AN INVESTIGATION ON BELIEF PROPAGATION DECODING OF POLAR CODES

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

ORKUN DOĞAN

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
ELECTRICAL AND ELECTRONICS ENGINEERING

DECEMBER 2015

Approval of the thesis:

AN INVESTIGATION ON BELIEF PROPAGATION DECODING OF POLAR CODES

submitted by **ORKUN DOĞAN** in partial fulfillment of the requirements for the degree of **Master of Science in Electrical and Electronics Engineering Department, Middle East Technical University** by,

Prof. Dr. Gülbin Dural Ünver
Dean, Graduate School of **Natural and Applied Sciences** ________________

Prof. Dr. Gönül Turhan Sayan
Head of Department, **Electrical and Electronics Engineering** ________________

Assoc. Prof. Dr. Melek Diker Yücel
Supervisor, **Electrical and Electronics Engineering Dep.** ________________

Examining Committee Members:

Prof. Dr. Yalçın Tanık
Electrical and Electronics Engineering Department, METU ________________

Assoc. Prof. Dr. Melek Diker Yücel
Electrical and Electronics Engineering Department, METU ________________

Prof. Dr. Erdal Arıkan
Electrical and Electronics Engineering Department, BİLKENT ________________

Prof. Dr. Ali Özgür Yılmaz
Electrical and Electronics Engineering Department, METU ________________

Assoc. Prof. Dr. Çağatay Candan
Electrical and Electronics Engineering Department, METU ________________

Date: ________________

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last Name: ORKUN DOĞAN

Signature :

ABSTRACT

AN INVESTIGATION ON BELIEF PROPAGATION DECODING OF POLAR CODES

Doğan, Orkun
M.S., Department of Electrical and Electronics Engineering
Supervisor: Assoc. Prof. Dr. Melek Diker Yücel

December 2015, 74 pages

Polar codes are provably symmetric capacity achieving codes for any given binary input discrete memoryless channel, with low encoding and decoding complexities. Polar codes introduced by Erdal Arıkan in 2009 are based on the channel polarization. N binary channels are synthesized out of N copies of binary input discrete memoryless channels, such that as N goes to infinity each of the synthesized channel's capacity goes to either 0 or 1; i.e., the channels are seen purely as noisy or noiseless channels. These synthesized channels are called polarized since they go towards extremes; as N goes to infinity, $NI(W)$ of them are perfect and $N(1-I(W))$ of them are purely noisy channels. One has to send data through the noiseless channels and send frozen bits, which are known by the receiver as well, through purely noisy channels to achieve the channel capacity. Arıkan proposed the Successive Cancellation (SC) decoding with low complexity $O(N\log N)$, and also used the Belief Propagation (BP) decoding that can perform better with the same complexity.

In this thesis, the encoding of polar codes by means of channel polarization is examined and BP decoding that employs round iterations is implemented on a factor graph with $\log_2 N$ stages, for the binary erasure channel (BEC). Each round iteration starts by forwarding only left messages from the encoder output to the input and proceeds with only right messages from the input to the output. The number of round iterations required for BP decoding of polar codes is determined by simulations. The performance dependence of the BP decoding algorithm upon factor graphs, corresponding number of frozen variables and related capacities of the synthesized information bit channels are examined. In addition, the performance of BP decoding using multiple factor graphs with $\log_2 N$ and $(\log_2 N)!$ parallel paths is compared to the case of single factor graph decoding.

Keywords: Channel Polarization, Polar Codes, Reed Muller Codes, Belief Propagation, Factor Graph.

ÖZ

KUTUPSAL KODLARIN İNANÇ YAYILIMI KOD ÇÖZÜMLEMESİ ÜZERİNE ARAŞTIRMA

Doğan, Orkun
Yüksek Lisans, Elektrik ve Elektronik Mühendisliği Bölümü
Tez Yöneticisi: Doç. Dr. Melek Diker Yücel

Aralık 2015, 74 sayfa

Kutupsal kodlar, hafızasız ikil girişli kanallar için simetrik kapasiteye ulaştığı kanıtlanan, kodlayıcısı ve kod çözücüsü düşük karmaşıklıkta kodlardır. Kutupsal kodlar Erdal Arıkan tarafından 2009 yılında kanal kutuplaşmasına dayanılarak bulunmuştur. Kanal kutuplaşması ile, hafızasız ikil girişli kanalın *N* kopyasından sentezlenen *N* ikil kanalın kapasiteleri, N sonsuza giderken ya 0'a ya da 1'e gitmektedir, yani kanallar ya tamamen gürültülü ya da tamamen gürültüsüz olarak görülmektedir. Sentezlenen kanallar sınırlara doğru gittiğinden kutuplaşmış kanallar olarak adlandırılır. N sonsuza giderken bunların $NI(W)$ tanesi mükemmel kanallar, $N(1 - I(W))$ tanesi ise tamamen gürültülü kanallardır. Kanal kapasitesine ulaşmak için gürültüsüz kanallardan bilgi gönderilirken, gürültülü kanallardan ise (alıcı tarafından da bilinen) donuk ikiller gönderilmektedir. Arıkan $O(N\text{log}N)$ düzeyinde düşük karmaşıklığa sahip Ardışık Eleme kod çözümlemesini önerdi, ve aynı karmaşıklıkla daha iyi performans gösteren İnanç Yayılımı kod çözümlemesini de kullandı.

Bu tezde, kutupsal kodların kanal kutuplaşması yardımıyla kodlanması incelenerek döngüsel iterasyon kullanan İnanç Yayılımı kod çözümleyicisisi, $\log_2 N$ aşamalı faktör grafiği üzerinde, ikili silinti kanalları için gerçeklenmiştir. İnanç Yayılımı çözümlemesi için gereken döngüsel iterasyon sayısı benzetimlerle belirlenmis; algoritma performansının faktör grafiklerine, ilgili donuk ikil sayılarına ve sentezlenen bilgi ikil kanalı kapasitelerine bağımlılığı incelenmiştir. Ayrıca çoklu grafik kullanan İnanç Yayılımı çözümlemesinde $\log_2 N$ veya $(\log_2 N)!$ paralel kol olması durumuyla, tek faktör grafikli çözümleme karşılaştırılmıştır.

Anahtar Kelimeler: Kanal Kutupsallaşması, Kutupsal Kodlar, Reed Muller Kodlar, İnanç Yayılımı, Faktör Grafiği.

To my family, friends and people who are reading this page

ACKNOWLEDGMENTS

I've had the chance to work with more talented and intelligent people from each other during my graduate studies. Thanks infinitely to everyone supporting me in my work and social life throughout this process.

The largest contribution in my educational development belongs to Assoc. Prof. Melek Diker Yücel, who has always supported me and transferred her knowledge and experience to me. Because of all these supports and her smiling face, I heartily indicate that it was an honor, pleasure and a big chance to work with her.

My close friends always supported me mentally and shared my distress. I don't want to give a name not to be ashamed for forgetting one of them. So, thanks to all of them collectively.

My largest thanks are for my family, who have always worked for me to live a comfortable and healthy life. I can not thank enough to my dear family who have always shown me the right way patiently, respected my decisions and loved me in the purest sense.

I would like to acknowledge the Scientific and Technological Research Council of Turkey (TÜBİTAK) BİDEB-2228 Scholarship Program for the support they have provided during the course of my graduate education.

