k increases. Thus, the decoding process succeeds to recover the

source bits with high probability if and only if

$$\left(1 - \frac{\Omega_{avg}}{k}\right)^n \leq \frac{1}{k^c}, \quad (3.31)$$

which can be asymptotically approximated to

$$e^{-\frac{n\Omega_{avg}}{k}} \leq \frac{1}{k^c}. \quad (3.32)$$

Then, by doing some manipulations, we get

$$\Omega_{avg} \geq c \cdot R \cdot \ln(k). \quad (3.33)$$

□

In the next subsection, the Monte Carlo optimization is used to optimize parameters c and δ with different objectives.

3.3.2 RSD Parameters via Monte Carlo Optimization

We set our objective as maximizing the code rate $R = k/n$ while having the same performance (zero error probability). Theoretically, the number of received encoded bits n must be at least $k + c \ln^2(k/\delta) \sqrt{k}$ [23]. As we show in Lemma (3.1), the code rate is directly proportional to $\sum_i \frac{\lambda_i}{i}$ and inversely proportional to $\sum_d \frac{\rho_d}{d}$. Therefore, one can maximize $\sum_i \frac{\lambda_i}{i}$ in order to maximize the code rate R . We set the following

Monte Carlo optimization (MCO1):

$$\begin{aligned}
 & \text{Maximize} && \sum_i \frac{\lambda_i}{i}, \\
 & \text{subject to} && n \geq k + c \ln^2(k/\delta) \sqrt{k}, \\
 & && \Omega_{avg} \geq cR \ln(k), \\
 & && \frac{1}{\sqrt{k} \ln(k/\delta)} \leq c \leq \frac{1}{2} \frac{\sqrt{k}}{\ln(k/\delta)}, \\
 & && 0 < \delta < 1.
 \end{aligned}$$

If we change the objective to minimize the BEP, the expression in Equation (3.27) can be transformed into a Monte Carlo optimization when generation length k , erasure probability ϵ , failure probability δ , and the code rate R are known. The corresponding Monte Carlo optimization (MCO2) is formulated as:

$$\begin{aligned}
 & \text{Minimize} && \sum_d \Lambda_d (M_{e \rightarrow s}^{j \rightarrow \infty})^d, \\
 & \text{subject to} && \frac{1}{\sqrt{k} \ln(k/\delta)} \leq c \leq \frac{1}{2} \frac{\sqrt{k}}{\ln(k/\delta)}, \\
 & && 0 < \delta < 1.
 \end{aligned}$$

3.4 Numerical Results

In this section, the performance of the optimized parameters is simulated through a series of examples for numerical evaluation.

Maximizing the Code Rate

In this subsection, we solve the MCO1 for different generation sizes at erasure probability $\epsilon = 0$. It should be noted that, similar results are obtained at higher erasure probabilities. At generation size $k = 128$, the optimized parameters $\delta = 0.9$ and $c = 0.06$ maximize the code rate to $R = 0.71081$. The corresponding code rate performances at $k = 128$ and $k = 512$ are shown in Figure 3.6 and Figure 3.7, respectively.

Figure 3.6 illustrates the code rate versus parameters c and δ at packet length of $k = 128$. The figure indicates that the maximum code rate we can achieve is $R = 0.71081$ at $c = 0.06$ and $\delta = 0.9$. Also, we can observe that as c decreases, the code rate improves until we get the point where the code rate degrades. The reason behind this is that small c 's are below the lower bound of c . Besides, we can see that δ changes the performance lightly. Note that, Shokrollahi distribution achieves a code rate $R = 0.5632$ at packet length of $k = 128$ [24]. Thus, we can improve the code rate by 26%.

Figure 3.7 shows the code rate versus parameters c and δ at packet length of $k = 512$. The figure indicates that the maximum code rate we can achieve is $R = 0.76331$ at $c = 0.05$ and $\delta = 0.9$. Also, we can observe the same behavior as in Figure 3.6. At this packet length $k = 512$, Shokrollahi distribution achieves a code rate $R = 0.7268$.

At larger packet length $k = 1024$, Figure 3.8 shows the code rate versus parameters c and δ . The figure shows the maximum code rate $R = 0.79241$ at $c = 0.04$ and $\delta = 0.8$.

To conclude this subsection, in our robust LT design if we target 95% (at least)

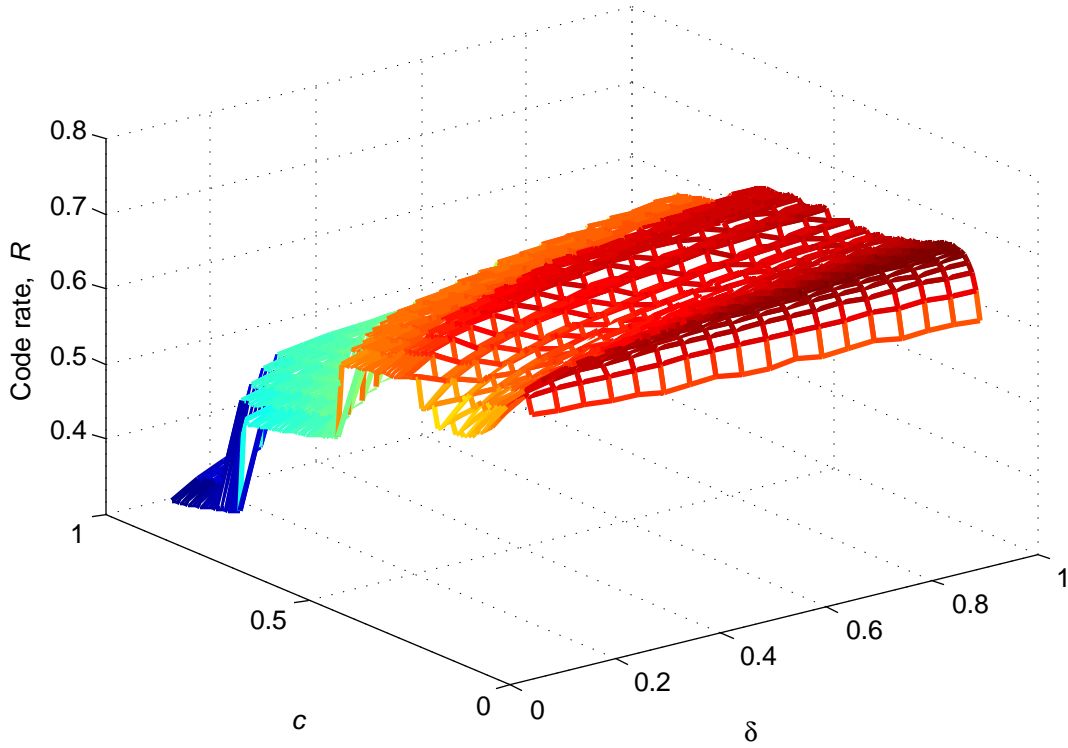


Figure 3.6: Code rate R of an LT code versus parameters c and δ at packet length $k = 128$.

of the maximum code rate over generation size $128 \leq k \leq 1024$, the optimized parameters are $0.85 \leq \delta \leq 0.9$ and $0.04 \leq c \leq 0.6$.

Minimizing the Erasure Probability

In this subsection, we solve the MCO2 for different erasure probabilities and different code rates. For example, at generation size $k = 128$, erasure probability $\epsilon = 0.02$, and code rate $R = 1/2$, the optimized parameters $\delta = 0.2$ and $c = 0.035$ minimize the BEP to 8.725×10^{-8} . The corresponding BEP performances at $k = 128$, $\epsilon = \{0, 0.02, 0.2\}$, and $R = \{1, 0.91, 0.84, 0.78, \dots, 0.5\}$ are shown in Figure 3.9 and Table 3.1.

Figure 3.9 illustrates the BEP versus the c at generation size $k = 128$, $\delta = 0.2$,

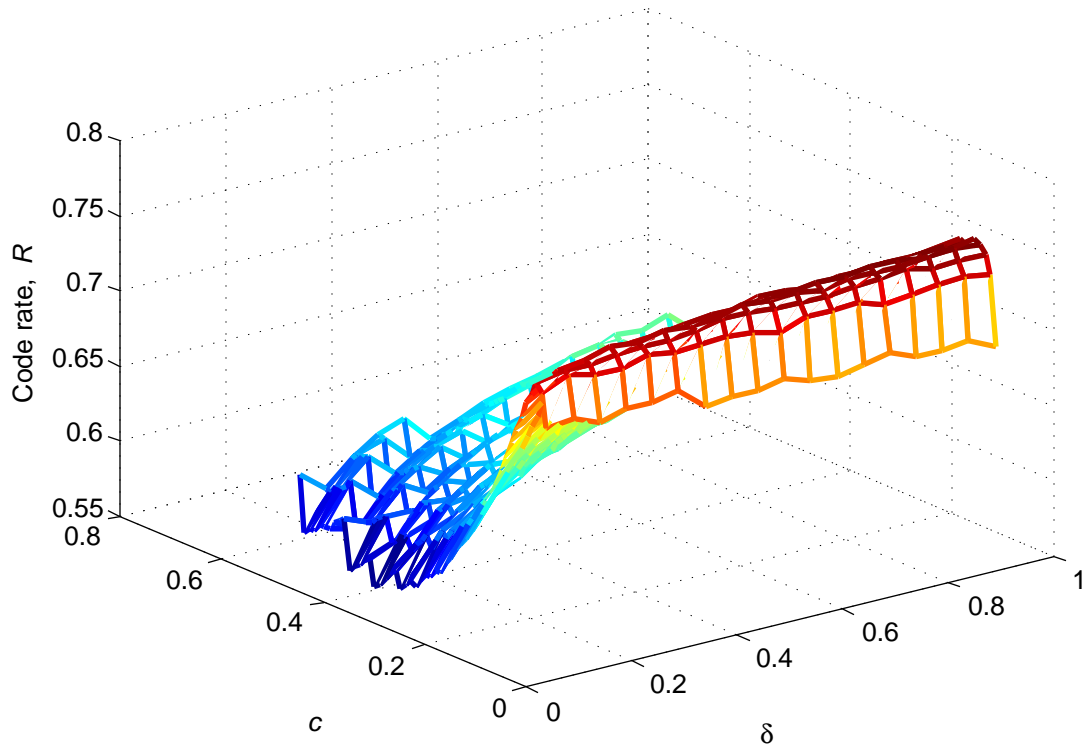


Figure 3.7: Code rate R of an LT code versus parameters c and δ at packet length $k = 512$.

and erasure probability $\varepsilon = 0.02$. The figure shows that the minimum BEP we can achieve is $\text{BEP} = 8.725 \times 10^{-8}$ at $c = 0.035$. We can observe that the minimum BEP is achieved at the same c , no matter what the code rate is. Small c values (below 0.015) have a bad performance because they are below the lower bound. Note that the code rate R in the figure ranges from $R = 1$ (upper curve) to $R = 1/2$ (lower curve).

Table 3.1 shows the optimization results of LT codes at $k = 128$, different erasure probabilities ε (columns), and different failure probabilities δ (rows). The table indicates the optimized c and its BEP. We can observe that the optimized c does not change as the erasure probability changes. Also, as δ increases, the BEP decreases.

To conclude this subsection, in our robust LT design if we target 110% (at most)

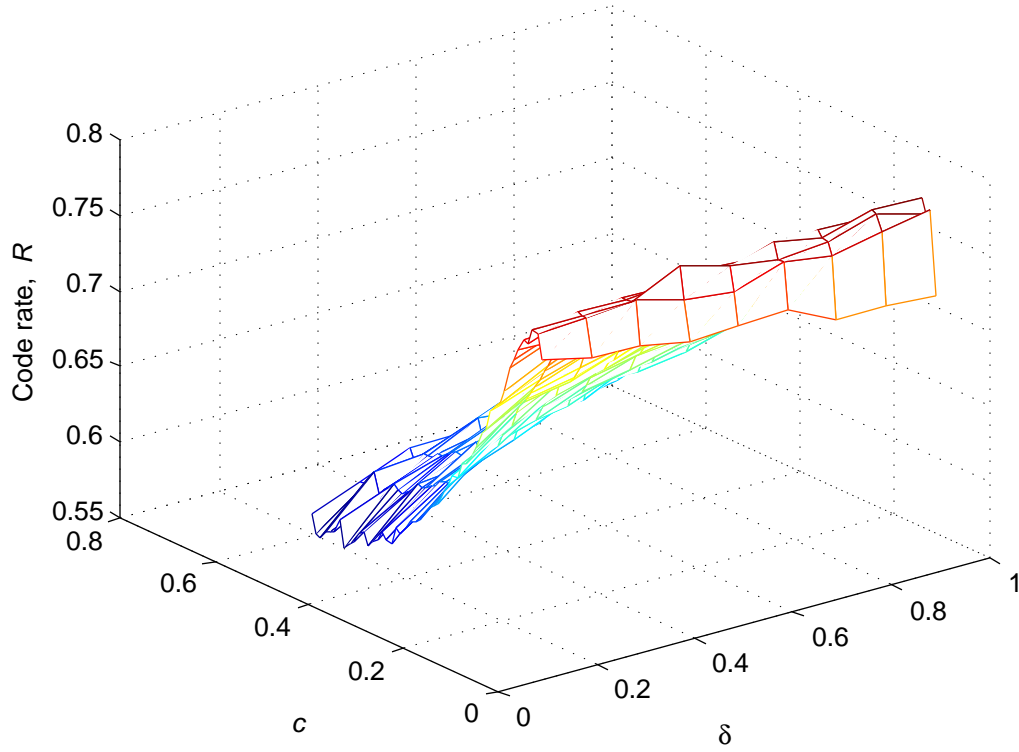


Figure 3.8: Code rate R of an LT code versus parameters c and δ at packet length $k = 1024$.

$\delta \backslash \epsilon$	0	0.02	0.2
0.1	4.575×10^{-12}	3.377×10^{-08}	2.181×10^{-06}
0.2	2.809×10^{-11}	8.725×10^{-08}	4.910×10^{-06}
0.5	3.316×10^{-10}	2.978×10^{-07}	1.407×10^{-05}
0.7	7.699×10^{-10}	4.636×10^{-07}	3.438×10^{-05}
0.9	1.461×10^{-09}	6.334×10^{-07}	2.694×10^{-05}

Table 3.1: Optimization results $\text{BEP} \backslash c$ at $k = 128$ and $R = 1/2$.

of the minimum BEP over the BEC with erasure probability ranging from $\epsilon = 0$ to $\epsilon = 0.2$ and any code rate, the optimized parameters are $0.1 \leq \delta \leq 0.2$ and

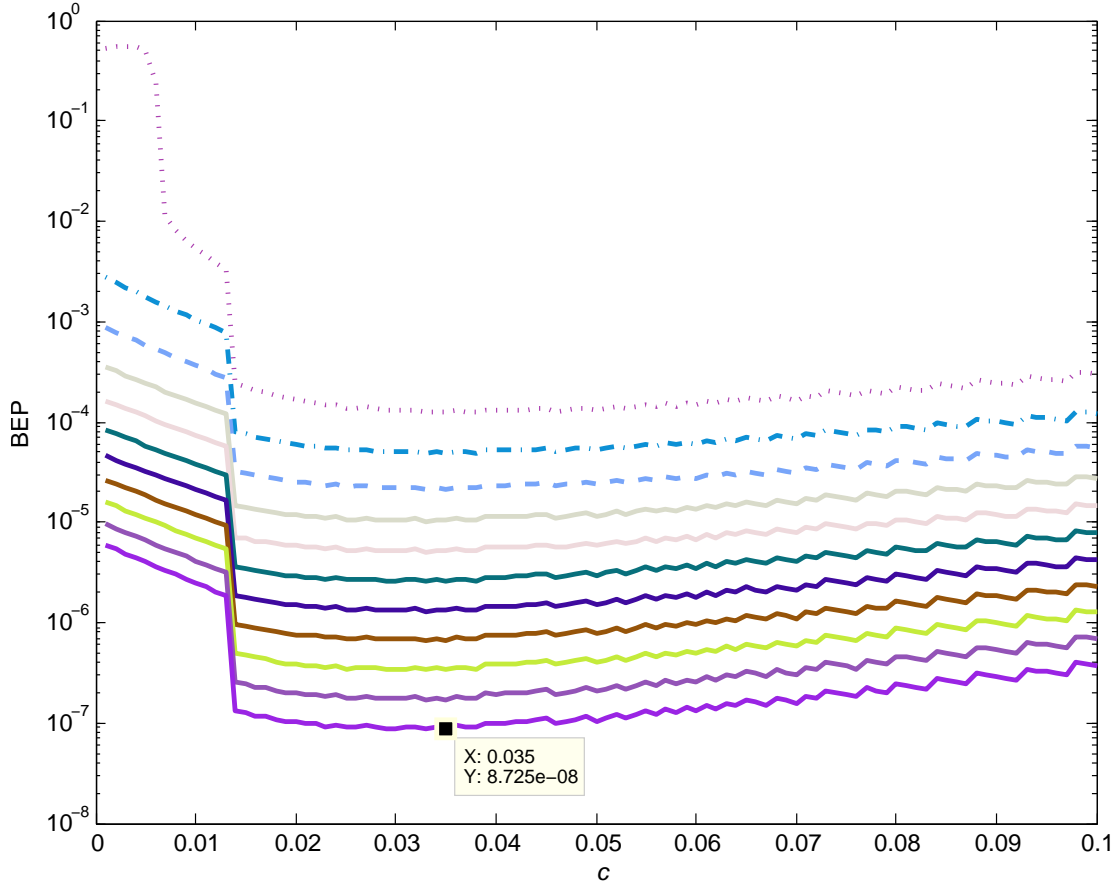


Figure 3.9: BEP versus parameter c at packet length $k = 128$, erasure probability $\epsilon = 0.02$, and $\delta = 0.2$. The code rate R ranges from $R = 1$ (upper curve) to $R = 1/2$ (lower curve).

$$0.025 \leq c \leq 0.06.$$

Maximizing the Erasure Threshold

As we showed in the code stability section, an LT code cannot be reliably transmitted over a BEC with erasure probability larger than the erasure threshold (i.e., when $\epsilon > \epsilon^*$). In this subsection, we set an upper bound on c parameter so that the

maximum erasure threshold can be achieved at fixed generation length k and code rate R .