TABLE OF CONTENTS

ABSTRACT v

ÖZ vii

ACKNOWLEDGMENTS x

TABLE OF CONTENTS xi

LIST OF TABLES xiii

LIST OF FIGURES xv

LIST OF ABBREVIATIONS xviii

CHAPTERS

1 INTRODUCTION 1

1.1 Belief Propagation Decoding of Polar Codes 3

1.2 Aim and Organization of the Thesis 6

2 CHANNEL POLARIZATION 9

2.1 Error Control Coding 9

2.2 Channel Polarization 11

2.3 Polar Encoding 18

2.4 Polar Code Decoding 22

3 SIMULATION RESULTS 29

3.1 Description of the Decoding Structure Used in Simulations . 30

3.1.1 Belief Propagation Decoding Algorithm 31

3.2 Decoder Structure and the Number of Frozen Variables for the BP Decoding Algorithm 36

3.3 Choice of the Number of Iterations for the BP Decoding Algorithm 40

3.4 Dependence of Bit Channel Capacities on the Decoder Structure 43

3.5 Performance Dependence of the BP Decoding Algorithm on the Decoder Structure - Stage Order “1-...-*n*” versus “*n*-...-1” 46

3.6 Multiple Factor Graph BP Decoding with $\log_2 N$ and $(\log_2 N)!$ Factor Graphs, and Two Proposals: Highest CS and Genie-Chosen Factor Graph Sets 49

3.7 Performance Dependence on the Number of Frozen Variables and Capacity Sum 53

3.8 Performance Comparison of Polar and Reed Muller Codes . . 56

4 CONCLUSION 59

REFERENCES 63

APPENDICES

A REQUIRED NUMBER OF ROUND ITERATIONS FOR THE BP DECODER FOR *N*=128 & 512 67

B CORRELATION BETWEEN THE NUMBER OF FROZEN VARIABLES AND THE CAPACITY SUMS 73

LIST OF TABLES

TABLES

Table 2.1 Capacities of 8 channels synthesized from a BEC(ϵ) for $N = 8$. 22

Table 3.1 An alternative way of displaying the frozen bits (0's in the table) corresponding to all factor graphs of the (8,4) polar code. 38

Table 3.2 For the (32,16) polar code, stage orders of the 12 diagrams having 44 frozen variables, whose input frozen bits are adjusted according to the "5-4-3-2-1" diagram. 39

Table 3.3 Number of decoded codewords out of 1,000 received words of the (32,16) polar code after each round iteration for the stage order 1-...-5 over a BEC(ϵ). 41

Table 3.4 Number of decoded codewords out of 1,000 received words of the (32,16) polar code after each round iteration for the stage order *n*-...-1 over a BEC(ϵ). 42

Table 3.5 Number of round iterations needed for BP decoding of rate 1/2 polar codes over a BEC(ϵ). 43

Table 3.6 Equal capacity sets of the (32,16) polar codes over a BEC(0.3125). . 45

Table 3.7 Decoded bit erasure probabilities of a (32,16) codes on a BEC(ϵ) over 10,000 trials and up to 10 iterations of the BP algorithm. 46

Table 3.8 Decoded bit erasure probabilities of a(128,64) codes on a BEC(ϵ) over 2,000 trials and up to 15 iterations of the BP algorithm. 46

Table 3.9 Decoded bit erasure probabilities of a (512,256) codes on a BEC(ϵ) over 1,000 trials and up to 20 iterations of the BP algorithm. 47

Table 3.10 $n = 6$ cyclic diagrams and their performances for the (64,32) code over 1,000 codewords each having 22 erasures. 50

Table 3.11 19 diagrams with the best 19 CS values for the (64,32) code over 1,000 codewords each having 22 erasures. 51

Table 3.12 5 diagrams that give the best performance for the (64,32) code over 1,000 codewords each having 22 erasures. 52

Table 3.13 The number of undecoded words out of 1,000 words (each with 22 erasures) in each simulation, with multiple factor graph BP decoding for the (64,32) code. 52

Table 3.14 4 diagrams that give the best performance for the (64,32) code over 1,000 codewords each having 22 erasures. 53

Table 3.15 Performance of BP decoding (128,64) polar code with different factor graphs and (128,64) RM code over a BEC(0.35). 57

Table A.1 Number of decoded codewords out of 1,000 received words of the (128,64) polar code after each round iteration for the stage order 1-...-*n* over a BEC(ϵ). 68

Table A.2 Number of decoded codewords out of 1,000 received words of the (128,64) polar code after each round iteration for the stage order *n*-...-1 over a BEC(ϵ). 69

Table A.3 Number of decoded codewords out of 1,000 received words of the (512,256) polar code after each round iteration for the stage order 1-...-*n* over a BEC(ϵ). 70

Table A.4 Number of decoded codewords out of 1,000 received words of the (512,256) polar code after each round iteration for the stage order *n*-...-1 over a BEC(ϵ). 71

Table B.1 Distribution of equal-CS sets (each having 12 elements) for the (32,16) code over a BEC(0.3125), expressed differently from Table 3.6. . . 73

Table B.2 Distribution of equal-CS sets (each having 4 elements) for the (64,32) polar code over a BEC(0.34375). 74

Table B.3 Distribution of equal-CS sets (each having 2 elements) for the (128,64) polar code over a BEC(0.35). 75

LIST OF FIGURES

FIGURES

Figure 1.1 Performance comparison of decoding schemes of polar code (solid curves) and WiMax LDPC code (dashed curve) (reproduced from [Tal and Vardy, 2011]). 3

Figure 2.1 Basic block diagram of a binary communication system. 10

Figure 2.2 Binary Discrete Memoryless Channel (B-DMC). 10

Figure 2.3 Binary Erasure Channel (BEC) and Binary Symmetric Channel (BSC). 11

Figure 2.4 Channel combining. 12

Figure 2.5 The basic transformation of channel combining. 12

Figure 2.6 4- input, 4-output W_4 channel. 13

Figure 2.7 Channel splitting. 14

Figure 2.8 W^- channel after splitting. 15

Figure 2.9 W^+ channel after splitting. 15

Figure 2.10 Channel splitting for $N = 4$. 16

Figure 2.11 Capacities of polarized channels for $N = 32, 128, 512, 2048$ over a BEC(0.5). 19

Figure 2.12 Channel capacities for $N = 8$ over a BEC(0.5). 21

Figure 2.13 Encoding of (8,4) polar code with $[1, 0, 1, 0]$ data bits. 21

Figure 2.14 Tanner graph of the *H* matrix of the (8,4) polar and RM codes. . . . 24

Figure 2.15 Factor graph representation of the *H* matrix of the (8,4) polar and RM codes. 25

Figure 2.16 Word error rate of SCL decoding (reproduced from [Tal and Vardy, 2011]). 26

Figure 2.17 Performance comparison of decoding schemes of polar code and WiMax LDPC code (reproduced from [Tal and Vardy, 2011]). 27

Figure 3.1 BP decoding diagram (or the factor graph) for an $N = 8$ code with $n = 3$ stages and $(n + 1)N = 32$ variable nodes (where 6 filled nodes correspond to the frozen variables of the diagram, 2 of them are generated by the 4 input frozen bits of the (8,4) code). 30

Figure 3.2 BP process unit (subscripts I and O indicate the input and output, i shows the stage, j and k show the rows of the diagram, L and R are the left and right messages, t is the iteration number). 33

Figure 3.3 BP decoding diagram that shows the input and output variables of each process unit in detail, for an $N = 8$ code with $n = 3$ stages. 35

Figure 3.4 Different decoding diagrams for the (8,4) polar code. 37

Figure 3.5 Distribution of 120 diagrams corresponding to the given number of frozen variables for the (32,16) polar code. 39

Figure 3.6 Frozen variables (shown by 0's) of the 5 cyclic diagrams for the (32,16) polar code, corresponding to stage permutations 1-2-3-4-5, 2-3-4-5-1, 3-4-5-1-2, 4-5-1-2-3 and 5-1-2-3-4 respectively. 40

Figure 3.7 Capacities of 2-1 stage order for a BEC(0.5). 44

Figure 3.8 Capacities of 1-2 stage order for a BEC(0.5). 44

Figure 3.9 Decoded bit erasure probabilities of (32,16) codes over a BEC(ϵ). . 47

Figure 3.10 Decoded bit erasure probabilities of (128,64) over a BEC(ϵ) codes. 48

Figure 3.11 Decoded bit erasure probabilities of (512,256) over a BEC(ϵ) codes. 48

Figure 3.12 Decoded bit erasure probabilities of (32,16), (128,64), (512,256) codes over a BEC(ϵ). 48

Figure 3.13 Minimum, maximum and average number of undecoded words for a (64,32) code over a BEC(0.34375). 54

Figure 3.14 Average number of undecoded words for a (64,32) code versus the capacity sum over a BEC(0.34375). 55

Figure 3.15 Capacity sums over a BEC(0.34375) versus number of frozen variables for all stage orders of the (64,32) code. 55

Figure 3.16 Average capacity sums over a BEC(0.34375) versus number of frozen variables for all stage orders of the (64,32) code. 56

LIST OF ABBREVIATIONS

B-DMC	Binary-input Discrete Memoryless Channel
BEC(ϵ)	Binary Erasure Channel with Erasure Probability ϵ
BEP	Bit Erasure Probability
BER	Bit Error Ratio
BP	Belief Propagation
BSC	Binary Symmetric Channel
CRC	Cyclic Redundancy Check
CS	Capacity Sum
FER	Frame Error Ratio
FV	Frozen variable
LDPC	Low Density Parity Check Matrix
MAP	Maximum A Posteriori
ML	Maximum Likelihood
MPA	Message Passing Algorithm
MS	Min-Sum
PE	Processing Element
RM	Reed Muller
SC	Successive Cancellation
SCL	Successive Cancellation List
SMS	Scaled Min-Sum

CHAPTER 1

INTRODUCTION

The main aim of a communication system is to transmit information from the source to the destination reliably, through a noisy medium. As clarified in Shannon's seminal work [Shannon, 1948], a suitable coding method is required to achieve this purpose, which adds redundancy to data in a clever way.