Figures 3.10 and 3.11 illustrate the decoding process of an LT code with $k = 128$, $n = 256$, RSD, and erasure probability $\epsilon = 0.45$ using EXIT chart. As can be seen from the figures, the upper bound of c that can achieve the erasure threshold is $c = 0.035$.

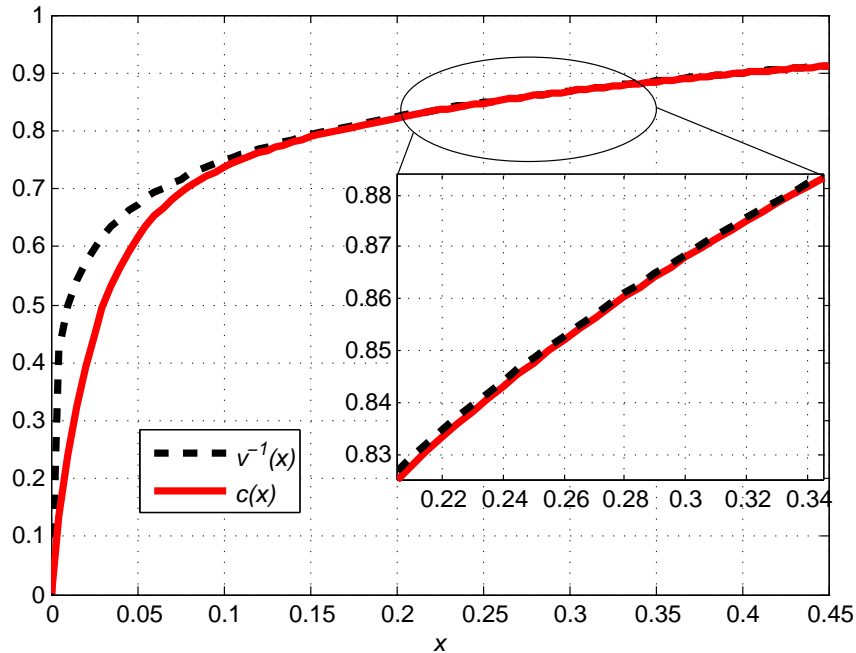


Figure 3.10: EXIT chart for an LT code with packet length $k = 128$ and code length $n = 256$ over a BEC of erasure probability $\epsilon = 0.45$. RSD degree distribution is used with parameters $c = 0.035$ and $\delta = 0.5$.

Table 3.2 shows the upper bound on the c parameter for LT codes with different packet lengths k and different code rates R . Also, the table shows the erasure threshold ϵ^* for each code. Note that, the erasure threshold is fixed at 90% of Shannon threshold (i.e., $1 - R$).

As can be seen in the table, the maximum c we can choose so that the erasure

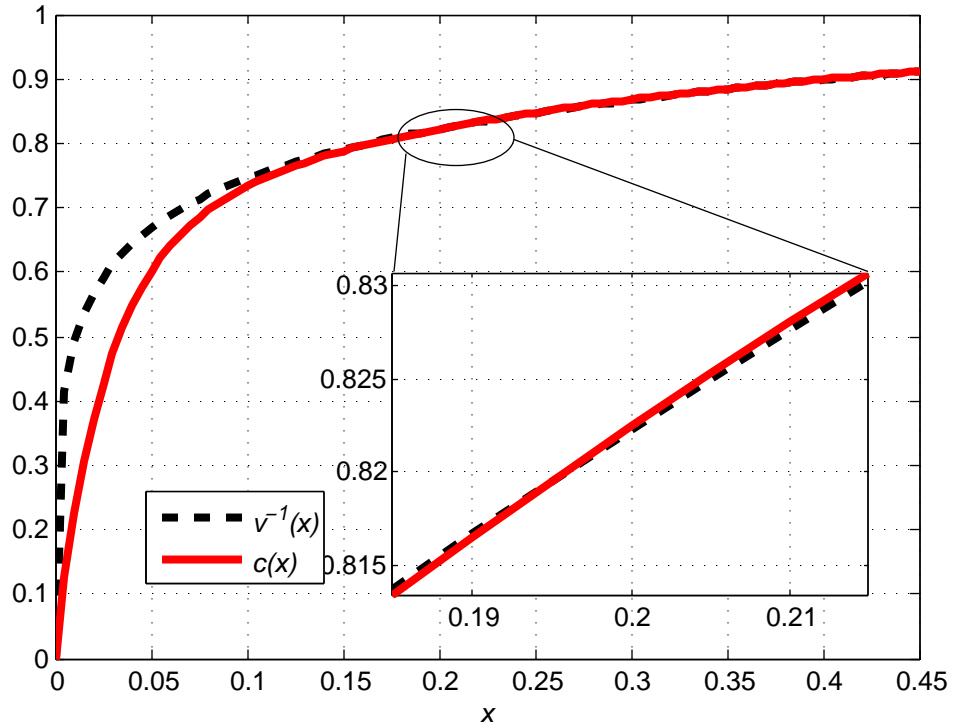


Figure 3.11: EXIT chart for an LT code with packet length $k = 128$ and code length $n = 256$ over a BEC of erasure probability $\epsilon = 0.45$. RSD degree distribution is used with parameters $c = 0.04$ and $\delta = 0.5$.

$k \backslash R$	$\frac{2}{3}$	$\frac{1}{2}$	$\frac{1}{3}$
128	0.025	0.038	0.075
256	0.025	0.038	0.075
512	0.025	0.038	0.075

Table 3.2: Upper bound on parameter c that can achieve the erasure threshold using EXIT chart for different LT codes. Each cell shows upper $c \setminus$ erasure threshold ϵ^* .

threshold is maximized is independent of the packet length k . However, this maximum c is increased when the code rate R is decreased.

3.5 Summary

In this chapter, we designed robust LT codes by optimizing the parameters of the degree distribution. The designed codes are used to maximize the code rate, minimize the erasure probability, and maximize the erasure threshold. Simulation results show that by choosing the best parameters c and δ , the performance of LT codes improves significantly. For example, at a source length $k = 128$, Shokrollahi distribution achieves code rate $R = 0.5632$ while our robust design achieves a code rate $R = 0.71081$, amounting to an improvement of 26%.

Chapter 4

Overlapped Fountain Codes

4.1 Introduction

During the past decade, much attention has been focused on designing good degree distributions for a variety of channels and applications. In the asymptotic regime¹, the optimal degree distribution for the BEC was derived by Luby in his original work. Robust soliton distribution (RSD) is *optimal* and *universal* in the sense of achieving the BEC capacity for any erasure rate [23]. However, for the finite-length regime, there have been many improvements to the LT code. These improvements have been reported as far back as the last decade and half.

Most of these works employed improved degree distributions schemes to outperform the RSD in terms of error rates and/or realized code rates [24, 25, 101–106]. All these contributions are classified under the umbrella of *soliton-like distributions* which provide an appropriate number of degree one, degree two, and maximum (*spike*) degree d_{max} [32, 35]. Zhu et al. [102] found the optimal degree distribution algorithm in terms of decreasing the overhead. However, this was impractical as the algorithm's

¹Practically when $k > 10^4$.

equations tend to be often unsolvable, especially, for large k . Then they proposed a suboptimal degree distribution that can be used in practical implementations. Next, Sorensen et al. [106] proposed a new algorithm based on a decreasing ripple size, and showed that their proposed code outperforms both the RSD and the suboptimal degree distribution proposed by Zhu. Sorensen et al. showed that the optimal degree distribution for LT codes should not exhibit a constant ripple as in the ISD and RSD. They suggested that the degree distribution should ensure a decreasing ripple size during the decoding process. More recently, we provided a new method to design better rateless systems via modifying the sampling method [107, 108]. The new technique is applicable to any degree distribution and when tuned, it outperforms LT and all other degree distributions known to date.

Fountain codes have been employed in the context of video broadcasting or media streaming in several papers. For instance, an LT codes based protocol has been proposed by Buyers et al. [20] targeting asynchronous video multicasting. The same concept was used but employing Tornado codes according to Byers et al. [109]. In addition, Raptor codes were used for streaming of stereoscopic video [110] as well as scalable video from multiple servers [111]. Furthermore, a protocol for live unicast video streaming based on rateless codes was introduced by Ahmad et al. [112, 113]. Their framework uses rateless codes and receiver feedback to protect video streaming. More specifically, the sender encodes the streaming data by using rateless codes and sends the encoding bits to a receiver until the sender receives an acknowledgment from the receiver confirming the completion of the transmission of a source block. The encoding bits are sent in separate bursts based on different time intervals and transmission rates. This scheme computes a block before sending the data; which

means it is sensitive to the acknowledgment path as an acknowledgment packet is required to verify that the sent block has been decoded correctly. While this scheme improves the bandwidth occupancy, it does not consider the delay. Similar ideas are described by Jenkac et al. [114], where the behavior of an asynchronous media streaming system based on fountain codes was investigated. In their work, the media stream is segmented (into small segments) and each segment is protected by a rateless code. Nevertheless, the small segment makes the code inefficient in terms of coding rate.

Other works in the literature address unequal error protection (UEP), where some of the source bits are given more protection (i.e., higher encoding priority) than others. For example, an algorithm was proposed to implement scalable data multicast to different user groups, assuming that the data are ordered according to their importance [115]. Embedded windows of increasing length are defined and used with different probabilities so that the most significant bits are included in parity checks with a high probability. The same principle was also reported in [116], where an expanding window of source bits was defined so that the most significant source bits are included in the innermost window and can be decoded with higher probability. Similar ideas were proposed by Cataldi et al. [117] where the concept of a sliding window is given to provide UEP using Raptor codes. Also, according to Bouabdallah et al. [118], the principle of dependency-aware UEP was proposed, where the existing data dependencies among I, P, and B video frames were exploited to achieve differentiated protection, and different erasure protection codes were allocated according to their impact on the reconstruction quality. Applications of Raptor codes to scalable video coding (SVC) were reported by Hellge et al. [119, 120] where redundancy bits

of a given layer are obtained using all the layers.

In broadcast applications, there are many users with different channel conditions, devices, distances, and quality requirements. The sender has to consider all of these differences and guarantee a QoS for all them. Using conventional codes (e.g., LDPC codes) in this situation, the code rate is either too high or too low. On the one hand, if the code rate is low, excessive redundant data will be received (i.e., bandwidth wastage). On the other hand, if the code rate is high, a reliable communication channel between the transmitter and the receiver is not guaranteed (i.e., system outage). Besides, as fountain codes do away with the traditional retransmissions in response to not acknowledged (NAK) messages, significant reductions in latency is also expected.

Primarily, data must be parsed into generations to allow for a practical system. In traditional fountain codes, generations are encoded and sent sequentially where the transmitter moves up from a generation l to $(l + 1)$ when all or the majority of the intended users have fully decoded or satisfied some other acknowledge (ACK) conditions. In large systems, users will experience a wide range of channel conditions, and as a result, while some users are awaiting the next generation, the rest might be in need of more encoded packets to satisfy their ACK requirements. In order to better balance the needs of all system users, we precisely propose the concept of overlapped fountain codes.

The rest of this chapter is structured as follows. First, we present the concept of overlapped fountain codes. In particular, we discuss overlapped LT (OLT) codes over the BEC. Then, we show the convergence speed of OLT codes analytically and compare it with conventional LT codes. In addition, we show the improvements of OLT codes over conventional LT codes in terms of the realized code rate and

complexity. Finally, we optimize system parameters depending on maximizing the realized code rate or minimizing complexity.

4.2 Overlapped LT (OLT) Codes

We propose a solution suitable for data composed of portions with different *priority* such as those motivated by priority encoding transmission (PET) schemes [121] and multi-level coding (MLC) schemes [122, 123]. In general, each generation is subdivided into a number of partitions, each partition with a different priority. The first partition has the source bits $\{u_1, u_2, \dots, u_m\}$ with the highest priority for encoding, and we refer to it as partition π_1 . Partition π_2 includes bits $\{u_{m+1}, u_{m+2}, \dots, u_{2m}\}$ and so on. There are a total of $\frac{k}{m} = \nu$ partitions in which π_ν is the least important partition.

By allowing for the overlap of generations intelligently, more flexibility in the system design is provided. This, as we will show later, will lead to better and wider range of performance-complexity trade-offs. In particular, we do not need the entire generation to be recovered before we can move on to the next generation. We implement this by allowing the last α bits of each generation to coincide with the first α bits of the following generation. We allow for α to encompass the first γ partitions of the following generation, that is, $\alpha = \gamma.m$. This is referred to as a $(K, k, \alpha, m, \gamma)$ OLT code. Our proposal is equally applicable to any degree distribution; we focus on the RSD in what follows. Equivalent returns are expected for any other degree distribution.

For simplicity, we choose only two partitions (i.e., $\alpha = m$ and $\gamma = 1$). Therefore, the OLT encoder divides the source data of size K bits into g overlapped partitions

each of size k bits. The overlap size α is given by

$$\alpha = \left\lceil \frac{g \cdot k - K}{g - 1} \right\rceil, \quad (4.1)$$

where $\lceil \cdot \rceil$ is the ceiling function.

The overlap size α must be chosen such that it reduces the complexity as much as possible while getting the highest benefit from the concept of overlapped generations. That is bounded by $0 \leq \alpha \leq k - 1$. A particular case when $\alpha = 0$, results in a conventional LT code. Figure 4.1 illustrates the required number of generations g to cover the whole source data versus the percentage of the overlap (i.e., α/k) at different generation sizes while the size of the source data is fixed $K = 1024$ bits.

As the overlapped percentage α/k (or equivalently the overlap size α) increases, the required number of generations g to cover all the source data increases as well, which means the complexity is increased in some way. We see that the performance is deferent in two regions. The first region is where the overlapped percentage ranges from 0% to 50%. In this region, we can observe that the relation between g and α/k is almost linear and moderate. The second region is where the overlapped percentage ranges from 50% to around 100% (it is exactly at $\alpha = k - 1$). This region has non-linear and rapidly-increasing behaviour. Thus, the overlapped percentage (or the overlap size) from the linear region is adopted in this thesis. Within the linear region, some of the source bits will be covered only in one generation, which means less protection at the decoding process. The size of these source bits (with less protection) increases as the overlapped percentage decreases. Thus, we choose the highest overlapped percentage in the linear region. Fortunately, at $\alpha/k = 50\%$, all

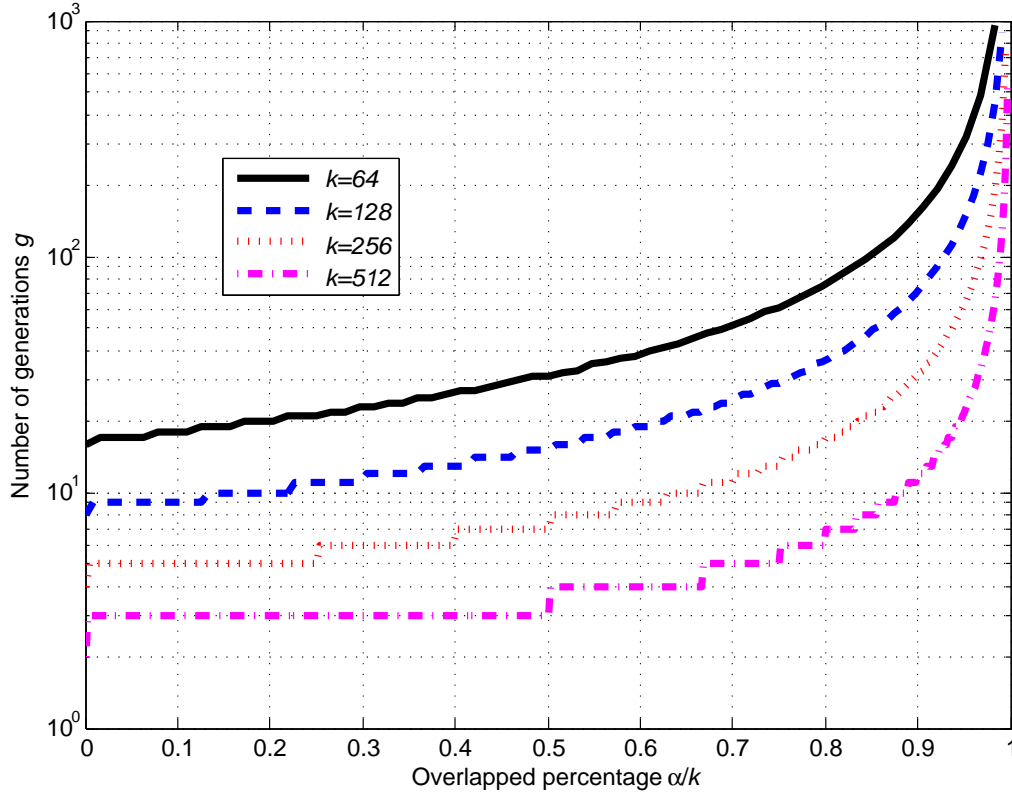


Figure 4.1: The required number of generations g to cover all the source data $K = 1024$ bits versus the overlapped percentage α/k at different generation sizes k .

the source bits are covered by exactly two generations². Thus, we argue that the overlapped percentage $\alpha/k = 50\%$ ($\alpha = k/2$) is the best and simplest choice.

The next two subsections address the encoding and decoding processes of OLT codes.

²Note that, the first $k - \alpha$ and the last α source bits are covered one time in any situation.

4.2.1 Encoding Process of OLT Codes

After we choose the best overlap size α at the transmitter, we divide the source data K , into g overlapped generation depending on the selected α . Each generation of size k source bits $\{u_1, u_2, u_3, \dots, u_k\}$ is assumed to have two partitions: the first partition π_1 has the first $(k - \alpha)$ source bits $\{u_1, u_2, \dots, u_{k-\alpha}\}$ while the second partition π_2 has the last α source bits $\{u_{k-\alpha+1}, u_{k-\alpha+2}, \dots, u_k\}$. Then, the transmitter applies Algorithm 3 to generate and broadcast the encoded bits. In Algorithm 3, we set $p > 0.5$ to prime the first partition.

Algorithm 3 Encoding OLT Codes

```

1: Input: Source bits  $\{u_1, u_2, u_3, \dots, u_K\}$ , selection probability  $p$ , and overlap size
    $\alpha$ 
2: Divide the source data into  $g$  generations  $g = (K - \alpha)/(k - \alpha)$ 
3: for  $l = 1$  to  $g$  do
4:   Divide the source bits in the  $l$  generation into two partitions,  $\pi_1 =$ 
    $\{u_1, u_2, u_3, \dots, u_{k-\alpha}\}$  and  $\pi_2 = \{u_{k-\alpha+1}, u_{k-\alpha+2}, u_{k-\alpha+3}, \dots, u_k\}$ 
5:    $i=1$ 
6:   while Not all receivers communicate their ACKs do
7:     Choose a degree  $d$  from the RSD distribution
8:     for  $s = 1$  to  $d$  do
9:       Randomly choose a parameter  $x$ , where  $0 \leq x \leq 1$ 
10:      if  $x < p$  then
11:        Select the first partition  $\pi_1$ 
12:      else
13:        Select the second partition  $\pi_2$ 
14:      end if
15:      Choose a source bit, uniformly at random, from the selected partition
16:    end for
17:    Perform XOR on the  $d$  chosen source bits in the previous steps  $c_i = \bigoplus_{j=1}^d U_j$ ,
   where  $U$  is a subset of the  $k$  source bits containing only the chosen  $d$  source bits
18:    Broadcast the encoded bit  $c_i$ 
19:     $i++$ 
20:  end while
21: end for

```

In Algorithm 3, each encoded bit is generated using the source bits from the first partition π_1 and second partition π_2 with probability p and $(1 - p)$, respectively. These (chosen) source bits are XORed to generate the encoded bit. The encoder keeps generating the encoded bits until it receives an ACK from each and every receiver. Once the encoder receives these ACKs, it moves forward to the next generation of the source data.

4.2.2 Decoding Process of OLT Codes

Each receiver will collect a different set of encoded bits according to its channel conditions. Once a receiver collects enough number of encoded bits, it starts the decoding process as shown in Algorithm 4. In OLT codes, the sufficient number of encoded bits n is slightly larger than $(k - \alpha)$ compared to k in the conventional LT codes. Thus, the OLT decoder starts the decoding process earlier which means reductions in latency. Because we prime the first partition in each generation by setting $p > 0.5$, the first partition will be decoded faster/earlier than the second partition. Once the decoder fully recovers the first partition, it sends an ACK to the transmitter. Starting from the second generation, the decoder benefits from the concept of overlapped generations; some of the source bits will be already recovered prior receiving any encoded bits of the current generation. These recovered source bits will help in the decoding process of the current generation. Another advantage of having recovered source bit beforehand is that the degree-one encoded bits are not needed to initiate the decoding process as in the case of conventional LT codes.

Algorithm 4 Decoding OLT Codes

- 1: **Input:** Encoded bits $\{c_1, c_2, c_3, \dots\}$ and overlap size α
 - 2: **while** The first $(k - \alpha)$ source bits are not recovered **do**
 - 3: Identify all degree-one encoded bits c_i and add them to the *ripple* Ψ .
 - 4: **if** $\Psi = \{\}$ **then**
 - 5: The decoder decides decoding failure and it requires more encoded bits
 - 6: **else**
 - 7: Choose one encoded bit c_j such that $c_j \in \Psi$.
 - 8: The (only) neighbour of c_j ; i.e., source bit u_i , is immediately recovered
(i.e., $u_i = c_j$).
 - 9: Remove the edge which connects u_i to c_j .
 - 10: Update all the neighbor encoded bits of the recovered source bit u_i in the
previous step
 - 11: Remove all edges of the recovered source bit u_i .
 - 12: **end if**
 - 13: **end while**
 - 14: Send an ACK to the transmitter
-

4.3 Performance Analysis

To analyze the proposed scheme, the ideal approach would be to find error expressions. However, these have been proven to be hard problems. Instead, one can study the *ripple* behavior of a given rateless code [106, 124, 125]. The dynamic behaviour of the decoding ripple is intimately related to the error performance, overhead, and complexity of a code. In the BEC case, a related feature of the ripple is the rate with which the edges of the decoding graph are removed. In what follows, we analyze the expected number of removed edges (ENRE) during the traditional BP decoding.

We denote the ENRE at a decoding step j as $\varphi(j)$. In our analysis, we assume that all the degree-one encoded bits are processed simultaneously (i.e., at the same decoding step). Also, each decoding step consists of messages (beliefs) passing from encoded bits toward source bits as well as messages (beliefs) passing from source bits toward encoded bits.

In the following, we find $\varphi(j)$ for conventional LT codes as well as OLT codes.

4.3.1 Conventional LT Codes

In this subsection, we derive the ENRE $\varphi(j)$ of conventional LT codes.

At the first decoding step, $\varphi(j = 1) = \Omega(1) \cdot n$, where $\Omega(1)$ is the probability of degree one. Then at the second step,

$$\varphi(2) = \Omega(1) \cdot n \left(\frac{n \sum_{i=1}^k i \cdot \Omega(i)}{k} - 1 \right), \quad (4.2)$$

where $\left(\frac{n \sum_{i=1}^k i \cdot \Omega(i)}{k} \right)$ is an average number of edges to the source bits.

At the third step $j = 3$, the ENRE is determined as

$$\varphi(3) = \sum_{j=2}^{\min(k, \lceil \Omega(1) \cdot n \rceil)} \Omega(j) \cdot n \frac{\binom{\lceil \Omega(1) \cdot n \rceil}{j-1} \binom{k - \lceil \Omega(1) \cdot n \rceil}{1}}{\binom{k}{j}}, \quad (4.3)$$

where $\frac{\binom{\lceil \Omega(1) \cdot n \rceil}{j-1} \binom{k - \lceil \Omega(1) \cdot n \rceil}{1}}{\binom{k}{j}}$ is a probability that a degree- j encoded bit is reduced to a degree-one at the previous step.

For $j \geq 4$, we can generalize the form as follows:

If j is even, then

$$\varphi(j \geq 4, j \text{ even}) = \varphi(j-1) \cdot (\mu(j) - 1), \quad (4.4)$$

where $(\mu(j) - 1)$ is the average number of edges to the source bits k at the decoding

step j and

$$\mu(j) = \frac{n \sum_{i=1}^k i \cdot \Omega(i) - \sum_{s=1}^{j-1} \varphi(s)}{k}. \quad (4.5)$$

If j is odd, then

$$\varphi(j \geq 4, j \text{ odd}) = \sum_{j=2}^{\min(k, \lceil \gamma(j) \rceil)} n \cdot \Omega(j) \frac{\binom{\lceil \gamma(j) \rceil}{j-1} \binom{k - \lceil \gamma(j) \rceil}{1}}{\binom{k}{j}}, \quad (4.6)$$

where $\gamma(j)$ is the number of recovered source bits up until step j and is given by

$$\gamma(j) = \sum_{s=1, s \text{ odd}}^{j-1} \varphi(s). \quad (4.7)$$

4.3.2 OLT Codes

As described in Section 4.2, the first partition π_1 contains the first $(k - \alpha)$ source bits while the second partition π_2 contains the last α source bits. In the considered BP decoder, it is assumed that there are also two types of encoded bits according to the probability of choosing the source bits. For example, if we choose the first partition π_1 of source bits with probability p and the second partition π_2 with probability $(1 - p)$, the encoded bits can be divided into two parts: $n_1 = n \times p$ for the first partition and $n_2 = n \times (1 - p) = n - n_1$ for the second partition. Thus, the ENRE at the j -th step, $\varphi(j)$, can be calculated as follows:

At the first decoding step ($j = 1$), the ENRE is given by

$$\varphi(1) = \Omega_1 n_1 + \Omega_1 n_2 = \Omega_1 (n_1 + n_2) = \Omega_1 n. \quad (4.8)$$

Then, at the second step, $\varphi(j)$ is given by

$$\begin{aligned} \varphi(2) = & \Omega(1) \cdot n_1 \left(\frac{n_1 \sum_{i=1}^{\pi_1} i \cdot \Omega(i)}{\pi_1} - 1 \right) \\ & + \Omega(1) \cdot n_2 \left(\frac{n_2 \sum_{i=1}^{\pi_2} i \cdot \Omega(i)}{\pi_2} - 1 \right), \end{aligned} \quad (4.9)$$

where $\left(\frac{n_m \sum_{i=1}^{\pi_m} i \cdot \Omega(i)}{\pi_m} \right)$, $m = \{1, 2\}$, is the average degree of the source bits from partition π_m .

At the third step $j = 3$, the ENRE is determined as

$$\begin{aligned} \varphi(3) = & \sum_{j=2}^{\min(\pi_1, \lceil \Omega(1) \cdot n_1 \rceil)} \Omega(j) \cdot n_1 \frac{\binom{\lceil \Omega(1) \cdot n_1 \rceil}{j-1} \binom{\pi_1 - \lceil \Omega(1) \cdot n_1 \rceil}{1}}{\binom{\pi_1}{j}} \\ & + \sum_{j=2}^{\min(\pi_2, \lceil \Omega(1) \cdot n_2 \rceil)} \Omega(j) \cdot n_2 \frac{\binom{\lceil \Omega(1) \cdot n_2 \rceil}{j-1} \binom{\pi_2 - \lceil \Omega(1) \cdot n_2 \rceil}{1}}{\binom{\pi_2}{j}}, \end{aligned} \quad (4.10)$$

where $\frac{\binom{\lceil \Omega(1) \cdot n_m \rceil}{j-1} \binom{\pi_m - \lceil \Omega(1) \cdot n_m \rceil}{1}}{\binom{\pi_m}{j}}$, $m = \{1, 2\}$, is a probability that a degree- j encoded bit is converted to a degree-one at the previous step in partition π_m .

For $j \geq 4$, we can generalize the form as follows:

If j is even, then

$$\varphi(j \geq 4) = \varphi(j-1) \cdot (\mu(j) - 1), \quad (4.11)$$

where $(\mu(j) - 1)$ is the average number of edges to the source bits k at step j and

$$\begin{aligned} \mu(j) = & \frac{n_1 \sum_{i=1}^{\pi_1} i \cdot \Omega(i) - \sum_{s=1}^{j-1} \varphi(s)}{\pi_1} \\ & + \frac{n_2 \sum_{i=1}^{\pi_2} i \cdot \Omega(i) - \sum_{s=1}^{j-1} \varphi(s)}{\pi_2}. \end{aligned} \quad (4.12)$$

If j is odd, then

$$\begin{aligned} \varphi(j \geq 4) = & \sum_{j=2}^{\min(\pi_1, \lceil \gamma_1(j) \rceil)} n_1 \cdot \Omega(j) \frac{\binom{\lceil \gamma_1(j) \rceil}{j-1} \binom{\pi_1 - \lceil \gamma_1(j) \rceil}{1}}{\binom{\pi_1}{j}} \\ & + \sum_{j=2}^{\min(\pi_2, \lceil \gamma_2(j) \rceil)} n_2 \cdot \Omega(j) \frac{\binom{\lceil \gamma_2(j) \rceil}{j-1} \binom{\pi_2 - \lceil \gamma_2(j) \rceil}{1}}{\binom{\pi_2}{j}}, \end{aligned} \quad (4.13)$$

where $\gamma_m(j)$, $m = \{1, 2\}$, is the number of recovered source bits from partition π_m up until step j and it is given by:

$$\gamma_m(j) = \sum_{s=1, s \text{ odd}}^{j-1} \varphi_m(s), \quad (4.14)$$

where $\varphi_m(s)$ is the m -th term of the odd steps s of $\varphi(s)$. We repeat this process for the other generations $\{2, \dots, g\}$, where g is the total number of generations. Then, we average the expected number of removed edges over g .

4.4 Numerical Results

In this section, the proposed OLT codes are constructed in comparison to conventional LT codes. Several simulation experiments are presented to show the improvements of OLT codes over conventional LT codes.

The performance of the proposed scheme is compared to that of conventional LT codes for $K = 1024$. The RSD parameters are $c = 0.02$ and $\delta = 0.1$. To show the performance over a wide range of conditions, we consider a broadcast scenario to five users with erasure probabilities of $\epsilon = \{0.3 \ 0.2 \ 0.1 \ 0.01 \ 0.001\}$ as shown in Figure 4.2. We have picked a 4 to 1 priority for the π_1 and π_2 partitions; that is $p = 0.8$. Also, we use OLT codes as illustrated in Figure 4.3.

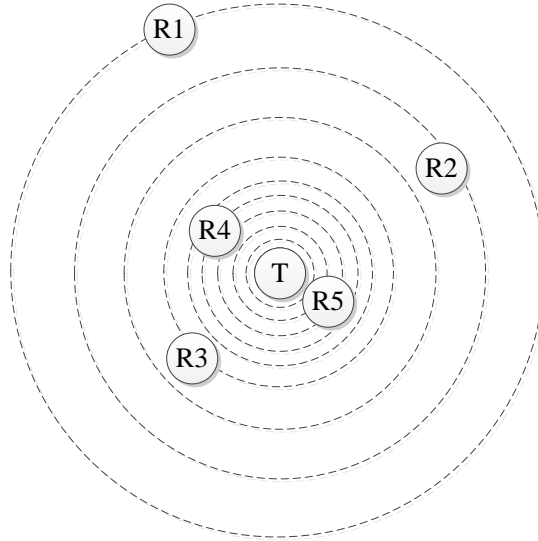


Figure 4.2: Broadcasting to five receivers with BEC's of erasure probabilities $\epsilon = \{0.3 \ 0.2 \ 0.1 \ 0.01 \ 0.001\}$.