For channel coding schemes to approach the Shannon limit, known as the channel capacity, the block length of the codeword at the encoder output needs to be large. As computational resources have developed, it becomes feasible to use larger and larger block lengths to approach the channel capacity. At the same time, corresponding encoding and decoding algorithms should have low computational complexities for practical applications.

The first algebraic codes developed were Hamming, Golay [Golay, 1949] and Reed-Muller ([Reed, 1954], [Muller, 1954]) codes, followed by BCH codes ([Hocquenghem, 1959], [Bose and Ray-Chaudhuri, 1960]) and Reed-Solomon codes [Reed and Solomon, 1960]. There are efficient decoding algorithms for all these algebraic codes and they are used successfully in today's CDs, DVDs and modems.

An improvement in performance was achieved with probabilistic decoding of codewords, i.e., using the channel outputs directly in the decoding algorithm, instead of giving hard decisions on symbols. Convolutional codes introduced by Elias [Elias, 1955] were suitable for such decoding; and proposals of efficient but suboptimum decoding algorithms like Viterbi [Viterbi, 1967] and BCJR [Bahl et al., 1974] speeded up their practical use.

Low density parity-check (LDPC) codes constructed by Gallager [Gallager, 1962] together with a low-complexity iterative decoding algorithm were very powerful codes, but this was not realized until 1995 because of the low computational resources of 60's. In 1995, MacKay and Neal found some codes having very good performance with belief propagation (BP) decoding, based on sparse parity-check matrices [MacKay and Neal, 1995]. Later, it was understood that these codes were a special form of LDPC codes and the belief propagation decoding was equivalent to the probabilistic decoding suggested by Gallager in 1962.

Turbo codes introduced by Berrou, Glavieux and Thitimajshima also achieved performance close to channel capacity, using a decoder with linear complexity [Berrou and Glavieux, 1996]. They are composed of two concatenated convolutional codes, with an interleaver between them. Decoding uses the BCJR algorithm iteratively in two branches, with mutual feedback from one to the other, to help each other's decisions. The original turbo code [Berrou and Glavieux, 1996] performed as well as the best existing schemes of late 90's, using half as much power.

Turbo codes and LDPC codes were unified under the framework of "codes on graphs" by Wiberg, Loeliger and Kötter ([Wiberg et al., 1995], [Wiberg, 1996]). Within the framework of "codes on graph", their decoding algorithms simply turned out to be different forms of the same algorithm. This framework also provided a useful bridge between sparse graph codes and other fields like machine learning, statistical mechanics and computer science [Korada, 2009].

Polar codes are linear block codes introduced by Arıkan that are based on channel polarization; which consists of combining and splitting N copies of a given binary channel suitably, so that the capacities of the polarized N new channels approach either to 0 or 1 [Arıkan, 2009]. After polarization, K channels with the highest capacity are used for transmission and the remaining $N-K$ bits are frozen. The choice of the frozen bits is not universal but it depends on the specific channel at hand.

Arıkan proves that polar codes achieve symmetric channel capacity with low encoding and decoding complexity. He originally suggests the successive cancellation (SC) decoding algorithm that has the complexity $O(N \log N)$ [Arıkan, 2009]. Korada argues that belief propagation (BP) decoding with the same complexity, $O(N \log N)$, applied to polar codes gives better results, and this is expected since SC decoding can be considered as a particular instance of BP decoding for the BEC [Korada, 2009]. He obtains the performance of the BP decoder as lying roughly half way between that of the SC decoder and that of the MAP decoder.

On the other hand, the polar code performance close to the ML bound is achieved with list decoding [Tal and Vardy, 2011] and simulations reveal that the (2048, 1008) polar code with a 16-bit CRC using a list-of-32 decoder shows very similar performance to the WiMAX (2304,1152) LDPC code, as shown in Figure 1.1 [Arıkan et al., 2015].

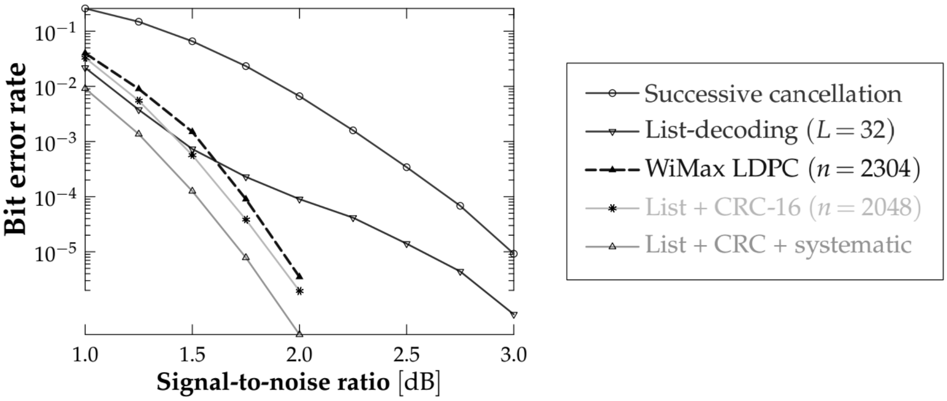


Figure 1.1: Performance comparison of decoding schemes of polar code (solid curves) and WiMax LDPC code (dashed curve)
(reproduced from [Tal and Vardy, 2011]).

1.1 Belief Propagation Decoding of Polar Codes

Polar codes can be seen as a special case of Reed Muller (RM) codes [Hussami et al., 2009] and basically, a few of the generator matrix rows differ for (N, K) polar and RM codes, if $N \geq 32$ [Arıkan, 2008]. Forney expressed RM codes and their Belief Propagation (BP) decoding [Forney Jr, 2001]. Arıkan gave experimental results on the performance of polar codes with BP decoding, showing that polar codes have better bit error ratio (BER) performance than RM codes; and the performance difference becomes more visible with increasing block length [Arıkan, 2008].

Hussami, Korada and Urbanke show that the performance of BP decoding is better than SC decoding [Hussami et al., 2009], and for a BEC(0.5) it lies roughly half way between the SC and MAP decoders [Korada, 2009]. Korada also considers BP decoding with multiple factor graphs, utilizing $\log_2 N$ parallel paths that are cyclic shifts of each other. He argues that the performance improves significantly by using multiple factor graphs, while the decoding complexity that increases linearly with the number of factor graphs becomes $O(N(\log N)^2)$ [Korada, 2009].

Eslami and Pishro-Nik investigate the stopping sets and the stopping distance for polar codes under BP decoding. The bigger stopping distance yields to a better error floor and it is shown in the paper that the stopping distance grows as $O(\sqrt{N})$ for polar codes [Eslami and Pishro-Nik, 2010]. Although LDPC codes have better BER performance and smaller encoding/decoding complexities, polar codes have better error floor. By using this error floor feature of polar codes, BP decoding with a guessing algorithm is suggested. Usually, whenever the BP decoding algorihm fails, there remains a few undecoded bits, although some of the erasures are filled correctly. In the guessing algorithm, one of these undecoded bits is chosen and guessed randomly, and the decoding continues with that guess until the decoding succeeds or reaches to the maximum number of guesses. The simulation results with a (8192,4096) polar code show that the guessing algorithm provides a magnitude of 2 improvement for the BEC and an SNR gain of 0.3 dB for the Gaussian channel [Eslami and Pishro-Nik, 2010].

The girth, which is the shortest cycle in the Tanner graph of the code is also considered in the same paper. For $N = 4$ polar codes, the girth is 12 and because of the recursive construction of larger block lengths from the small ones, all other sizes of the polar code have the same girth. For LDPC codes 12 is a reasonable girth size that is aimed at, so one may say that polar codes have a good size of girth [Eslami and Pishro-Nik, 2010].

Eslami and Pishro-Nik give another method to improve the performance of polar codes under BP decoding [Eslami and Pishro-Nik, 2013]. In this paper, they offer a new rule to increase the stopping distance of the polar codes. For a (8192, 4096) polar code, information bits with row weight smaller than 2^8 are replaced by frozen bits with row weight larger than 2^8, starting from the smallest Bhattacharyya parameter.

With this new rule, the performance of the SC decoder decreases because the rule forces the use of channels with smaller capacities. On the contrary, for BP decoding the performance increases since channels with higher stopping distance are used. Eslami and Pishro-Nik also suggest a concatenated polar - LDPC code to be used in Optical Transport Networks, polar as the outer and LDPC as the inner code. Decoding is performed with BP decoding for both polar and LDPC codes. This concatenated design performs significantly better than some existing coding schemes such as RS(2720, 2550) and G.975.1 LDPC codes [Eslami and Pishro-Nik, 2013].

Guo, Qin, Fabregas and Siegel propose an enhanced concatenated polar code under BP decoding [Guo et al., 2014]. As in [Eslami and Pishro-Nik, 2013], they use the LDPC code as the outer code, and polar code as the inner code. In case of finite length polar codes, channels don't polarize perfectly. So, some intermediate channels are used to transmit data. To prevent errors in these channels, they suggest to use the good channels for the uncoded data, the bad channels for the frozen bits and the intermediate channels for the coded data. Thus, these concatenated intermediate channels become as well as good channels. For an $N = 4096$ polar code with SNR $= 4$ dB, they obtain approximately 0.3 dB improvement in the frame error ratio (FER).

Conventional BP decoding algorithm is not suitable for hardware implementation. So, to avoid overflow, min-sum (MS) approximation is used in [Pamuk, 2011], but this imposes some sacrifice from performance. Yuan and Parhi propose scaled min-sum (SMS) BP decoding [Yuan and Parhi, 2014a]. By adding a scale like 0.9375, the SMS algorithm improves the performance about 0.5 dB for a (1024,512) polar code. Thus, SMS compensates the performance loss of the MS approximation.

Yuan and Parhi propose a hybrid decoding for polar codes, which is a combination of BP and SC decoding methods [Yuan and Parhi, 2014b]. In the first step, BP decoding tries to decode the received word. If BP decoding cannot decode the word successfully, SC decoding tries to decode the same word. The key part of this hybrid decoding method is that if BP decoding fails to decode the word after the maximum given number of iterations, it sends the de-noised word to SC decoding. It means that the BP decoding algorithm decreases the noise with iterations and the input to the SC decoding becomes less erased. The simulation results show that the hybrid decoding has a performance advantage around 0.2 dB over BP decoding for all SNR values. For low SNRs the latency difference is high between BP and hybrid decoders, but as the SNR increases, the latency difference gets very small and it is reasonable to lose that much from latency in return for the 0.2 dB performance gain.

Xu, Che and Choi propose an express journey for the BP decoding [Xu et al., 2015]. Their main idea is to simplify the operations by not visiting some variables and using constituent codes in the factor graph that will not affect the decoding performance. This idea makes the implementation of low energy decoders possible in hardware. With this simplified decoding, the number of computations for the (1024,512) polar code decreases by $\%60$ at $E_b/N_0 = 3.5$ dB. Xu, Che and Choi argue that although the conventional BP decoders compute the left and right messages simultaneously, they suggest first to calculate left messages only, while going to the left from the rightmost to the leftmost of the factor graph; and then right messages only, while going to the right from the leftmost to the rightmost. So, they define a round iteration as the combination of all left messages followed by all of the right messages. The number of computations does not change with this method but the number of iterations required to decode the word successfully is reduced. For example for $E_b/N_0 = 3.5$ dB and $(N, K) = (1024, 512)$, the average number of iterations decreases from 24 to 4. This corresponds to $\%83$ efficiency in the number of computations. By combining both the simplified algorithm and the idea of round iteration, the average number of operations for decoding decreases by $\%90$. Simulation results [Xu et al., 2015] show that the performance is improved significantly and becomes as good as scaled min-sum (SMS) version of the BP algorithm [Yuan and Parhi, 2014a].

1.2 Aim and Organization of the Thesis

The purpose of this thesis is to examine polar codes, analyze their performance and compare with Reed-Muller codes over the binary erasure channel (BEC), under belief propagation (BP) decoding. Performance dependence of polar codes with respect to parameters like *i*) the chosen factor graph (diagram), *ii*) corresponding number of frozen bits, *iii*) the depth of frozen bits, *iv*) sum of capacities for the chosen (unfrozen) input bits are investigated. Additionally, multiple factor graph BP decoding of polar codes that uses ($\log_2 N$) parallel paths is compared with the one that uses ($\log_2 N$)! parallel paths.

In Chapter 2, polar codes and the concept of channel polarization is reviewed. The encoding structure of polar codes, the choice of the generator matrix and frozen bits for a given binary discrete memoryless channel, and decoding by successive cancellation & belief propagation is explained.

In Chapter 3, some decoding simulations are presented for the binary erasure channel (BEC) and compared with similar results from the literature. Simulation results related to the decoding of Reed-Muller, polar and adaptive polar codes by the BP decoding algorithm are given. The structure of the implemented BP decoder is explained with emphasis on its discriminating feature of "to the leftmost with left/to the rightmost with right messages" as in [Xu et al., 2015]. The choice of a decoding diagram that is also called a ‘factor graph’, corresponding frozen variables, the proper choice for the minimum sufficient number of BP iterations, the correlation between bit channel capacities and the number of frozen variables are discussed. The performance of the BP decoder is found for two extreme versions of the single factor graph case; and also for multiple factor graph decoding with $\log_2 N$ cyclic diagrams as in [Korada, 2009], [Hussami et al., 2009], and with all possible $(\log_2 N)!$ diagrams.

Chapter 4 summarizes and discusses the conclusions of this work.

CHAPTER 2

CHANNEL POLARIZATION

In this chapter, we review the idea of the channel polarization and polar codes presented by Arıkan [Arıkan, 2009]. First, some basic information about error control coding is given in Section 2.1. Section 2.2 summarizes how the synthesized channels polarize after combining and splitting N copies of a binary-input discrete memoryless channel, as N gets larger. The capacities of the channels synthesized from a binary erasure channel with erasure probability 0.5 are computed for $N = 4$. In Section 2.3, we describe the encoding structure of polar codes, the choice of the generator matrix and frozen bits for a given binary discrete memoryless channel. Finally in Section 2.4, the successive cancellation (SC) and belief propagation (BP) decoding methods for polar codes are explained briefly.

2.1 Error Control Coding

Error control coding is a branch of communication that deals with reliable transmission of information over noisy channels while using power and bandwidth resources efficiently. Figure 2.1 depicts an extremely simplified model for a binary communication system, in which the encoder and decoder serve the purpose of error control coding. The information U is encoded and sent through a binary-input discrete memoryless channel (B-DMC). At the receiver end, it is decoded as $\hat{U}$ with the ultimate purpose that $\hat{U} = U$.

Most of the commonly used codes are linear codes. A linear block code is a subspace of a vector space; so, any linear combination of its elements (codewords) is also a

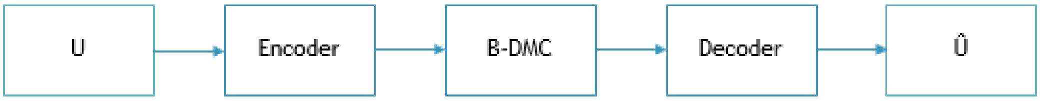


Figure 2.1: Basic block diagram of a binary communication system.

codeword. K symbols (bits for binary codes) of information are mapped onto a vector of length N, where $K < N$, to obtain an (N, K) linear code with rate, $R = K/N$. For an (N, K) code, the number of redundant symbols is $N - K$. One is interested in the minimum amount of redundancy (or maximum R) that can achieve error-free communications.

Claude E. Shannon's Capacity Theorem [Shannon, 1948] states that error-free transmission is possible as long as the transmitter does not exceed the channel's capacity C; i.e., if $R \leq C$, the probability of error can reach 0 as N goes to infinity.

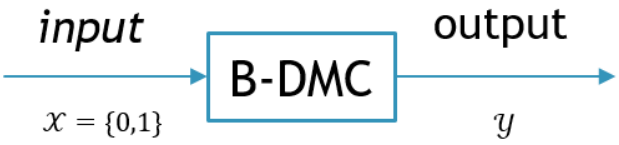


Figure 2.2: Binary Discrete Memoryless Channel (B-DMC).