Figure 4.4 shows the resulting coding rate R versus erasure probability ϵ for three

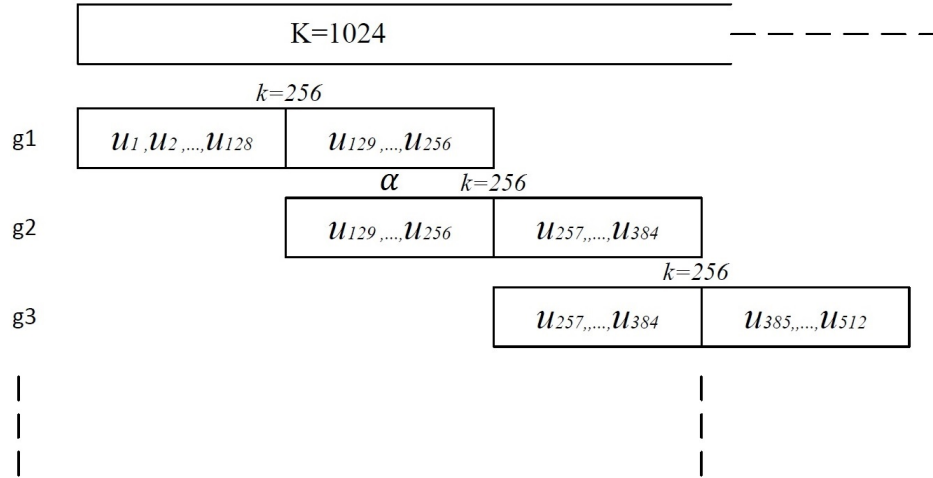


Figure 4.3: OLT codes at $K = 1024$, $k = 256$, and $\alpha = 128$.

schemes. The first scheme is the conventional LT code with a generation size of $k = 256$. In this scheme, we need four generations to send the whole data set. The second scheme is the overlapped LT codes (1024, 256, 128, 128, 1) where we need $g = 7$ overlapped generations to cover all the source bits. The third scheme is the conventional LT code with a generation size of $k = 1024$ where we need only one generation for the entire sequence. Our novel scheme performs at tantamount realized rates as the LT code with $k = 1024$ while it affords significant reductions in complexity as shown in Figure 4.5.

Figure 4.5 shows the encoding/decoding complexity for the above-mentioned three schemes. The complexity is measured in terms of the average number of edges per bit. As can be seen in the figure, the conventional LT code at $k = 1024$ has the highest encoding complexity no matter what the channel characteristic is. Also, the conventional LT code at $k = 256$ has the lowest complexity. However, our proposed scheme lies between these two conventional LT schemes. In addition, for a channel with high erasure probability, our proposed scheme is approaching the performance

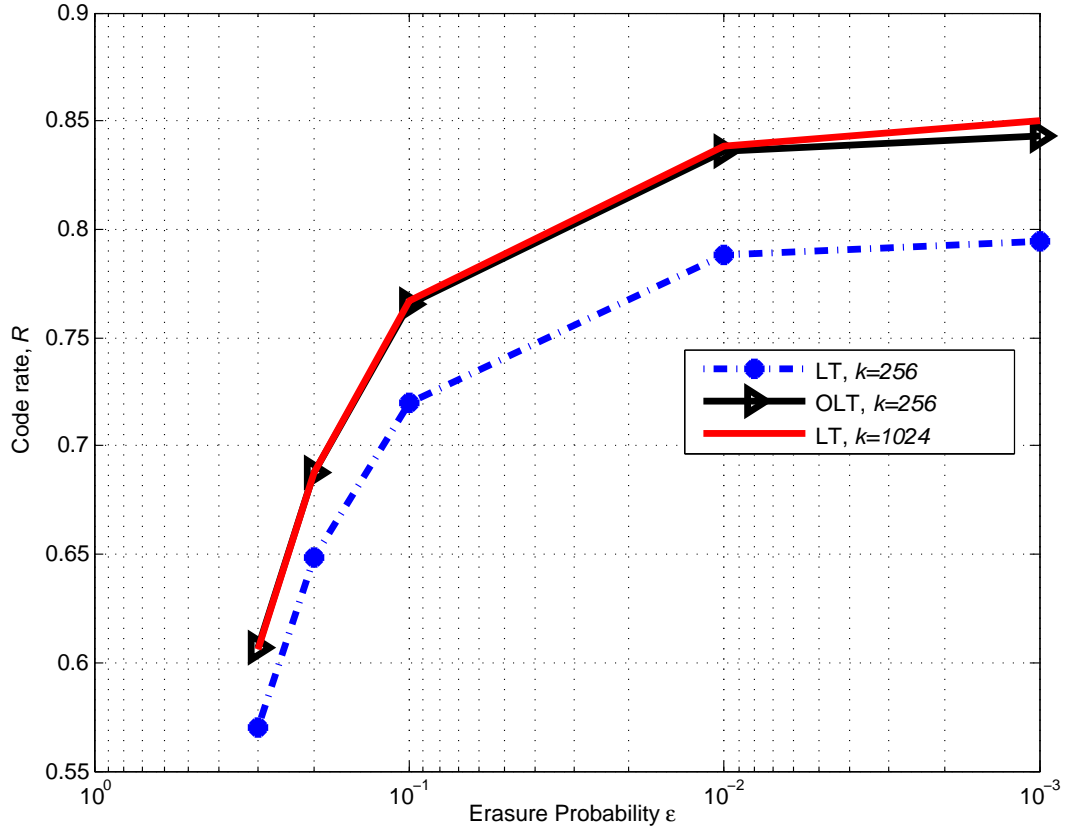


Figure 4.4: Code rate R of five users with different erasure probabilities using different schemes.

of conventional LT code at $k = 256$.

From the previous two figures, we can observe that our proposed scheme reaches the performance of conventional LT code at $k = 1024$ in terms of code rate while reducing the complexity of conventional LT codes at $k = 1024$ to get closer to the complexity of conventional LT codes at $k = 256$. Moreover, our proposed scheme reduces the delay time for each receiver. Equivalent returns can be achieved for other

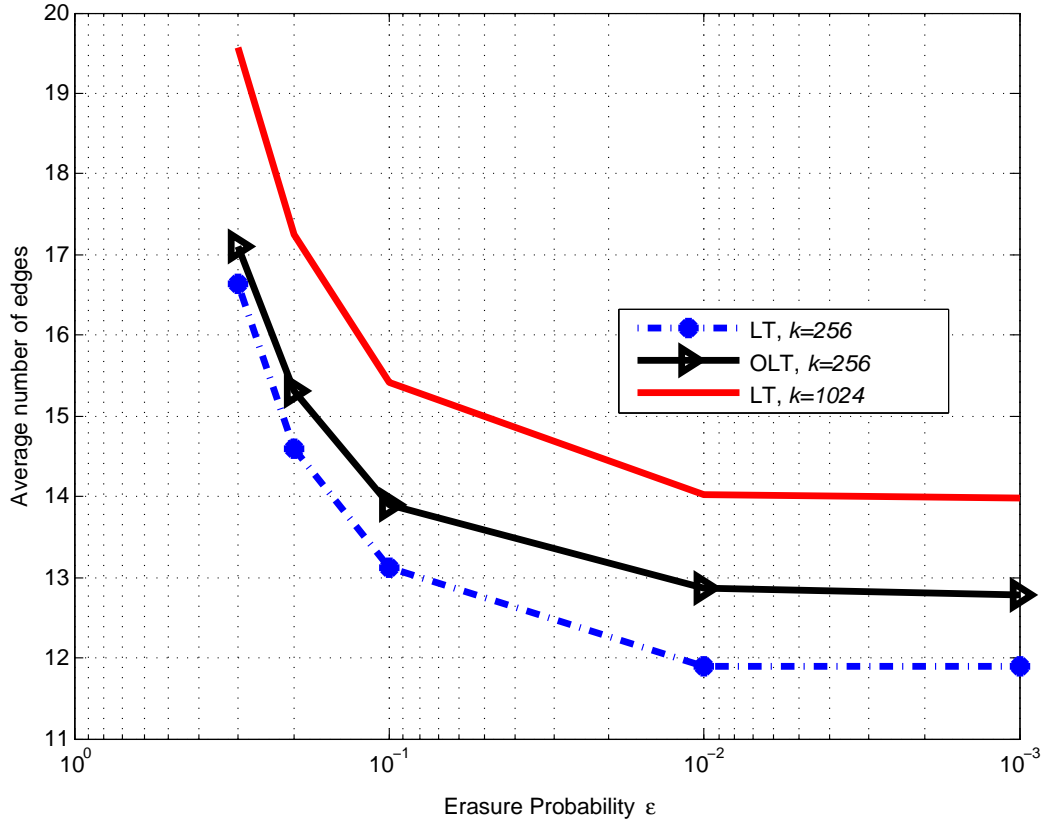


Figure 4.5: Average number of edges versus erasure probability for different schemes.

sets of parameters.

In the following, we use the ENRE derived in Section 4.3 to show more improvements of OLT codes over conventional LT codes. We use data source of $K = 1024$ bits. The source data is divided into $g = 7$ overlapped generations each of size $k = 256$. Thus, the overlap size is $\alpha = 128$. We set the selection probability of the first partition as $p = 0.65$. We used OLT codes with parameters as shown in Figure 4.3.

Note that as shown in the analytical results in Section 4.3, the code rate is fixed (number of encoded bits n is fixed) at the first step and so on. Besides, we calculate each decoding step as forward (which has an odd number) and backward (which has an even number) steps of the decoding.

Figure 4.6 depicts the number of removed edges at each decoding step of LT and OLT codes at different code rates. These curves are obtained using the analytical results in Section 4.3. Three advantages of OLT codes over conventional LT codes can be clearly observed from the figure: 1) OLT codes start with a higher number of removed edges at the first decoding step. For instance, with a code rate $R = 0.5$, the OLT decoder removes 423 edges at the first decoding step comparing with only 123 edges in the case of the conventional LT decoder. 2) OLT codes require less number of decoding steps at the same code rate of conventional LT codes. For example, OLT codes need 4 decoding steps to complete the decoding process while LT codes need 5 decoding steps at a code rate $R = 0.5$. 3) The total number of edges in OLT codes is less than that in conventional LT codes. For example, at code rate $R = 0.75$, the total number of edges in OLT codes is 1989 edges compared to 2580 edges in LT codes; that is a reduction of 23% in the graph complexity (i.e., encoding and decoding complexities).

In practice, OLT codes improve the performance of conventional LT codes in terms of code rate. Thus, one must show their decoding speed in the rateless criteria (i.e., at zero error probability). Figure 4.7 depicts the simulation results of LT and OLT codes at zero error probability. Equivalent advantages as in Figure 4.6 can be observed. For instance, OLT codes start with an average of 560 removed edges at the first decoding step while LT codes remove only 60 edges at the same step. Moreover,

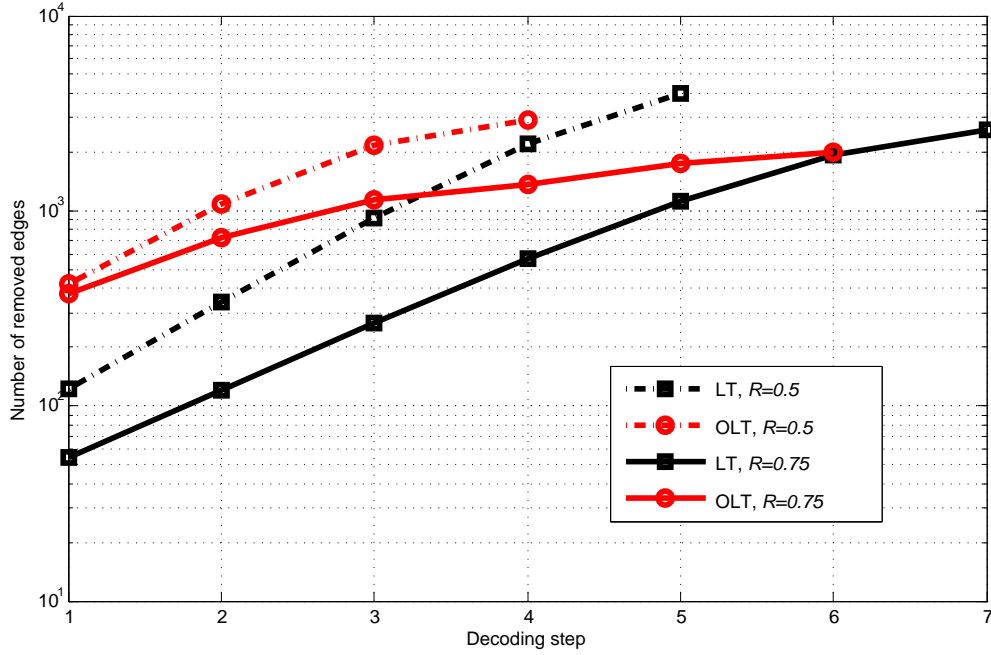


Figure 4.6: Analytical comparison of the number of removed edges at each decoding step between LT and OLT codes with different code rates at $K = 1024$, $k = 256$, $\alpha = 128$, and $p = 0.65$.

the total number of edges is smaller in OLT codes (2639 edges) compared to LT codes (3053 edges). Finally, OLT codes need around 25 decoding steps, on average, for fully decoding while conventional LT codes require around 35 decoding steps; that is an improvement of 29%.

After we show the performance improvements of our OLT codes over conventional LT codes, we now focus on optimizing the parameters of OLT codes, specifically the selection probability p .

Figure 4.8 depicts the average complexity (on the left) and average code rate R (on the right) versus the selection probability p at zero erasure probability based on parameters shown in Figure 4.3. Note that the average complexity is defined as

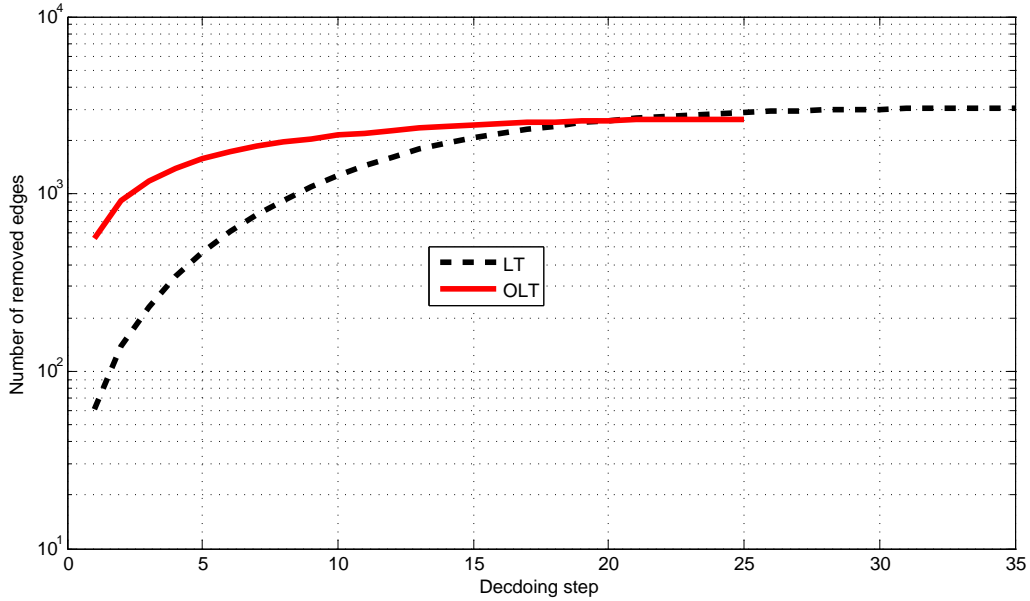


Figure 4.7: Simulation comparison of the number of removed edges at each decoding step between LT and OLT codes with zero erasure probability at $K = 1024$, $k = 256$, $\alpha = 128$, and $p = 0.65$.

the total number of edges per each source bit. Therefore, we have two observations: 1) The code rate R (dashed curve) has the highest value (0.8642) at the selection probability of $p = 0.6$. 2) The lowest average complexity (10.46) is at the selection probability $p = 0.6$.

For different erasure probability, i.e., $\epsilon = 0.1$, Figure 4.9 shows the average complexity and average code rate. As can be seen in the figure, the highest code rate $R = 0.7889$ and the lowest complexity 10.33 are achieved at selection probability of $p = 0.6$. These findings are similar to the previous observations. As a result, our design for the selection probability p is universal and robust to the channel characteristics (i.e., erasure probability)

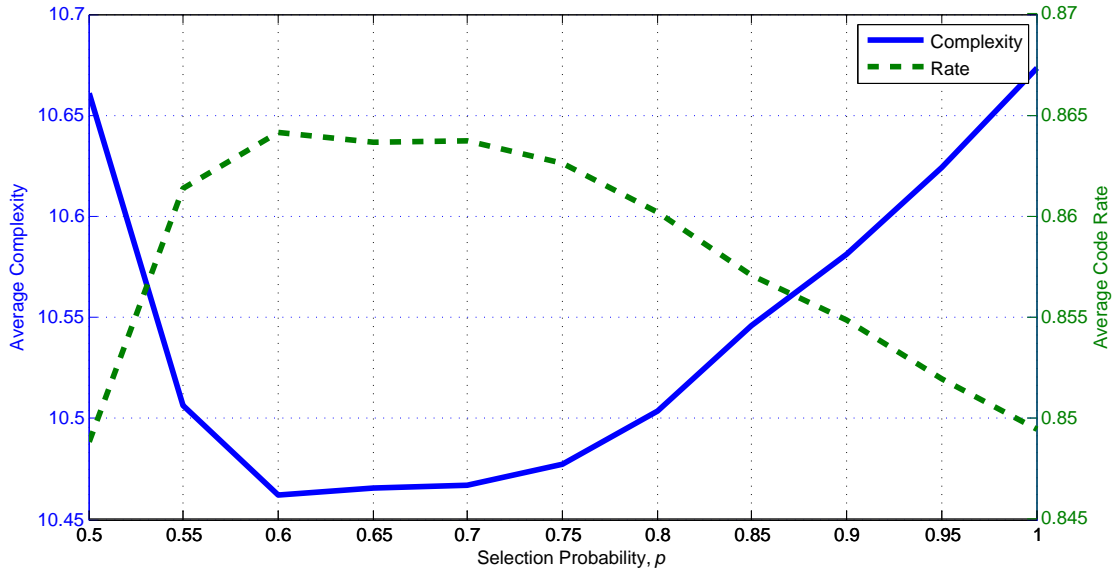


Figure 4.8: Average complexity (right) and average code rate (left) of OLT codes versus selection probability p at zero erasure probability $\epsilon = 0$, $K = 1024$, $k = 256$, and $\alpha = 128$.

4.5 Summary

We proposed a new rateless coding scheme suitable for broadcast scenarios with a wide range of channel conditions. Our proposed scheme uses the concept of overlapped generations for LT codes; it is equally applicable to any other coding distributions. The proposed scheme improves the performance in terms of convergence speed, code rate and latency. We showed via analytical and simulation methods these improvements. The cost of this scheme is a slight increase in the computational complexity. For instance, for a channel with erasure probability of $\epsilon = 0.01$ and packet length $k = 256$, the improvement in terms of the coding rate is 6.1% at an average cost of an extra edge in the factor graph. When we compared this result with the performance of LT codes at $k = 1024$, the proposed scheme has almost the same coding rate, however,

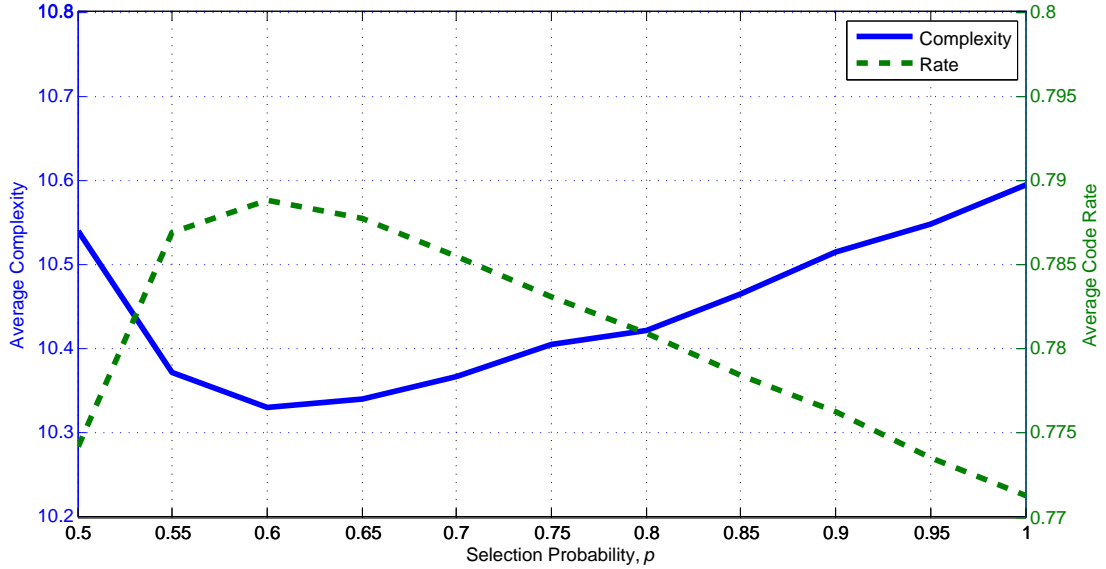


Figure 4.9: Average complexity (right) and average code rate (left) of OLT codes versus selection probability p at erasure probability $\epsilon = 0.1$, $K = 1024$, $k = 256$, and $\alpha = 128$.

LT code at $k = 1024$ has an extra one symbol over our scheme (two symbols over LT codes at $k = 256$) on average in the encoding complexity. However, the analytical results show that OLT code converges faster than conventional LT code, and it needs less number of iteration for full decoding. Also, we optimized the selection probability used in OLT codes. At source length $k = 256$ and overlap size $\alpha = 128$, the simulation results show that the optimum selection probability $p = 0.6$ in terms of maximizing the code rate and minimize the decoding complexity.

Chapter 5

OLT Codes over AWGN Channels

5.1 Introduction

It has been proven that good codes over the BEC are also good codes for many practical channels including, but not limited to, AWGN channels and BSC [71, 87, 126]. Therefore, our proposed ‘overlapped generations’ concept is applied over AWGN channels in this chapter. As expected, improvements over conventional LT codes are obtained. In addition, using overlapped generations reduces the delay time at each decoder. The cost of these improvements is a marginal increase in the complexity. While we focus on the AWGN channel, it should be noted that the proposed encoding scheme can be used in other channels such as BSC and fading channels to improve the realized code rate and/or the BER.

Over AWGN channel, several works have shown that *systematic LT codes* are better than nonsystematic LT codes at different channel conditions and different code rates [107, 127–130]. Furthermore, Tran et al. [130] proved that the percentage of systematic connections of LT codes depends on the number of BP iterations. As the

total number of BP increases, the required percentage of systematic connections decreases. Another benefit of using systematic connection is obtained when the channel is perfect or has few lose. Therefore, we tend to focus our attention on systematic LT codes in this chapter.

Definition 5.1: Systematic codes

A code C is called a systematic code if and only if the output encoded bits explicitly contain the source bits. The generator matrix G of a systematic code is written as $G = [I_k|P]$, where I_k and P are the identity matrix of size k and the parity check matrix, respectively.

5.2 System Model over AWGN Channels

We consider the transmission of a block of information of size B bits through a BIAWGN channel. The set $\{0, 1\}$ is mapped to ± 1 levels with antipodal signaling for transmission in zero-mean white Gaussian noise with a two-sided power spectral density of $\sigma^2 = \frac{N_0}{2}$ W/Hz. The entire data is first parsed into *generations* each of size k bits: we will have $l = \lceil \frac{B}{k} \rceil$ generations each requiring its own session.

In what follows, we review the systematic LT encoding and decoding over BI-AWGN channel.

5.2.1 Encoding Systematic LT Codes

For a given generation with k source bits $V = (v_1, v_2, \dots, v_k)$, the LT encoder generates a potentially limitless output stream $U = (u_1, u_2, \dots)$. For the systematic implementation, the encoder first relays (v_1, v_2, \dots, v_k) followed by the parity bits encoded as in

Algorithm 5.

Algorithm 5 LT Encoding

1. Choose a degree d by sampling the RSD Ω_d .
2. Choose d source bits uniformly at random from the k available.
3. Perform bitwise XOR on the chosen d source bits; this is the output bit to be transmitted.

$$u_i = v_{i_1} \oplus v_{i_2} \oplus \dots \oplus v_{i_d}. \quad (5.1)$$

4. Repeat.
-

By repeating the steps (in Algorithm 5) in an i.i.d. manner, the output bits are generated one at a time. The output sequence will be $U = (v_1, v_2, \dots, v_k, u_{k+1}, u_{k+2}, \dots)$.

For an LT code with code rate of $R = k/n$, where n is the total number of received encoded bits, the encoded bits $U = (u_1, u_2, \dots, u_n)$ are binary phase shift keying (BPSK) modulated to produce a vector of modulated bits $X = (x_1, x_2, \dots, x_n)$, where $x_i = (-1)^{u_i}$. Then, the modulated bits, X , are sent over an AWGN channel.

5.2.2 Decoding over AWGN Channels

The received bits $Y = (y_1, y_2, \dots, y_n)$ can be expressed as $y_i = x_i + \eta_i$, where η_i is an i.i.d. AWGN sample with zero mean and variance $\sigma^2 = \frac{N_0}{2}$ W/Hz. Therefore, we can obtain the following conditional pdfs using Equation (2.4).

$$p(Y = y_i | x_i = -1) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y+1)^2}{2\sigma^2}}. \quad (5.2)$$

$$p(Y = y_i | x_i = +1) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y-1)^2}{2\sigma^2}}. \quad (5.3)$$

We assume that the soft inputs come from a maximum-likelihood (ML) detector

computing LLR for each bit from the received sequence Y . As a result, the LLR for each received signal y_i (or equivalently the LLR for each output bit in the TG) is given by

$$\begin{aligned} \text{LLR}(u_i) &= \ln \frac{p(v_i = 0|y_i)}{p(v_i = 1|y_i)} \\ &= \ln \frac{p(x_i = +1|y_i)}{p(x_i = -1|y_i)}. \end{aligned} \quad (5.4)$$

Next, we use Bayes' rule

$$p(x_i|y_i) = p(y_i|x_i)p(x_i)/p(y_i), \quad (5.5)$$

we get

$$\begin{aligned} \text{LLR}(u_i) &= \ln \frac{p(y_i|x_i = +1)p(x_i = +1)/p(y_i)}{p(y_i|x_i = -1)p(x_i = -1)/p(y_i)} \\ &= \ln \frac{p(y_i|x_i = +1)p(x_i = +1)}{p(y_i|x_i = -1)p(x_i = -1)}. \end{aligned} \quad (5.6)$$

We assume that the source is equiprobable; i.e., $p(x_i = +1) = p(x_i = -1) = 1/2$, we have

$$\begin{aligned} \text{LLR}(u_i) &= \ln \frac{p(y_i|x_i = +1)}{p(y_i|x_i = -1)} \\ &= \ln \frac{\frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y-1)^2}{2\sigma^2}}}{\frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y+1)^2}{2\sigma^2}}} \\ &= \frac{2}{\sigma^2} y. \end{aligned} \quad (5.7)$$

At the receiver, a BP algorithm is used where an iterative algorithm for computing marginal of functions is performed [46]. The BP algorithm can easily be described using the TG associated with the decoder. The TG decoder is first initialized with all-zero messages. Each iteration round of the BP decoder starts with all output bits passing messages $m_{t,i}$ to all their neighboring input bits. The message passed from output bit t to source bit i is determined as

$$m_{t,i} = 2 \operatorname{artanh} \left(\tanh \frac{\text{LLR}(u_t)}{2} \cdot \prod_{j=1, j \neq i}^{d_t} \tanh \frac{l_{t,j}}{2} \right), \quad (5.8)$$

where $1 \leq t \leq n$ and d_t is the degree of the output node u_t as shown in Figure 5.1.

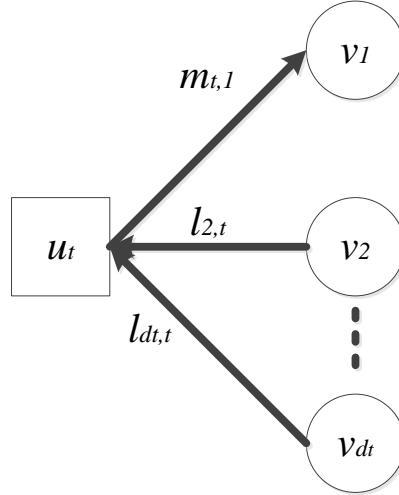


Figure 5.1: Message passing from output bit u_t to its neighbor source bit v_1 .

Also, $l_{t,i}$, where $1 \leq i \leq k$ are the messages sent from source bits to all their neighboring output bits according to:

$$l_{t,i} = \sum_{j=1, j \neq i}^{d_t} m_{t,i}, \quad (5.9)$$

where t is between 1 and n and d_t is the degree of the source bit v_t as shown in Figure 5.2.

These iterations continue until a certain number of iterations has been carried out or a target BER has been reached. Then, the messages to every input bits are summed up to find the estimate a-posteriori probability for each source bits v_t as

$$\sum_{j=1}^{d_t} m_{t,j}. \quad (5.10)$$

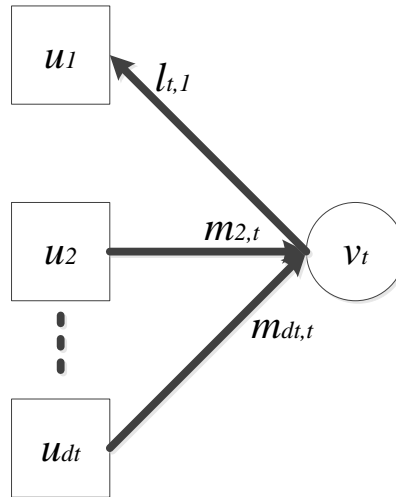


Figure 5.2: Message passing from source bit v_t to its neighbor output bit u_1 .

5.3 OLT Codes

In this section, OLT codes over AWGN channel are presented. In OLT codes, the source data K is divided into a number of generations (g) each of size k and we allow

any two adjacent generations to overlap by α . The overlap, α can be determined as:

$$\alpha = \left\lceil \frac{gk - K}{g - 1} \right\rceil \quad (5.11)$$

where $\frac{K}{k} \leq g \leq K - k + 1$. Clearly, if $g = \frac{K}{k}$, this is the conventional LT code where there is no overlap, $\alpha = 0$. Also, when $g = K - k + 1$, this is the case where any two adjacent generations overlap in all k bits but 1.

5.3.1 Encoding OLT Codes

In the encoding process, each generation is subdivided into a number of partitions, each partition has a specific priority in the encoding process. Similar to the case of the BEC in Chapter 4, we use only two partitions in this work. The first partition with size $(k - \alpha)$ and the second partition with size α . The first partition is chosen with probability p in the encoding process. Consequently, the second partition is chosen with probability $(1 - p)$ as described in Algorithm 6. Our goal is to allow the receiver recovering the first partition faster than the second partition. To do so, we set $p > 1 - p$; that is $p > 0.5$.

Figure 5.3 shows an example of the encoding process of the proposed scheme. The first generation contains the source bits $\{v_1, v_2, \dots, v_8\}$ (yellow and red) and the encoded bits for this generation are in yellow $\{u_1, u_2, \dots, u_5\}$. The second generation contains the source bits $\{v_5, v_6, \dots, v_{12}\}$ (red and green) and the encoded bits for this generation are in green $\{u_6, u_7, \dots, u_{11}\}$. We can see that the first and second generations are overlapped by α bits (in red), namely, $\{v_5, v_6, \dots, v_8\}$.

Algorithm 6 Encoding OLT Codes over AWGN channels

```

1: Input: Source bits  $\{v_1, v_2, v_3, \dots, v_K\}$ , selection probability  $p$ , and overlap size  $\alpha$ 
2: Divide the source data into  $g$  generations;  $g = (K - \alpha)/(k - \alpha)$ 
3: for  $l = 1$  to  $g$  do
4:   Divide the source bits in the  $l$  generation into two partitions,  $\pi_1 = \{v_1, v_2, v_3, \dots, v_{k-\alpha}\}$  and  $\pi_2 = \{v_{k-\alpha+1}, v_{k-\alpha+2}, v_{k-\alpha+3}, \dots, v_k\}$ 
5:    $i=1$ 
6:   while Not all receivers communicate their ACKs do
7:     Choose a degree  $d$  from the RSD distribution
8:     for  $s = 1$  to  $d$  do
9:       Randomly choose a parameter  $x$ , where  $0 \leq x \leq 1$ 
10:      if  $x < p$  then
11:        Select the first partition  $\pi_1$ 
12:      else
13:        Select the second partition  $\pi_2$ 
14:      end if
15:      Choose a source bit, uniformly at random, from the selected partition
16:    end for
17:    Perform XOR on the  $d$  chosen source bits in the previous steps  $u_i = \bigoplus_{j=1}^d V_j$ ,
    where  $V$  is a subset of the source bits  $k$  containing only the chosen  $d$  source bits
18:    Modulate the encoded bit;  $x_i = (-1)^{u_i}$ 
19:    Broadcast the modulated bit  $x_i$ 
20:     $i++$ 
21:  end while
22: end for

```

5.3.2 Decoding OLT Codes

At the receiver, we use the same BP algorithm as discussed in Section 5.2.2. To kick-start the decoding process, the first packet must have degree-one encoded bits. That is because the \tanh function is used to calculate the messages sent from encoded bits to source bits in which the other edges must be all non-zero to have a non-zero message.