Considering the binary discrete memoryless channel $W : \mathcal{X} \rightarrow \mathcal{Y}$ in Figure 2.2, where $\mathcal{X} = \{0, 1\}$ is the input, $\mathcal{Y}$ is the output and $W(y|x)$ is the channel transition probability for $x \in \mathcal{X}$ and $y \in \mathcal{Y}$, the symmetric capacity of the B-DMC is

$$I(W) \triangleq \frac{1}{2} \sum_{y \in \mathcal{Y}} \sum_{x \in \mathcal{X}} W(y|x) \log \frac{W(y|x)}{\frac{1}{2}W(y|0) + \frac{1}{2}W(y|1)}. \tag{2.1}$$

Another important parameter for a B-DMC is measure of reliability $Z(W)$ (2.2), called as the Bhattacharyya parameter . $Z(W)$ is an upper bound for the probability of maximum likelihood (ML) decision error [Arıkan, 2009].

$$Z(W) \triangleq \sum_{y \in \mathcal{Y}} \sqrt{W(y|0)W(y|1)} \tag{2.2}$$

Two of the most common and simple binary discrete memoryless channel models used in coding and information theory are the binary erasure channel (BEC) and the binary symmetric channel (BSC), which are given in Figure 2.3. For the binary erasure channel, each information bit is transmitted either correctly with probability $1-\epsilon$ or erased with probability ϵ. It is not possible to receive 1 while the information is 0 or vice versa. Such a channel with probability ϵ is abbreviated as BEC(ϵ). The channel capacity (C) of the BEC(ϵ) is $1 - \epsilon$. On the other hand, the binary symmetric channel transmit each information bit either correctly with probability $1 - p$ or incorrectly with probability p. The channel capacity of the BSC is $1 - H(p)$, where $H(p) = -p\text{log}p - (1-p)\text{log}(1-p)$ is the binary entropy function.

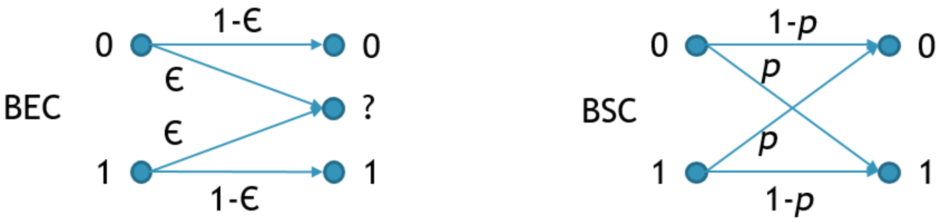


Figure 2.3: Binary Erasure Channel (BEC) and Binary Symmetric Channel (BSC).

2.2 Channel Polarization

The main idea of the channel polarization is to construct N binary-input channels $(W_N^{(i)} : 1 \leq i \leq N)$ out of N independent binary-input discrete memoryless channels *W*, such that as *N* goes large, the newly constructed channels polarize, i.e., the channel capacity of each channel goes to extremes, either to 0 or 1. There are two phases for channel polarization, *i*) channel combining and *ii*) channel splitting.

i) Channel Combining

In this phase, *N* copies of binary-input discrete memoryless channel *W*, are combined recursively to produce a vector channel called W_N, where *N* is the code length. As seen in Figure 2.4, N independent channels are transformed into a channel with N inputs and N outputs.

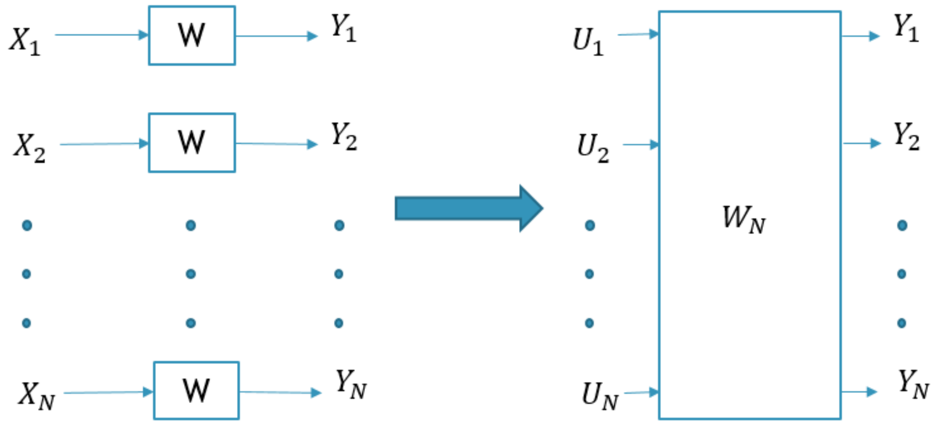


Figure 2.4: Channel combining.

In Figure 2.5, two independent B-DMC's W are combined to create a new vector channel of size 2, $W_2 : \mathcal{X}^2 \to \mathcal{Y}^2$. This is the basic transformation used in channel combining, and the recursion starts with this transformation.

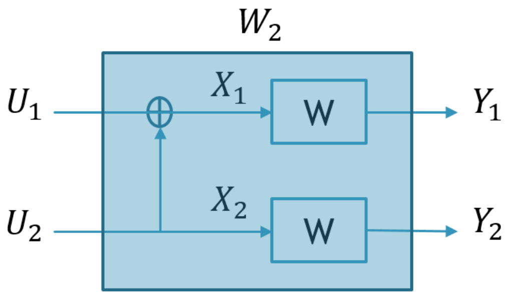


Figure 2.5: The basic transformation of channel combining.

The transition probability of the vector channel W_2 is

$$W_2(y_1, y_2|u_1, u_2) = W(y_1|u_1 \oplus u_2)W(y_2|u_2). \tag{2.3}$$

The linear transform between vector $U = (U_1, U_2)$ and $X = (X_1, X_2)$ is a one-to-one mapping, and if we take U_i's as identically independent distributed (iid) then X_i's are also iid. So the following equation holds:

$$\begin{aligned} I(W_2) = I(U_1, U_2; Y_1, Y_2) = I(X_1, X_2; Y_1, Y_2) \\ = I(X_1; Y_1) + I(X_2; Y_2) = 2I(W) \end{aligned} \tag{2.4}$$

It is clear from (2.4) that, there is no loss in total channel capacity.

As mentioned before, channel combining is done recursively and the recursion starts as shown in 2.5. In the second step of the recursion, 4-input, 4-output channel W_4 is combined by using 2 independent W_2 channels as indicated in Figure 2.6.

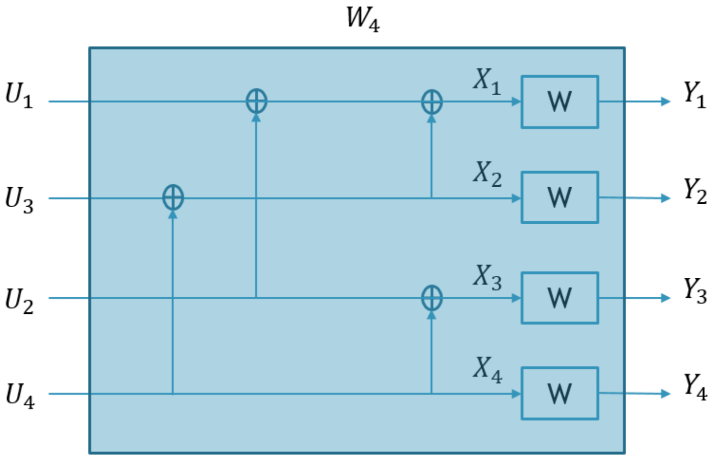


Figure 2.6: 4- input, 4-output W_4 channel.