Algorithm 7 describes the decoding process of OLT codes over AWGN channels.

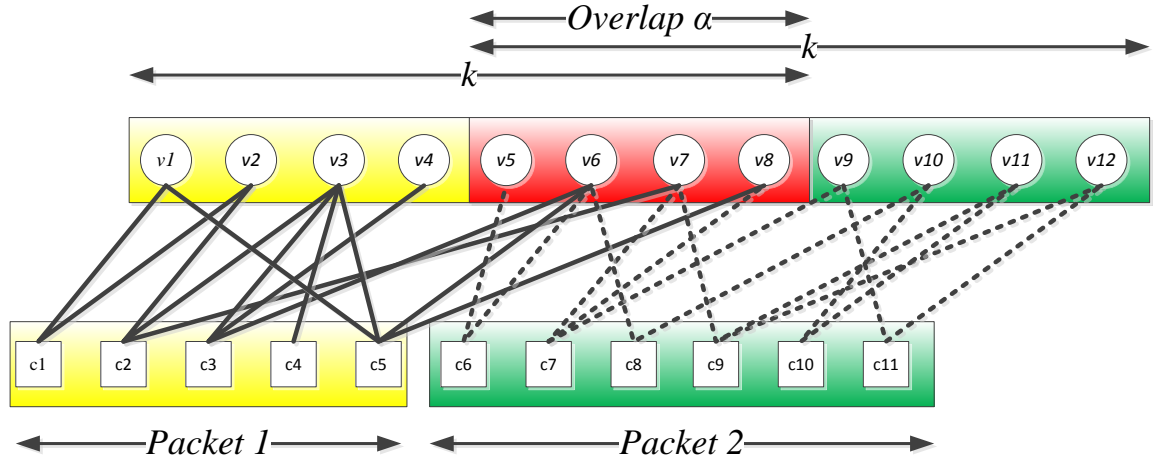


Figure 5.3: Encoding OLT codes.

Because of the higher priority of the first source partition, the receiver recovers the first partition faster than the second one. Once the stopping criteria of the first partition is fulfilled, the receiver sends an ACK to the encoder. When all the receivers send their ACKs to the source successfully, the encoder moves to the next generation and so on.

Algorithm 7 Decoding OLT Codes over AWGN channels

- 1: **Input:** Received bits $\{y_1, y_2, y_3, \dots\}$, overlap size α , and AWGN variance σ^2
 - 2: Calculate the LLR for each received bit using Equation (5.7)
 - 3: **while** The stopping criteria of the first $(k - \alpha)$ source bits is not met **do**
 - 4: Send the beliefs of each encoded bit (Equation 5.8) to its neighbor source bits
 - 5: Send the beliefs of each source bit (Equation 5.9) to its neighbor encoded bits
 - 6: Update the stopping criteria
 - 7: **end while**
 - 8: Send an ACK to the transmitter
-

In the example of Figure 5.3, u_4 is a degree-one encoded bit. The decoder starts the process by sending the incoming LLR from the channel on u_4 to v_3 . Once the

decoder decodes the source bits in yellow, it moves forward to the next packet. One of the benefits of using OLT codes is shown here (in generations 2,3,4,...) where as the decoder does not need degree-one encoded bits to kick-start the decoding process. The reason behind this is because, at the beginning of the process, all the variable nodes send down their beliefs of what they are to the neighbors check nodes. For instance, v_6, v_7 , and v_8 have non-zero beliefs from the previous decoding process. That means u_6 has the LLR coming from the channel and a message from v_6 , so it can send a message to v_5 . The same result holds for u_7 and u_8 .

5.4 Smart Overlapped LT (SOLT) Codes

The benefit of using the proposed OLT codes in noisy channels is twofold. First, the overall performance is improved, and second, it allows for the design of a new degree distribution that can get rid of degree-one encoded bits. In fact, it has been shown that degree-one encoded bits are not efficient as they reduce the code performance [25]. The only benefit of using degree-one encoded bits is to kick-start the decoding process. Therefore, we propose a new degree distribution by eliminating the degree-one encoded bits entirely starting from the second generation, and we use systematic connections only in the first generation. The proposed scheme is referred to as smart overlapped LT (SOLT) codes.

Encoding process of SOLT codes is described in Algorithm 8. Note that the modified RSD is the traditional RSD (Equation 2.20) but with zero probability of degree one; that is $\Omega_1 = 0$, and its probability is distributed over other degrees such that $\sum_{d=2}^k \Omega_d = 1$. The decoding process of SOLT codes is similar to that of OLT codes which described in Algorithm 7.

Algorithm 8 Encoding SOLT Codes over AWGN channels

```

1: Input: Generation number  $l$ , source bits  $\{v_1, v_2, v_3, \dots, v_K\}$ , selection probability
    $p$ , and overlap size  $\alpha$ 
2: Divide the source data into  $g$  generation;  $g = (K - \alpha)/(k - \alpha)$ 
3: for  $l = 1$  to  $g$  do
4:   Divide the source bits in the  $l$  generation into two partitions,  $\pi_1 =$ 
    $\{v_1, v_2, v_3, \dots, v_{k-\alpha}\}$  and  $\pi_2 = \{v_{k-\alpha+1}, v_{k-\alpha+2}, v_{k-\alpha+3}, \dots, v_k\}$ 
5:    $i=1$ 
6:   while Not all receivers communicate their ACKs do
7:     if  $l = 1$  then
8:       Choose a degree  $d$  from the RSD distribution
9:     else
10:      Choose a degree  $d$  from the modified RSD distribution
11:    end if
12:    for  $s = 1$  to  $d$  do
13:      Randomly choose a parameter  $x$ , where  $0 \leq x \leq 1$ 
14:      if  $x < p$  then
15:        Select the first partition  $\pi_1$ 
16:      else
17:        Select the second partition  $\pi_2$ 
18:      end if
19:      Choose a source bits, uniformly at random, from the selected partition
20:    end for
21:    Perform XOR on the  $d$  chosen source bits in the previous steps  $u_i = \bigoplus_{j=1}^d V_j$ ,
   where  $V$  is a subset of the source bits  $k$  containing only the chosen  $d$  source bits
22:    Modulate the encoded bit;  $x_i = (-1)^{u_i}$ 
23:    Broadcast the modulated bit  $x_i$ 
24:     $i++$ 
25:  end while
26: end for

```

5.5 Numerical Results

In this section, extensive simulation results are presented to demonstrate the performance of the proposed OLT and SOLT codes over AWGN channel. These simulation results show that the best selection probability of the first partition π_1 is $p = 0.75$.

We utilize RSD with the parameters $c = 0.02$ and $\delta = 0.1$. The BER is obtained by averaging over 1000 transmission blocks of length $K = 1024$. Each block is divided into generations each with size $k = 256$, and the overlap α is set to be 128, thus the number of generations $g = 7$. The maximum number of iterations at the decoder is set to be 20.

Figure 5.4 represents the performance of LT, OLT, and SOLT codes using RSD versus inverse code rate. We can observe that OLT codes improve the performance of the conventional LT codes in terms of BER. Also, SOLT codes have further improved the performance. It can be seen in the figure that SOLT codes outperform LT and OLT codes in all region of the overhead (or inverse code rate) at the same SNR. For example, if we target $\text{BER}=10^{-4}$ at $\text{SNR}=9$ dB, SOLT codes can achieve the target at a code rate of $R=\frac{1}{2.1}=0.476$ while the conventional LT codes achieve the target at $R=\frac{1}{2.4}=0.417$. This means that SOLT codes require less overhead to achieve the same BER.

As depicted in the figure, conventional LT codes with RSD perform very poorly in AWGN channel. That is because of the need of degree-one encoded bits. RSD distribution has very low probability of degree-one (less than 1%).

To show that the proposed OLT and SOLT codes are still doing better than the conventional LT codes, we use Shokrollahi's degree distribution which is one of the most well-known degree distributions in the literature [24].

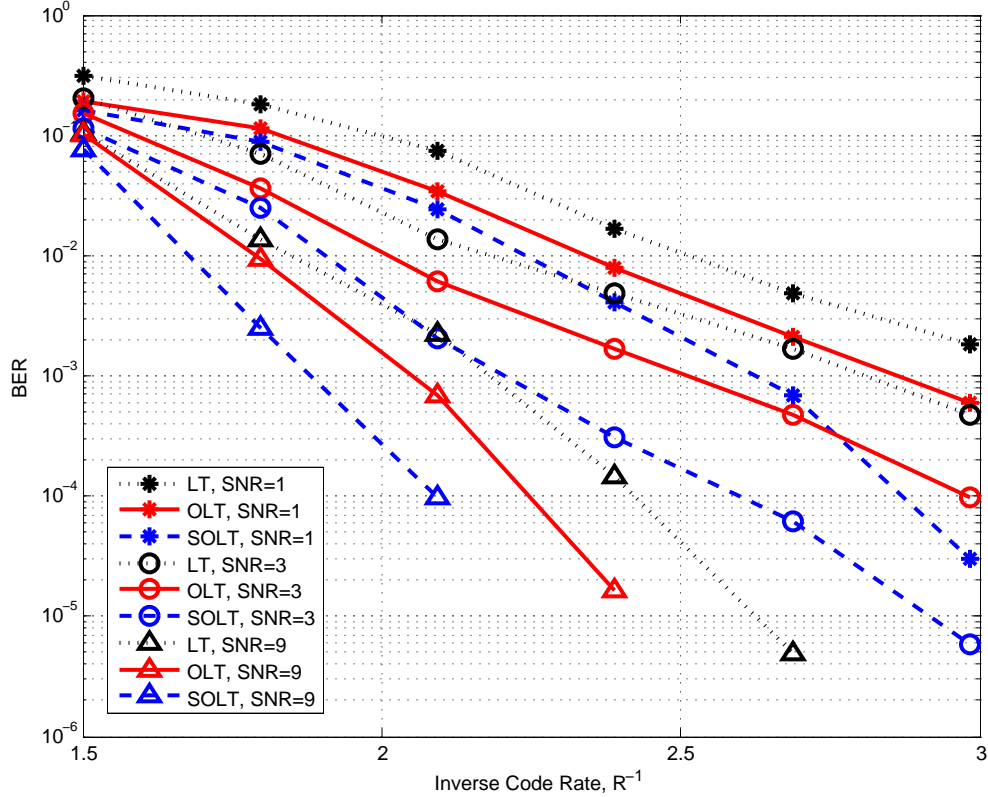


Figure 5.4: BER versus inverse code rate for LT, OLT, and SOLT codes at different SNR's.

Shokrollahi's degree distribution at $k = 256$ is given by

$$\Omega(x) = 0.18x + 0.33x^2 + 0.26x^4 + 0.14x^8 + 0.05x^{16} + 0.01x^{32} + 0.03x^{64}. \quad (5.12)$$

Simulation results of BER for LT, OLT, and SOLT codes using Shokrollahi's degree distribution are shown in Figure 5.5. Similar to Figure 5.4, SOLT codes outperform LT and OLT codes but using Shokrollahi's degree distribution instead of RSD. Again, the improvement of using SOLT codes increases as the rate decrease

(more overhead). In this figure, the performance of LT codes using Shokrollahi's degree distribution is better than using RSD distribution (Figure 5.4). One of the reasons behind this is increasing the probability of degree-one encoded bits (from 1% to 18%).

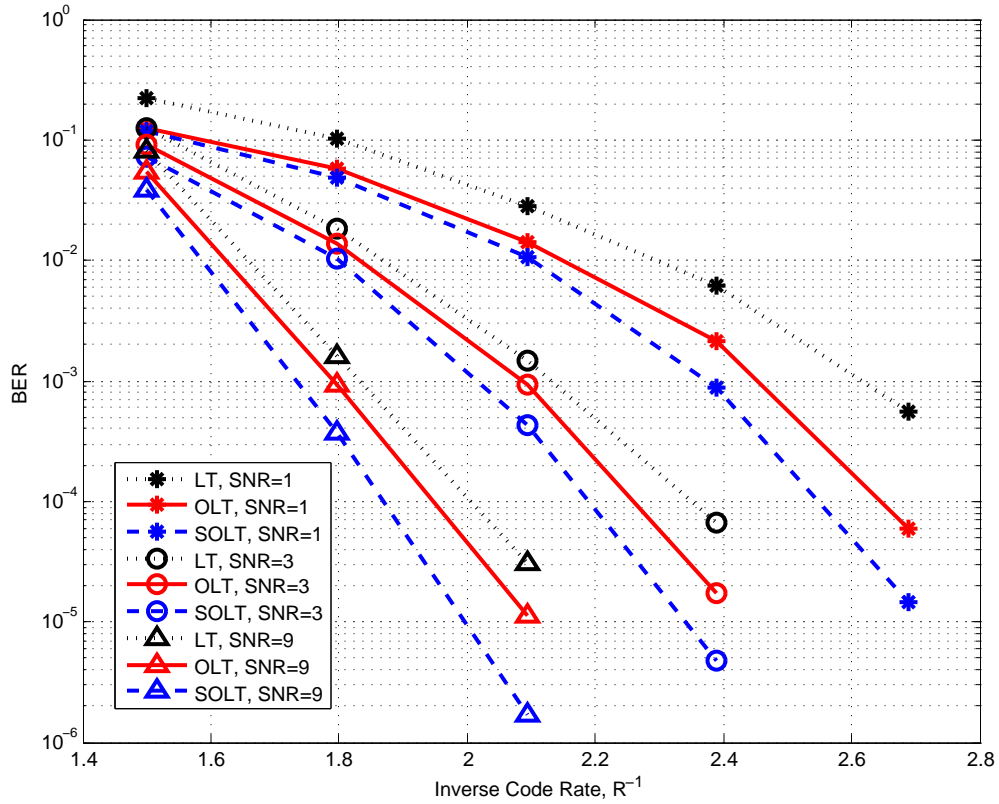


Figure 5.5: BER versus inverse code rate for LT, OLT, and SOLT codes at different SNR's. These codes are generated using Shokrollahi's degree distribution.

5.6 Summary

In this chapter, we applied the idea of overlapped generation of LT codes over AWGN channel. The approach is equivalent to that for the BEC. Assessment of result shows

that the proposed OLT codes outperform the conventional LT codes over AWGN channels. As a next step, we aimed to design the best fit distribution for our idea. The new scheme is referred to as SOLT codes. The proposed OLT and SOLT codes can be used to improve the code performance in a variety of ways such as to reduce the error rate or to increase the realized rate (reduction of delay and redundancy). The new schemes are highly adaptive and scalable for broadcast and multicast scenarios. Our simulation results show that the proposed techniques achieve much lower BER at the same code rate and SNR. For example, at an SNR of 9 dB, SOLT codes improve the rate by 12.5% at a target BER of 10^{-4} while using traditional RSD. Using Shokrollahi's degree distribution at the same SNR of 9 dB and a fixed rate of $R = 0.5$, SOLT codes achieve an error performance that is around 10 times better than LT codes.

Chapter 6

Conclusions

With the invention of new technologies for smart cities and the Internet-of-things (IoT), the demand for scalable, reliable, and cost-efficient codes is highly increased. While conventional fountain codes solve the old problems of fixed-rate codes in many applications and scenarios, they are not able to support high error resilience and throughput, particularly, in the finite-length regime where every practical application resides.

In this thesis, we propose new fountain codes referred to as OLT codes to improve the performance of the conventional LT codes. OLT codes perform better than the conventional LT codes in terms of code rate, complexity, and decoding speed. The superiority is present in almost all applications from unicast to broadcast settings. With the proper choice of overlap size, selection probability, generation size, and degree distribution, the new OLT codes can outperform all other schemes in the literature. While utilizing the RSD, this choice involves the careful selection of c and δ . This was another issue addressed in this work in Chapter 3. It is further possible to optimize these variables to benefit from robustness in addition to all the abovementioned advantages.

OLT codes can utilize fountain codes with any degree distribution. There is no shortage of such distributions in the modern coding literature. Any previously crafted degree distribution or fountain code architecture, e.g., Raptor codes, can equally benefit from the overlapping technique. One can contrast OLT codes against traditional fountain codes in an analogy with convolutional codes against block codes.

The overlapping of generations has the same effect of increasing the generation size in terms of performance improvement, yet, allows for curbing the increased complexity. Thus, better error/erasure probability, realized rate, and encoding/decoding complexity trade-offs can be achieved.