The transition probability of this vector channel is

$$\begin{aligned} W_4(y^4|u^4) = W(y_1|u_1 \oplus u2 \oplus u3 \oplus u4)W(y_2|u_3 \oplus u4)W(y_3|u_2 \oplus u4)W(y_4|u4) \\ = W_2(y_1, y_2|u_1 \oplus u_2, u_3 \oplus u4)W_2(y_3, y_4|u_2, u_4). \end{aligned} \tag{2.5}$$

Channel combining of N input vector channel $W_N : \mathcal{X}^N \rightarrow \mathcal{Y}^N$ is done recursively in $\log_2 N$ steps and the transition probability is given as

$$W_N(y^N|u^N) = W_{N/2}(y^{N/2}|u_o^N \oplus u_e^N)W_{N/2}(y_{N/2+1}^N|u_e^N), \tag{2.6}$$

where $u_o^N = \{u_1, u_3, u_5, ..., u_{N-1}\}$ and $u_e^N = \{u_2, u_4, u_6, ..., u_N\}$. More details about channel combining can be found in [Arıkan, 2009].

ii) Channel Splitting

In the second phase of the channel polarization, the vector channel that is combined from N many W's as explained in the previous section is split into N binary channels as seen in Figure 2.7. After this phase, polarized channels $W_N^{(i)}$ are obtained.

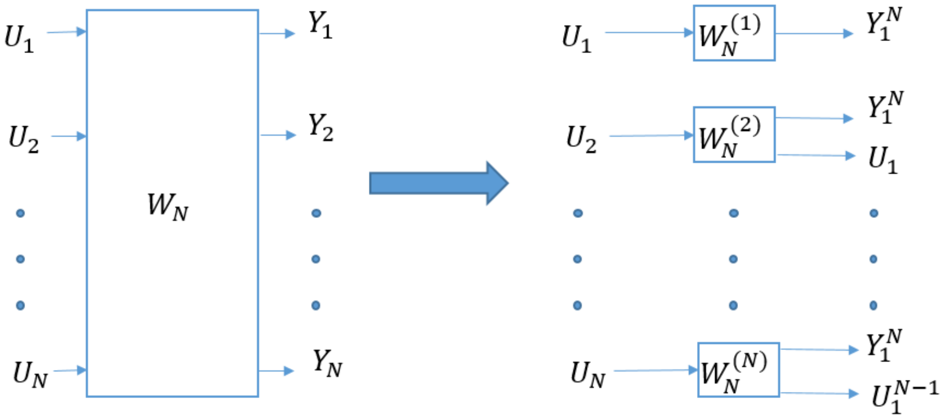


Figure 2.7: Channel splitting.

Channel splitting can be explained starting from the basic transformation given in Figure 2.5. There are 2 inputs, U_1 and U_2, so there will be 2 binary channels after splitting. The mutual information of W_2 channel is $I(U_1, U_2; Y_1, Y_2)$, and by using the chain rule of mutual information it can be split into two terms as

$$2I(W) = I(U_1, U_2; Y_1, Y_2) = I(U_1; Y_1, Y_2) + I(U_2; Y_1, Y_2, U_1). \qquad (2.7)$$

One can define a channel with the input U_1 and the outputs Y_1, Y_2, while considering U_2 as random. The mutual information of this channel is $I(U_1; Y_1, Y_2)$), the first term on the right hand side of (2.7). The corresponding channel is $W_2^{(1)} : \mathcal{X} \to \mathcal{Y}^2$, given in Figure 2.8, which is called W^-. The transition probability of this channel is

$$W_2^{(1)}(y_1, y_2|u_1) = \frac{1}{2} \sum_{u_2 \in \{0,1\}} W(y_1|u_1 \oplus u_2) W(y_2|u_2). \qquad (2.8)$$

Another channel to be defined takes U_2 as the input and Y_1, Y_2, U_1 as the outputs. The mutual information of this channel is $I(U_1; Y_1, Y_2, U_1)$, that is the last term of (2.7).

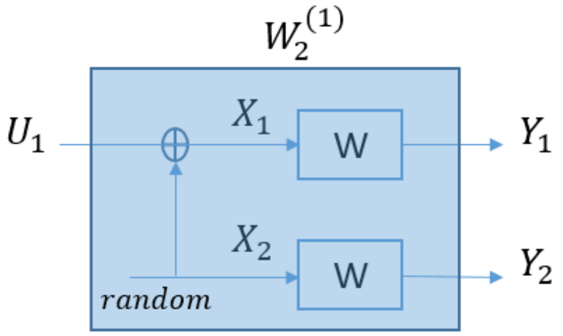


Figure 2.8: W^- channel after splitting.

The corresponding channel is $W_2^{(2)} : \mathcal{X} \rightarrow \mathcal{Y}^2 \times \mathcal{X}$, which is also called W^+ 2.9. The transition probability of this channel is given by

$$W_2^{(2)}(y_1, y_2, u_1|u_2) = \frac{1}{2}W(y_1|u_1 \oplus u_2)W(y_2|u_2). \tag{2.9}$$

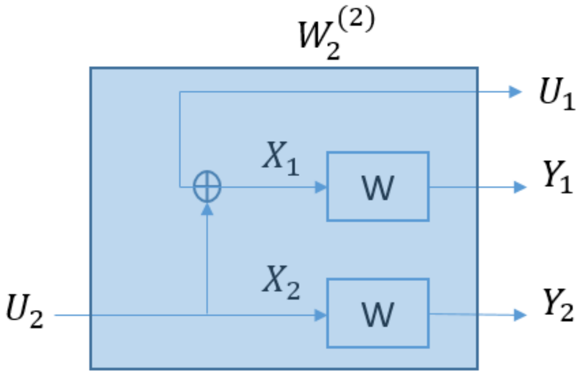


Figure 2.9: W^+ channel after splitting.

The mutual information of W^+ channel can be written as

$$\begin{aligned} I(W^+) = I(U_2; Y_1, Y_2, U_1) &= H(U_2) - H(U_2|Y_1, Y_2, U_1) \\ &\geq H(U_2) - H(U_2|Y_2) = I(U_2; Y_2) = I(W). \end{aligned} \tag{2.10}$$

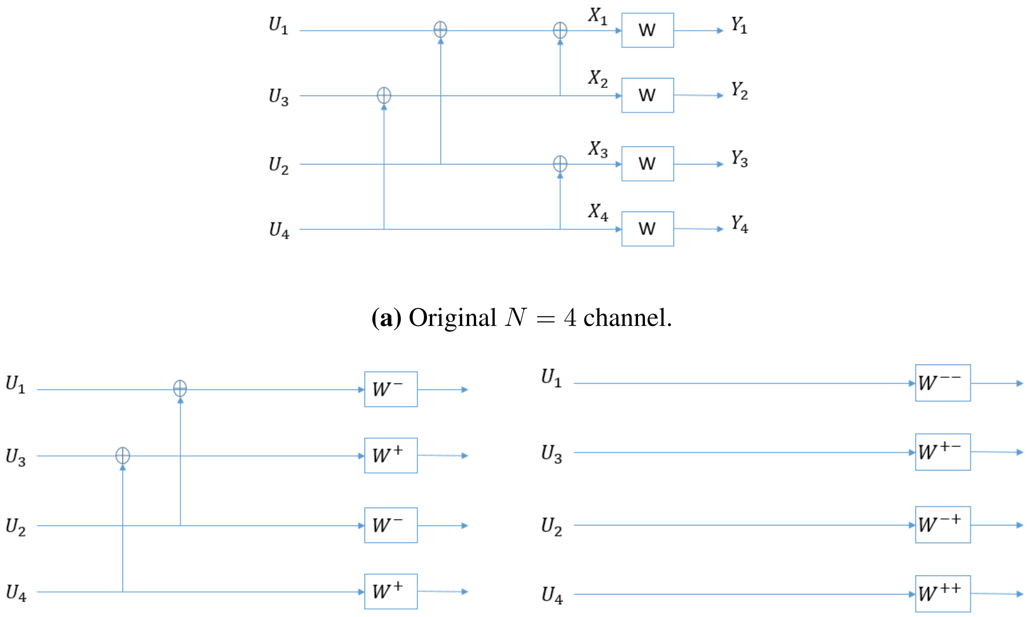
Using (2.7) and (2.10), one concludes that

$$I(W^-) \leq I(W) \leq I(W^+), \tag{2.11}$$

with equality if and only if $I(W)$ is at the extremes, 0 or 1.

It can be seen that, $I(W^-)$ and $I(W^+)$ are pushed to the extremes 0 and 1 while their sum is still $2I(W)$. Polarizing starts at this point.

For $N = 4$, channel splitting can be seen graphically in Figure 2.10, where the same thing for $N = 2$ is simply repeated. There are two W^- and two W^+ channels after the first splitting. Two W^- channels are split into W^{--} and W^{-+}, and two W^+ channels are split into W^{+-} and W^{++} channels.



(a) Original $N = 4$ channel.

(b) First step of splitting.

(c) Second step of splitting.

Figure 2.10: Channel splitting for $N = 4$.