6.1 Summary

Much research has been done in the area of Luby-transform (LT) codes using different values of c and δ parameters without much attention to important properties such as robustness. In the absence of universality, robust designs are particularly important in networks with many diverse users and changing channels conditions. In Chapter 3, we use three analysis techniques to optimize the parameters of LT codes based on different objectives. These techniques are density evolution (DE) and extrinsic information transfer (EXIT) chart and code stability. For example, using DE technique to minimize bit erasure probability (BEP) over a binary erasure channel (BEC) with erasure probability $0 \leq \epsilon \leq 0.2$, the parameters of LT code must be $0.1 \leq \delta \leq 0.2$ and $0.025 \leq c \leq 0.06$. Another example, using code stability to maximize the erasure threshold with generation size $128 \leq k \leq 512$, the parameter c must be $0.025 \leq c \leq 0.075$.

In Chapter 4, we proposed the concept of overlapped generations. We applied this concept for LT codes over the BEC. Analytically, we proved three advantages of OLT code over LT codes: 1) OLT codes start the decoding process by eliminating higher number of edges over the Tanner graph (TG) associated with the code at the first decoding step. 2) The total number of required decoding steps for fully decoding process is much less in OLT codes. 3) The total number of edges in the TG of OLT codes is smaller than that in conventional LT codes. As a result, the encoding and decoding complexities are smaller in OLT codes. Furthermore, we numerically showed that the OLT codes have a better code rate as well as lower complexity. At the end of the chapter, we optimized the parameter of the proposed OLT codes with different objectives.

In Chapter 5, we applied the idea of overlapped generations over AWGN channel. Improvements over conventional LT codes are obtained. Furthermore, we designed a new degree distribution in which the probability of degree-one encoded bits can be eliminated on the generations other than the first generation. The designed degree distribution is referred to as smart OLT (SOLT) codes. we showed that both OLT and SOLT codes improve the performance of LT codes in terms of error probability and/or code rate.

6.2 Future Work

Many research areas are still possible in the topic of overlapped fountain codes, particularly, OLT codes. Here, we provide new directions of some of these areas:

Overlapped Raptor Codes:

A natural extension of OLT codes is to apply the idea of overlapped generations to Raptor codes which can be done over different types of channels including BEC, BSC, and AWGN channels. As mentioned earlier in Chapter 2, Raptor codes were introduced to reduce the encoding and decoding complexities (they are linear in the code length) as well as eliminate error floor phenomena in LT codes. Therefore, we expect substantial gains in the code rate and/or error probability. In fact, overlapped Raptor codes can be used in many practical applications such as mobile TV, video-conferencing, etc.

Unequal Error Protection:

In some applications, parts of the source data are more susceptible to errors (induced by the channel) than the other parts. In other applications, parts of the source data carry more valuable information than other parts and they need more protection. The idea of overlapped generations can be used in such cases to provide unequal error protection (UEP). The first partition in the generation is given the highest protection (i.e., highest selection probability at the encoding process) and must be fully decoded at the receiver while the other partitions are given less protection and the receiver should recover as much as it can.

Left Degree Distribution:

To further improve the performance of OLT codes, One can use a selection method other than uniformly at random selection (which is used in the traditional LT codes). Consequently, the left degree distributions will be no longer Poisson distribution.

Thus, we can choose the selection methods depending on our application and the required performance. For instance, one can select portion of source data with high probability which will lead to EUP as mentioned in the previous point.

Bit Error Rate:

We were able to analytically prove the improvements of OLT codes over LT codes using the expected number of removed edges at each decoding step. However, BER or ripple size analyses can be used to further prove these improvements as well as optimize the selection probability.

Bibliography

- [1] K. Schwab, *The Fourth Industrial Revolution*. Penguin UK, 2017. 1
- [2] D. Giusto, A. Iera, G. Morabito, and L. Atzori, *The Internet of Things: 20th Tyrrhenian Workshop on Digital Communications*. Springer Science & Business Media, 2010. 1
- [3] L. Atzori, A. Iera, and G. Morabito, “The Internet of Things: A Survey,” *Computer networks*, vol. 54, no. 15, pp. 2787–2805, 2010. 1
- [4] A. Zanella, N. Bui, A. Castellani, L. Vangelista, and M. Zorzi, “Internet of Things for Smart Cities,” *IEEE Internet of Things Journal*, vol. 1, no. 1, pp. 22–32, Feb 2014. 1
- [5] A. Osseiran, F. Boccardi, V. Braun, K. Kusume, P. Marsch, M. Maternia, O. Queseth, M. Schellmann, H. Schotten, H. Taoka, H. Tullberg, M. A. Uusitalo, B. Timus, and M. Fallgren, “Scenarios for 5G Mobile and Wireless Communications: the Vision of the METIS Project,” *IEEE Commun. Mag.*, vol. 52, no. 5, pp. 26–35, May 2014. 1
- [6] T. S. Rappaport, S. Sun, R. Mayzus, H. Zhao, Y. Azar, K. Wang, G. N. Wong, J. K. Schulz, M. Samimi, and F. Gutierrez, “Millimeter Wave Mobile Communications for 5G Cellular: It Will Work!” *IEEE Access*, vol. 1, pp. 335–349, 2013. 2
- [7] P. Demestichas, A. Georgakopoulos, D. Karvounas, K. Tsagkaris, V. Stavroulaki, J. Lu, C. Xiong, and J. Yao, “5G on the Horizon: Key Challenges for the Radio-Access Network,” *IEEE Vehicular Technology Magazine*, vol. 8, no. 3, pp. 47–53, 2013. 2
- [8] E. Dahlman, G. Mildh, S. Parkvall, J. Peisa, J. Sachs, and Y. Selén, “5G Radio Access,” *Ericsson review*, vol. 6, pp. 2–7, 2014. 2
- [9] E. Dahlman, G. Mildh, S. Parkvall, J. Peisa, J. Sachs, Y. Seln, and J. Skld, “5G Wireless Access: Requirements and Realization,” *IEEE Communications Magazine*, vol. 52, no. 12, pp. 42–47, December 2014. 2

- [10] V. Jungnickel, K. Manolakis, W. Zirwas, B. Panzner, V. Braun, M. Lossow, M. Sternad, R. Apelfrojd, and T. Svensson, “The Role of Small Cells, Coordinated Multipoint, and Massive MIMO in 5G,” *IEEE Communications Magazine*, vol. 52, no. 5, pp. 44–51, 2014. 2
- [11] C. Bockelmann, N. Pratas, H. Nikopour, K. Au, T. Svensson, C. Stefanovic, P. Popovski, and A. Dekorsy, “Massive Machine-Type Communications in 5G: Physical and MAC-layer Solutions,” *IEEE Communications Magazine*, vol. 54, no. 9, pp. 59–65, 2016. 2
- [12] M. N. Tehrani, M. Uysal, and H. Yanikomeroğlu, “Device-to-Device Communication in 5G Cellular Networks: Challenges, Solutions, and Future Directions,” *IEEE Communications Magazine*, vol. 52, no. 5, pp. 86–92, May 2014. 2
- [13] J. Qiao, X. S. Shen, J. W. Mark, Q. Shen, Y. He, and L. Lei, “Enabling Device-to-Device Communications in Millimeter-Wave 5G Cellular Networks,” *IEEE Communications Magazine*, vol. 53, no. 1, pp. 209–215, January 2015. 2
- [14] A. Neubauer, J. Freudenberger, and V. Kuhn, *Coding Theory: Algorithms, Architectures and Applications*. Jhon wiley and Sons Ltd, 2007. 2, 19
- [15] T. K. Moon, *Error Correction Coding: Mathematical Methods and Algorithms*. John Wiley & Sons, Inc., 2005. [Online]. Available: <http://dx.doi.org/10.1002/0471739219.fmatter> 2, 59
- [16] S. Lin and D. J. Costello, *Error Control Coding (2nd Edition)*. Pearson Prentice Hall, New Jersey, U.S.A., 2004. 3, 5, 20, 21, 25
- [17] Specification, Protocol, “Transmission Control Protocol,” 1981. 3
- [18] A. Shokrollahi and M. Luby, “Raptor Codes,” *Foundations and Trends in Communications and Information Theory*, vol. 6, no. 3-4, pp. 213–322, 2011. [Online]. Available: <http://dx.doi.org/10.1561/0100000060> 4
- [19] G. Vitetta, D. P. Taylor, G. Colavolpe, F. Pancaldi, and P. A. Martin, *Wireless Communications: Algorithmic Techniques*. John Wiley & Sons, 2013. 5
- [20] J. W. Byers, M. Luby, M. Mitzenmacher, and A. Rege, “A Digital Fountain Approach to Reliable Distribution of Bulk Data,” in *Proceedings of the ACM SIGCOMM '98 conference on Applications, technologies, architectures, and protocols for computer communication*, ser. SIGCOMM '98. New York, NY, USA: ACM, 1998, pp. 56–67. [Online]. Available: <http://doi.acm.org/10.1145/285237.285258> 5, 7, 29, 80

- [21] J. Byers, M. Luby, and M. Mitzenmacher, "A Digital Fountain Approach to Asynchronous Reliable Multicast," *Selected Areas in Communications, IEEE Journal on*, vol. 20, no. 8, pp. 1528–1540, 2002. 5, 29
- [22] D. J. C. MacKay, "Fountain Codes," *IEE Proceedings - Communications*, vol. 152, no. 6, pp. 1062–1068, Dec 2005. 5, 12, 30, 43
- [23] M. Luby, "LT Codes," in *Proceedings of the 43rd Ann. IEEE Symposium on Foundations of Computer Science*. MISC, 2002, pp. 271–280. 6, 12, 36, 40, 44, 45, 56, 65, 69, 79
- [24] A. Shokrollahi, "Raptor Codes," *Information Theory, IEEE Transactions on*, vol. 52, no. 6, pp. 2551–2567, 2006. 6, 40, 49, 56, 71, 79, 113
- [25] O. Etesami and A. Shokrollahi, "Raptor Codes on Binary Memoryless Symmetric Channels," *Information Theory, IEEE Transactions on*, vol. 52, no. 5, pp. 2033–2051, 2006. 7, 56, 79, 111
- [26] R. Palanki and J. S. Yedidia, "Rateless Codes on Noisy Channels," in *Information Theory, 2004. ISIT 2004. Proceedings. International Symposium on*, 2004, p. 37. 7
- [27] J. Castura and Y. Mao, "Rateless Coding over Fading Channels," *Communications Letters, IEEE*, vol. 10, no. 1, pp. 46–48, 2006. 7
- [28] B. Yang, R. Carrasco, and A. Adams, "Implementation of Fountain Codes over Fading Channels," in *Wireless, Mobile and Multimedia Networks, 2006 IET International Conference on*, 2006, pp. 1–4. 7
- [29] A. Molisch, N. Mehta, J. S. Yedidia, and J. Zhang, "Performance of Fountain Codes in Collaborative Relay Networks," *Wireless Communications, IEEE Transactions on*, vol. 6, no. 11, pp. 4108–4119, 2007. 7
- [30] J. Castura and Y. Mao, "Rateless Coding for Wireless Relay Channels," *IEEE Transactions on Wireless Communications*, vol. 6, no. 5, pp. 1638–1642, May 2007. 7
- [31] X. Liu and T. J. Lim, "Fountain Codes over Fading Relay Channels," *Wireless Communications, IEEE Transactions on*, vol. 8, no. 6, pp. 3278–3287, 2009. 7
- [32] A. Liau, S. Yousefi, and I.-M. Kim, "Binary Soliton-Like Rateless Coding for the Y-Network," *Communications, IEEE Transactions on*, vol. 59, no. 12, pp. 3217–3222, 2011. 7, 79

- [33] A. Apavatjirut, C. Goursaud, K. Jaffres-Runser, C. Comaniciu, and J. M. Gorce, "Toward Increasing Packet Diversity for Relaying LT Fountain Codes in Wireless Sensor Networks," *IEEE Communications Letters*, vol. 15, no. 1, pp. 52–54, January 2011. 7
- [34] A. F. Molisch, *Relaying, MultiHop, and Cooperative Communications*. Wiley-IEEE Press, 2011, pp. 521–563. [Online]. Available: <http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=5635453> 7
- [35] A. Liao, I. M. Kim, and S. Yousefi, "Improved Low-Complexity Soliton-Like Network Coding for a Resource-Limited Relay," *IEEE Transactions on Communications*, vol. 61, no. 8, pp. 3327–3335, August 2013. 7, 79
- [36] G. Liva, E. Paolini, and M. Chiani, "Performance Versus Overhead for Fountain Codes over Fq," *IEEE Communications Letters*, vol. 14, no. 2, pp. 178–180, February 2010. 7
- [37] C. E. Shannon, "A Mathematical Theory of Communication," *The Bell System Technical Journal*, vol. 27, no. 4, pp. 623–656, Oct 1948. 11
- [38] S. Verdú, "Fifty Years of Shannon Theory," *IEEE Transactions on Information Theory*, vol. 44, no. 6, pp. 2057–2078, Oct 1998. 11
- [39] R. V. Hartley, "Transmission of Information," *Bell Labs Technical Journal*, vol. 7, no. 3, pp. 535–563, 1928. 12
- [40] K. F. Hayajneh, J. Spencer, and S. Yousefi, "Robust Quaternary Fountain Codes in AWGN Interference," *IET Communications*, submitted 2017. 15
- [41] P. Elias, "Coding for Two Noisy Channels," in *Proc. 3rd London Symp. Information Theory, London, U.K.*, 1955, pp. 61–76. 16, 21
- [42] T. M. Cover and J. A. Thomas, *Elements of Information Theory*. John Wiley & Sons, 2012. 16, 17
- [43] A. Shokrollahi, "LDPC codes: An Introduction," Digital Fountain, Inc., Tech. Rep. 2, 2003. 17, 25
- [44] R. W. Hamming, "Error Detecting and Error Correcting Codes," *The Bell System Technical Journal*, vol. 29, no. 2, pp. 147–160, April 1950. 19, 20
- [45] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon Limit Error-Correcting Coding and Decoding: Turbo-Codes. 1," in *Communications, 1993. ICC '93 Geneva. Technical Program, Conference Record, IEEE International Conference on*, vol. 2, May 1993, pp. 1064–1070. 19, 21, 22

- [46] R. Gallager, “Low-Density Parity-Check Codes,” *IRE Transactions on information theory*, vol. 8, no. 1, pp. 21–28, 1962. [19](#), [20](#), [22](#), [106](#)
- [47] R. C. Bose and D. K. Ray-Chaudhuri, “On a Class of Error Correcting Binary Group Codes,” *Information and control*, vol. 3, no. 1, pp. 68–79, 1960. [20](#)
- [48] A. Hocquenghem, “Codes Correcteurs D’erreurs,” *Chiffres*, vol. 2, no. 147-156, pp. 8–5, 1959. [20](#)
- [49] I. S. Reed and G. Solomon, “Polynomial Codes over Certain Finite Fields,” *Journal of the Society for Industrial and Applied Mathematics*, vol. 8, no. 2, pp. 300–304, 1960. [Online]. Available: <http://dx.doi.org/10.1137/0108018> [20](#)
- [50] R. Tanner, “A Recursive Approach to Low Complexity Codes,” *IEEE Transactions on Information Theory*, vol. 27, no. 5, pp. 533–547, Sep 1981. [20](#)
- [51] G. Forney, “Convolutional Codes I: Algebraic Structure,” *IEEE Transactions on Information Theory*, vol. 16, no. 6, pp. 720–738, Nov 1970. [21](#)
- [52] R. Johannesson and K. S. Zigangirov, *Fundamentals of Convolutional Coding*. John Wiley & Sons, 2015, vol. 15. [21](#)
- [53] G. Ungerboeck, “Channel Coding with Multilevel/Phase Signals,” *IEEE Transactions on Information Theory*, vol. 28, no. 1, pp. 55–67, Jan 1982. [21](#)
- [54] —, “Trellis-Coded Modulation with Redundant Signal Sets Part I: Introduction,” *IEEE Communications Magazine*, vol. 25, no. 2, pp. 5–11, February 1987. [21](#)
- [55] —, “Trellis-Coded Modulation with Redundant Signal Sets Part II: State of the Art,” *IEEE Communications Magazine*, vol. 25, no. 2, pp. 12–21, February 1987. [21](#)
- [56] R. Gallager, *Low-Density Parity-Check Codes*. MIT Press, Cambridge, MA, USA, 1963. [22](#)
- [57] M. Sipser and D. A. Spielman, “Expander Codes,” in *Proceedings 35th Annual Symposium on Foundations of Computer Science*, Nov 1994, pp. 566–576. [23](#)
- [58] —, “Expander Codes,” *IEEE Transactions on Information Theory*, vol. 42, no. 6, pp. 1710–1722, 1996. [23](#)
- [59] N. Wiberg, H.-A. Loeliger, and R. Kotter, “Codes and Iterative Decoding on General Graphs,” in *Proceedings of 1995 IEEE International Symposium on Information Theory*, Sep 1995, p. 468. [23](#)

- [60] —, “Codes and Iterative Decoding on General Graphs,” *European Transactions on Telecommunications*, vol. 6, no. 5, pp. 513–525, 1995. [Online]. Available: <http://dx.doi.org/10.1002/ett.4460060507> 23
- [61] D. J. C. MacKay and R. M. Neal, “Near Shannon Limit Performance of Low Density Parity Check Codes,” *Electronics letters*, vol. 32, no. 18, pp. 1645–1646, 1996. 23
- [62] —, “Near Shannon Limit Performance of Low Density Parity Check Codes,” *Electronics Letters*, vol. 33, no. 6, pp. 457–458, Mar 1997. 23
- [63] D. J. C. MacKay, “Good Error-Correcting Codes Based on Very Sparse Matrices,” *IEEE Transactions on Information Theory*, vol. 45, no. 2, pp. 399–431, Mar 1999. 23, 26
- [64] D. J. MacKay, *Information Theory, Inference and Learning Algorithms*. Cambridge university press, 2003. 25, 30
- [65] M. G. Luby, M. Mitzenmacher, M. A. Shokrollahi, D. A. Spielman, and V. Stemann, “Practical Loss-Resilient Codes,” in *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*. ACM, 1997, pp. 150–159. 25, 27, 29, 56
- [66] M. Luby, M. Mitzenmacher, M. Shokrollahi, and D. Spielman, “Efficient Erasure Correcting Codes,” *Information Theory, IEEE Transactions on*, vol. 47, no. 2, pp. 569–584, 2001. 25, 29, 56
- [67] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, 1988. 25
- [68] J. fu Cheng and R. J. McEliece, “Some High-Rate Near Capacity Codes for the Gaussian Channel,” in *34th Allerton Conference on Communications, Control and Computing*, 1996. 26
- [69] J. Garcia-Frias and W. Zhong, “Approaching Shannon Performance by Iterative Decoding of Linear Codes with Low-Density Generator Matrix,” *IEEE Communications Letters*, vol. 7, no. 6, pp. 266–268, June 2003. 26
- [70] T. R. Oenning and J. Moon, “A Low-Density Generator Matrix Interpretation of Parallel Concatenated Single Bit Parity Codes,” *IEEE Transactions on Magnetism*, vol. 37, no. 2, pp. 737–741, Mar 2001. 26
- [71] T. J. Richardson, M. A. Shokrollahi, and R. L. Urbanke, “Design of Capacity-Approaching Irregular Low-Density Parity-Check Codes,” *IEEE Transactions on Information Theory*, vol. 47, no. 2, pp. 619–637, Feb 2001. 27, 55, 56, 102

- [72] M. G. Luby, M. Mitzenmacher, M. A. Shokrollahi, and D. A. Spielman, "Improved Low-Density Parity-Check Codes Using Irregular Graphs," *IEEE Transactions on Information Theory*, vol. 47, no. 2, pp. 585–598, 2001. 27
- [73] M. P. C. Fossorier, M. Mihaljevic, and H. Imai, "Reduced Complexity Iterative Decoding of Low-Density Parity Check Codes Based on Belief Propagation," *IEEE Transactions on Communications*, vol. 47, no. 5, pp. 673–680, May 1999. 27
- [74] J. Feldman, M. J. Wainwright, and D. R. Karger, "Using Linear Programming to Decode Binary Linear Codes," *IEEE Transactions on Information Theory*, vol. 51, no. 3, pp. 954–972, March 2005. 27
- [75] D. Mandelbaum, "An Adaptive-Feedback Coding Scheme Using Incremental Redundancy (Corresp.)," *IEEE Transactions on Information Theory*, vol. 20, no. 3, pp. 388–389, May 1974. 28
- [76] J. Ha, J. Kim, and S. W. McLaughlin, "Rate-Compatible Puncturing of Low-Density Parity-Check Codes," *IEEE Transactions on Information Theory*, vol. 50, no. 11, pp. 2824–2836, Nov 2004. 28
- [77] J. Ha, J. Kim, D. Klinc, and S. W. McLaughlin, "Rate-Compatible Punctured Low-Density Parity-Check Codes with Short Block Lengths," *IEEE Transactions on Information Theory*, vol. 52, no. 2, pp. 728–738, Feb 2006. 28
- [78] J. Hagenauer, N. Seshadri, and C. E. W. Sundberg, "The Performance of Rate-Compatible Punctured Convolutional Codes for Future Digital Mobile Radio," in *38th IEEE Vehicular Technology Conference*, Jun 1988, pp. 22–29. 28
- [79] J. Hagenauer, "Rate-Compatible Punctured Convolutional Codes (RCPC Codes) and Their Applications," *IEEE Transactions on Communications*, vol. 36, no. 4, pp. 389–400, Apr 1988. 28
- [80] A. S. Barbulescu and S. S. Pietrobon, "Rate Compatible Turbo Codes," *Electronics Letters*, vol. 31, no. 7, pp. 535–536, Mar 1995. 28
- [81] P. Jung and J. Plechinger, "Performance of Rate Compatible Punctured Turbo-Codes for Mobile Radio Applications," *Electronics Letters*, vol. 33, no. 25, pp. 2102–2103, Dec 1997. 28
- [82] J. Bloemer, M. Kalfane, R. Karp, M. Karpinski, M. Luby, and D. Zuckerman, "An XOR-Based Erasure-Resilient Coding Scheme," 1995. 30

- [83] J. Nonnenmacher, E. Biersack, and D. Towsley, “Parity-Based Loss Recovery for Reliable Multicast Transmission,” in *ACM SIGCOMM Computer Communication Review*, vol. 27, no. 4. ACM, 1997, pp. 289–300. 30
- [84] L. Rizzo, “Effective Erasure Codes for Reliable Computer Communication Protocols,” *ACM SIGCOMM computer communication review*, vol. 27, no. 2, pp. 24–36, 1997. 30
- [85] C. Papadopoulos, G. Parulkar, and G. Varghese, “An Error Control Scheme for Large-Scale Multicast Applications,” in *Proceedings of the seventeenth annual ACM symposium on Principles of distributed computing*. ACM, 1998, p. 310. 30
- [86] A. Shokrollahi, “Raptor Codes,” in *International Symposium on Information Theory, 2004. ISIT 2004. Proceedings.*, June 2004, p. 36. 49
- [87] A. Ashikhmin, G. Kramer, and S. ten Brink, “Extrinsic Information Transfer Functions: Model and Erasure Channel Properties,” *IEEE Transactions on Information Theory*, vol. 50, no. 11, pp. 2657–2673, 2004. 55, 59, 60, 102
- [88] T. J. Richardson and R. L. Urbanke, “The Capacity of Low-Density Parity-Check Codes Under Message-Passing Decoding,” *IEEE Transactions on Information Theory*, vol. 47, no. 2, pp. 599–618, Feb 2001. 56, 60
- [89] D. Divsalar, S. Dolinar, and F. Pollara, “Iterative Turbo Decoder Analysis Based on Density Evolution,” *IEEE Journal on Selected Areas in Communications*, vol. 19, no. 5, pp. 891–907, May 2001. 56
- [90] M. Luby, M. Mitzenmacher, and M. A. Shokrollahi, “Analysis of Random Processes via And-Or Tree Evaluation,” in *SODA*, vol. 98, January 1998, pp. 364–373. 56, 59, 60
- [91] M. Luby, M. Mitzenmacher, A. Shokrollah, and D. Spielman, “Analysis of Low Density Codes and Improved Designs Using Irregular Graphs,” in *Proceedings of the thirtieth annual ACM symposium on Theory of computing*. ACM, 1998, pp. 249–258. 56
- [92] D. Sejdinovic, R. J. Piechocki, and A. Doufexi, “AND-OR Tree Analysis of Distributed LT Codes,” in *Networking and Information Theory, 2009. ITW 2009. IEEE Information Theory Workshop on*. IEEE, pp. 261–265. 56
- [93] P. Maymounkov, “Online Codes,” New York University, Tech. Rep., 2002. 56, 57

- [94] S. ten Brink, “Convergence of Iterative Decoding,” *Electronics Letters*, vol. 35, no. 10, pp. 806–808, May 1999. 59
- [95] —, “Designing Iterative Decoding Schemes with the Extrinsic Information Transfer Chart,” in *AEU Int. J. Electron. Commun.* Citeseer, 2000. 59
- [96] —, “Convergence Behavior of Iteratively Decoded Parallel Concatenated Codes,” *IEEE Transactions on Communications*, vol. 49, no. 10, pp. 1727–1737, Oct 2001. 59
- [97] S.-Y. Chung, T. J. Richardson, and R. L. Urbanke, “Analysis of Sum-Product Decoding of Low-Density Parity-Check Codes Using a Gaussian Approximation,” *IEEE Transactions on Information Theory*, vol. 47, no. 2, pp. 657–670, Feb 2001. 60
- [98] M. Ardakani and F. R. Kschischang, “A More Accurate One-Dimensional Analysis and Design of Irregular LDPC Codes,” *IEEE Transactions on Communications*, vol. 52, no. 12, pp. 2106–2114, Dec 2004. 60
- [99] Z. Cheng, J. Castura, and Y. Mao, “On the Design of Raptor Codes for Binary-Input Gaussian Channels,” *IEEE Transactions on Communications*, vol. 57, no. 11, pp. 3269–3277, Nov 2009. 60
- [100] T. Richardson and R. Urbanke, *Modern Coding Theory*. Cambridge University Press, 2008. 62
- [101] E. Hyytia, T. Tirronen, and J. Virtamo, “Optimizing the Degree Distribution of LT Codes with an Importance Sampling Approach,” in *Proceedings of RESIM. MISC*, 2006. 79
- [102] H. Zhu, G. Zhang, and G. Li, “A Novel Degree Distribution Algorithm of LT Codes,” in *Communication Technology, 2008. ICCT 2008. 11th IEEE International Conference on*, Nov 2008, pp. 221–224. 79
- [103] M. Zhang, W. Lei, and X. Xie, “Combined Degree Distribution: A Simple Method to Design the Degree Distribution of Fountain Codes,” in *Information Science and Technology (ICIST), 2013 International Conference on*, March 2013, pp. 1089–1092. 79
- [104] S. Puducheri, J. Kliever, and T. Fuja, “The Design and Performance of Distributed LT Codes,” *Information Theory, IEEE Transactions on*, vol. 53, no. 10, pp. 3740–3754, 2007. 79

- [105] C.-M. Chen, Y. ping Chen, T.-C. Shen, and J. Zao, “On the Optimization of Degree Distributions in LT Code with Covariance Matrix Adaptation Evolution Strategy,” in *Evolutionary Computation (CEC), 2010 IEEE Congress on*, July 2010, pp. 1–8. 79
- [106] J. Sorensen, P. Popovski, and J. Ostergaard, “Design and Analysis of LT Codes with Decreasing Ripple Size,” *Communications, IEEE Transactions on*, vol. 60, no. 11, pp. 3191–3197, 2012. 79, 80, 88
- [107] K. F. Hayajneh and S. Yousefi, “Improved Systematic Fountain Codes in AWGN Channel,” in *2013 13th Canadian Workshop on Information Theory*, June 2013, pp. 148–152. 80, 102
- [108] K. F. Hayajneh, S. Yousefi, and M. Valipour, “Left Degree Distribution Shaping for LT Codes over the Binary Erasure Channel,” in *2014 27th Biennial Symposium on Communications (QBSC)*, June 2014, pp. 198–202. 80
- [109] J. W. Byers, M. Luby, and M. Mitzenmacher, “Accessing Multiple Mirror Sites in Parallel: Using Tornado Codes to Speed up Downloads,” in *INFOCOM '99. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, vol. 1, Mar 1999, pp. 275–283 vol.1. 80
- [110] A. Tan, A. Aksay, C. Bilen, G. Akar, and E. Arikan, “Rate-Distortion Optimized Layered Stereoscopic Video Streaming with Raptor Codes,” in *Packet Video 2007*, Nov 2007, pp. 98–104. 80
- [111] J.-P. Wagner, J. Chakareski, and P. Frossard, “Streaming of Scalable Video from Multiple Servers Using Rateless Codes,” in *Multimedia and Expo, 2006 IEEE International Conference on*, July 2006, pp. 1501–1504. 80
- [112] S. Ahmad, R. Hamzaoui, and M. Al-Akaidi, “Robust Live Unicast Video Streaming with Rateless Codes,” in *Packet Video 2007*, Nov 2007, pp. 78–84. 80
- [113] —, “Adaptive Unicast Video Streaming With Rateless Codes and Feedback,” *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 20, no. 2, pp. 275–285, Feb 2010. 80
- [114] H. Jenkac and T. Stockhammer, “Asynchronous Media Streaming over Wireless Broadcast Channels,” in *Multimedia and Expo, 2005. ICME 2005. IEEE International Conference on*, July 2005, pp. 1318–1321. 81

- [115] D. Vukobratovic, V. Stankovic, D. Sejdinovic, L. Stankovic, and Z. Xiong, “Scalable Video Multicast Using Expanding Window Fountain Codes,” *Multimedia, IEEE Transactions on*, vol. 11, no. 6, pp. 1094–1104, Oct 2009. 81
- [116] —, “Expanding Window Fountain Codes for Scalable Video Multicast,” in *Multimedia and Expo, 2008 IEEE International Conference on*, June 2008, pp. 77–80. 81
- [117] P. Cataldi, M. Grangetto, T. Tillo, E. Magli, and G. Olmo, “Sliding-Window Raptor Codes for Efficient Scalable Wireless Video Broadcasting With Unequal Loss Protection,” *Image Processing, IEEE Transactions on*, vol. 19, no. 6, pp. 1491–1503, June 2010. 81
- [118] A. Bouabdallah and J. Lacan, “Dependency-Aware Unequal Erasure Protection Codes,” *Journal of Zhejiang University - Science A*, vol. 7, no. 1, pp. 27–33, January 2006. [Online]. Available: <http://oatao.univ-toulouse.fr/4002/> 81
- [119] C. Hellge, T. Schierl, and T. Wiegand, “Receiver Driven Layered Multicast with Layer-Aware Forward Error Correction,” in *Image Processing, 2008. ICIP 2008. 15th IEEE International Conference on*, Oct 2008, pp. 2304–2307. 81
- [120] —, “Multidimensional Layered Forward Error Correction Using Rateless Codes,” in *Communications, 2008. ICC '08. IEEE International Conference on*, May 2008, pp. 480–484. 81
- [121] A. Albanese, J. Blomer, J. Edmonds, M. Luby, and M. Sudan, “Priority Encoding Transmission,” *IEEE Transactions on Information Theory*, vol. 42, no. 6, pp. 1737–1744, Nov 1996. 83
- [122] U. Wachsmann, R. F. H. Fischer, and J. B. Huber, “Multilevel Codes: Theoretical Concepts and Practical Design Rules,” *IEEE Transactions on Information Theory*, vol. 45, no. 5, pp. 1361–1391, Jul 1999. 83
- [123] J. Hou, P. H. Siegel, L. B. Milstein, and D. Pfister, “Multilevel Coding with Low-Density Parity-Check Component Codes,” in *Global Telecommunications Conference, 2001. GLOBECOM '01. IEEE*, vol. 2, 2001, pp. 1016–1020 vol.2. 83
- [124] K. K. Yen, Y. C. Liao, C. L. Chen, and H. C. Chang, “Modified Robust Soliton Distribution (MRSD) with Improved Ripple Size for LT Codes,” *IEEE Communications Letters*, vol. 17, no. 5, pp. 976–979, May 2013. 88

-
- [125] J. H. Sorensen, T. Koike-Akino, P. Orlik, J. Ostergaard, and P. Popovski, "Ripple Design of LT Codes for BIAWGN Channels," *IEEE Transactions on Communications*, vol. 62, no. 2, pp. 434–441, February 2014. 88
- [126] S.-Y. Chung, "On the Construction of Some Capacity-Approaching Coding Schemes," Ph.D. dissertation, Massachusetts Institute of Technology, 2000. 102
- [127] T. Nguyen, L. L. Yang, and L. Hanzo, "Systematic Luby Transform Codes and Their Soft Decoding," in *Signal Processing Systems, 2007 IEEE Workshop on*, 2007, pp. 67–72. 102
- [128] X. Yuan and L. Ping, "On Systematic LT Codes," *Communications Letters, IEEE*, vol. 12, no. 9, pp. 681–683, 2008. 102
- [129] T. Grobler, E. R. Ackermann, J. Olivier, and A. J. Van Zyl, "Systematic Luby Transform Codes as Incremental Redundancy Scheme," in *AFRICON, 2011*, 2011, pp. 1–5. 102
- [130] V. H. Tran, K. T. Lay, and L. C. Peng, "Modified LT Coding with Systematic Connections," in *Anti-counterfeiting, Security, and Identification*, Aug 2012, pp. 1–4. 102