In general, N binary input channels are split by using the chain rule of mutual information

$$\begin{aligned} NI(W) &= I(X^N; Y^N) \\ &= I(U^N; Y^N) \\ &= \sum_{i=1}^{N} I(U_i; Y^N, U^{i-1}) \end{aligned} \quad . \tag{2.12}$$

These N channels are represented by $W_N^{(i)} : \mathcal{U} \to \mathcal{Y}^N \times \mathcal{U}^{i-1}, 1 \leq i \leq N$. The transition probabilities of these channels

$$W_N^{(i)}(y_1^N, u_1^{i-1}|u_i) \triangleq \sum_{u_{i+1}^N \in \mathcal{X}^{N-i}} \frac{1}{2^{N-1}} W_N(y_1^N|u_1^N), \tag{2.13}$$

where u_i is the input and (y_1^N, u_1^{i-1}) is the outputs of the $W_N^{(i)}$ channel.

Synthesized Channel Capacities After Combining and Splitting of a BEC

Now, we will show the computation of the synthesized channel capacities for a BEC, W, with erasure probability $\epsilon = 0.5$. First the capacities of W^- and W^+ will be calculated. For the channel W^- depicted in Figure 2.8, $u_1 = x_1 + x_2$. Hence, to estimate u_1, both x_1 and x_2 are needed, they must not be erasures. So the erasure probability of this channel is $\epsilon^- = 1 - (1-\epsilon)^2$. For the channel W^+ shown in Figure 2.9, there are 2 observations, $u_2 = x_2$ and $u_2 = x_1 + u_1$. To estimate u_2, only one of these observations is enough, and we assume that u_1 is known from the channel W^-. So the erasure probability of the channel W^+ is $\epsilon^+ = \epsilon^2$. For $\epsilon = 0.5, \epsilon^- = 0.75$ and $\epsilon^- = 0.25$.

To calculate the erasure probabilities of $W^{--}, W^{-+}, W^{+-}, W^{++}$, we use the erasure probabilities of W^- and W^+ found in the previous step. The erasure probability of W^{--} is $\epsilon^{--} = 1 - (1-\epsilon^-)^2$, the erasure probability of W^{-+} is $\epsilon^{-+} = (\epsilon^-)^2$, the erasure probability of W^{+-} is $\epsilon^{+-} = 1 - (1-\epsilon^+)^2$, and the erasure probability of W^{++} is $\epsilon^{++} = (\epsilon^+)^2$. Hence erasure probabilities for $\epsilon = 0.5$ are computed as

$$\begin{aligned} \epsilon^{--} &= 0.9375 \\ \epsilon^{-+} &= 0.5625 \\ \epsilon^{+-} &= 0.4375 \\ \epsilon^{++} &= 0.0625 \end{aligned}.$$

For a BEC, W, with erasure probability ϵ, which is abbreviated as BEC(ϵ), the capacity is $I(W) = 1 - \epsilon$, so the capacities of the channels for $N = 4$, corresponding to the erasure probabilities computed above are

$$
\begin{aligned}
I(W^{--}) &= 0.0625 \\
I(W^{-+}) &= 0.4375 \\
I(W^{+-}) &= 0.5625 \\
I(W^{++}) &= 0.9375
\end{aligned}
.
$$

In Figure 2.11, capacities of the channels synthesized from a BEC(0.5) are calculated for $N = 32, 128, 512, 2048$. One can see that as N grows large, the capacities of channels reach to the extremes and the fraction of channels with average capacity gets smaller. As N goes to infinity, the fraction of good channels and bad channels will both be equal to 0.5 (since $\epsilon = 1 - \epsilon = 0.5$). The following theorem that explains the behavior in Figure 2.11 is given by Arıkan and proven in the related paper [Arıkan, 2009].

Theorem 1 *For any B-DMC, while N goes to infinity, the synthesized channels polarize such that for any $\delta \in (0, 1)$, we have*

$$\lim_{N\to\infty} \frac{\textit{Number of channels with } I(W_N^{(i)}) \in (1-\delta, 1]}{N} = I(W) \textit{ and}$$

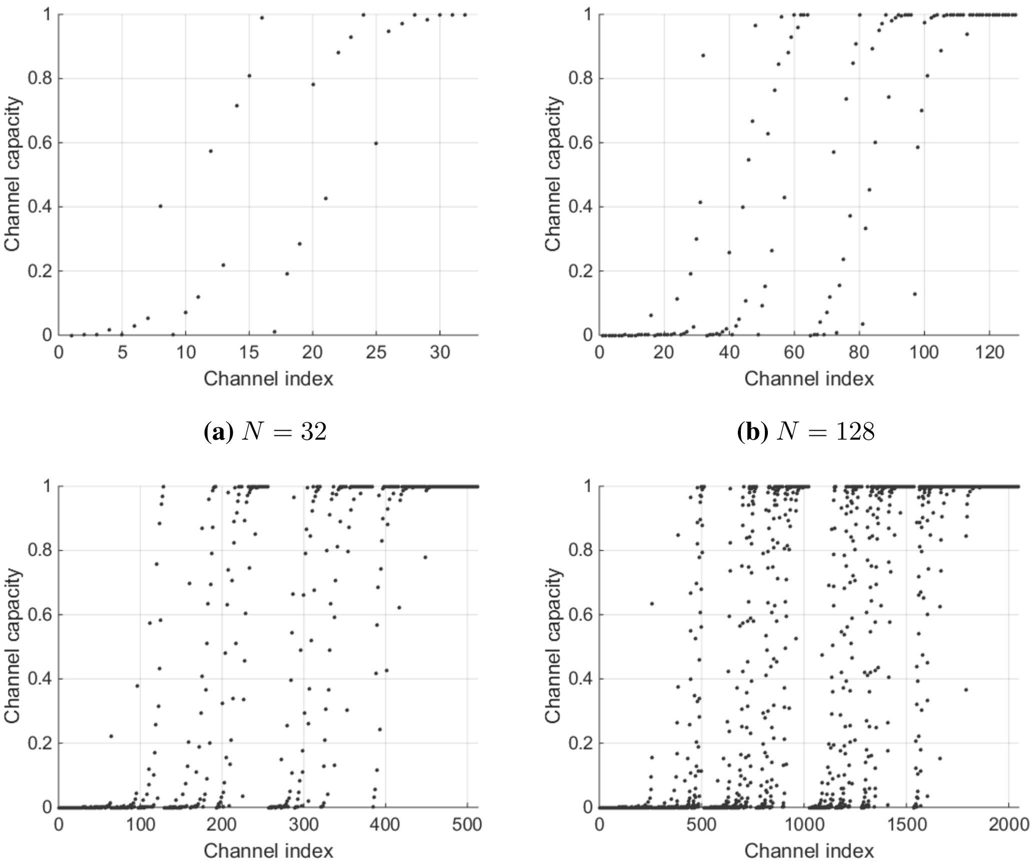
$$\lim_{N\to\infty} \frac{\textit{Number of channels with } I(W_N^{(i)}) \in [0, \delta)}{N} = 1 - I(W).$$

This implies that as N goes to infinity, the capacity of each channel goes to either 0 or 1 precisely, and the fraction of channels with good capacity is $I(W)$.

2.3 Polar Encoding

Polar codes are linear block codes, such that any linear combination of its codewords is also a codeword. The main idea of polar coding is to send data over the $W_N^{(i)}$ channels with $I(W_N^{(i)})$ tending to 1, and freeze the $W_N^{(i)}$ channels with $I(W_N^{(i)})$ is tend to 0. Channel inputs satisfy

$$x_1^N = u_1^N F^{\otimes n}, \tag{2.14}$$



(a) $N = 32$ **(b)** $N = 128$

(c) $N = 512$ **(d)** $N = 2048$

Figure 2.11: Capacities of polarized channels for $N = 32, 128, 512, 2048$ over a BEC(0.5).

where $n = \log_2 N$ and $F^{\otimes n}$ is the n^{th} Kronecker power of $F = \left(\begin{smallmatrix} 1 & 0 \\ 1 & 1 \end{smallmatrix}\right)$. The receiver also knows which one of the $W_N^{(i)}$ channels are used for information bits and which $W_N^{(i)}$ channels are frozen.

Kronecker Power

n^{th} Kronecker power of F is calculated according to formula $F^{\otimes n} = F^{\otimes n-1} \otimes F$. Hence, with $F = \left(\begin{smallmatrix} 1 & 0 \\ 1 & 1 \end{smallmatrix}\right)$ for $n = 2$ one obtains

$$F^{\otimes 2} = \begin{pmatrix} F & 0 \\ F & F \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 \end{pmatrix}.$$

By using $F^{\otimes 2}$ we find

$$F^{\otimes 3} = \begin{pmatrix} F^{\otimes 2} & 0 \\ F^{\otimes 2} & F^{\otimes 2} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}.$$

Choosing the Information Bits

After channel combining and splitting operations, one obtains N channels with different capacities. To approach the Shannon capacity for large N, one must send data over the channels with high capacity, and send frozen bits over those with low capacity. To put it briefly, in order to design an (N, K) polar code for a given B-DMC, one first needs to calculate the capacities of N binary channels synthesized from that B-DMC, and order them according to their capacities. Then, the K channels with the highest capacities are used to send data, **i** $= [i_1, i_2, ..., i_K]$, and the remaining $N - K$ channels are used to send frozen bits, known both by the sender and the receiver. Generally these frozen bits are the 0 vector, **f** $= [0, 0, ..., 0]_{(N-K)}$.

Usually, the erasure probability ϵ of the original binary erasure channel W is unknown and set to 0.5. On the other hand, if a B-DMC W has known parameters (for instance if the erasure probability ϵ of a BEC is known), the ranking of the computed capacities may change. The corresponding code is called an adaptive polar code [Arıkan, 2008].

In order to explain the choice of the information and frozen bits of polar codes by an example, we consider an $(N, K) = (8, 4)$ polar code, which has 4 information bits and 4 frozen bits. We first calculate the capacity of each channel synthesized from a BEC(0.5) in a recursive manner starting with $N = 2$, and proceeding with $N = 4$ and 8. In Figure 2.12, the capacities of bit channels are given for a BEC(0.5). Secondly channels are sorted in descending order of capacities as

$[U_8, U_7, U_6, U_4, U_5, U_3, U_2, U_1]$. Hence data bits are sent from $[U_8, U_7, U_6, U_4]$ and the inputs corresponding to $[U_5, U_3, U_2, U_1]$ are frozen.

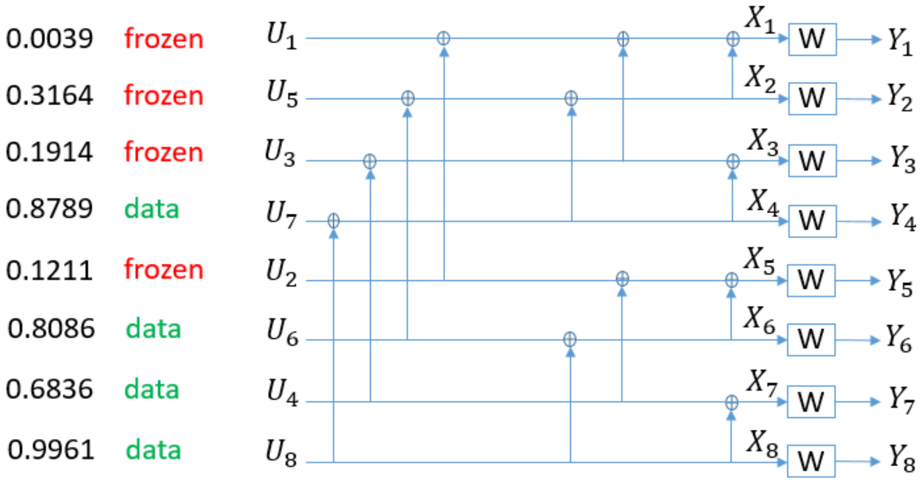


Figure 2.12: Channel capacities for $N = 8$ over a BEC(0.5).

In Figure 2.13, considering a specific input, information bits $[1, 0, 1, 0]$ are placed to $[U_4, U_6, U_7, U_8]$, and frozen bits $[0, 0, 0, 0]$ to $[U_5, U_3, U_2, U_1]$. The input of the factor graph becomes $[0, 0, 0, 1, 0, 0, 1, 0]$; N node values are found at each stage and finally, N values of the codeword are computed at the output of the $(\log_2 N)^{\text{th}}$ stage. So, the complexity of encoding is $N \log_2 N$.

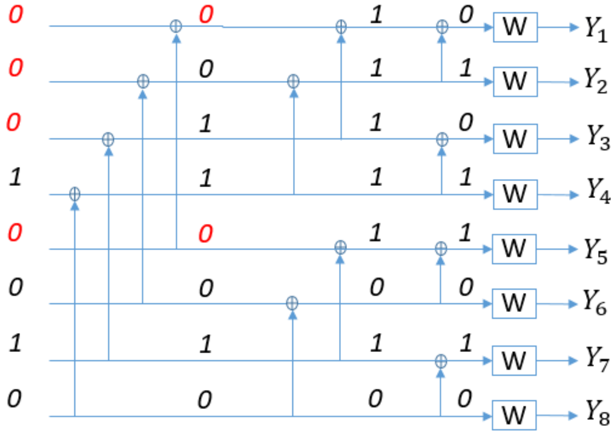


Figure 2.13: Encoding of (8,4) polar code with $[1, 0, 1, 0]$ data bits.

Interestingly, the (8,4) polar code designed for a BEC(0.5) is exactly the same as the (8,4) Reed-Muller (RM) code. Moreover, for $N = 8$, the choice of frozen bits remains the same for all erasure probabilities of the original channel W as can be observed in Table 2.1 that tabulates the capacities of $N = 8$ channels synthesized from a BEC(ϵ) for different values of ϵ. The equivalence between the polar and the RM codes is also observed for $N = 16$. Difference in generator matrices starts for higher blocklengths, such as the (32,16) codes, where the polar code replaces one of the weight-8 basis vectors of the RM code by a weight-4 basis vector [Arıkan, 2008].

Table 2.1: Capacities of 8 channels synthesized from a BEC(ϵ) for $N = 8$.

Synthesized channnel	ϵ=0.1	ϵ=0.2	ϵ=0.3	ϵ=0.4	ϵ=0.5	ϵ=0.6	ϵ=0.7	ϵ=0.8	ϵ=0.9
W^{---}	0.43	0.17	0.06	0.02	0.00	0.00	0.00	0.00	0.00
W^{+--}	0.96	0.85	0.69	0.50	0.32	0.17	0.07	0.02	0.00
W^{-+-}	0.93	0.76	0.55	0.35	0.19	0.09	0.03	0.01	0.00
W^{++-}	1.00	1.00	0.98	0.95	0.88	0.76	0.58	0.35	0.12
W^{--+}	0.88	0.65	0.42	0.24	0.12	0.05	0.02	0.00	0.00
W^{+-+}	1.00	0.99	0.97	0.91	0.81	0.65	0.45	0.24	0.07
W^{-++}	1.00	0.98	0.93	0.83	0.68	0.50	0.31	0.15	0.04
W^{+++}	1.00	1.00	1.00	1.00	1.00	0.98	0.94	0.83	0.57

2.4 Polar Code Decoding

There are several decoding methods for polar codes. ML decoding is the optimum but it has very high computational complexity for large block lengths. Suboptimum decoding methods are preferred in practice, whenever they have low complexities. Iterative structure of the polar encoding diagram is also suitable for the implementation of low complexity decoders. Arıkan proposes the successive cancellation (SC) decoding [Arıkan, 2009] for the polar codes.

Successive Cancellation Decoding

SC decoding is performed successively starting from u_1 to u_N. If u_i is frozen then there is nothing to compute. One fixes $\hat{u}_i$ to the frozen value, which is known by both the sender and the receiver. Otherwise if u_i is the information bit, one calculates the following log likelihood ratio:

$$L_N^{(i)}(y^N, \hat{u}^{i-1}) = \log \frac{W_N^{(i)}(y^N, \hat{u}^{i-1}|0)}{W_N^{(i)}(y^N, \hat{u}^{i-1}|1)} \tag{2.15}$$

If the log likelihood ratio (2.15) is larger than zero $\hat{u}_i$ is estimated as 0, otherwise it is estimated as 1.

$$\hat{u}_i = \begin{cases} 0 & \text{if } L_N^{(i)} \geq 0 \\ 1 & \text{otherwise} \end{cases} \tag{2.16}$$

As the encoding procedure, decoding is also recursive. To calculate L_N values, $L_{N/2}$ values are used. Recursion starts with L_1, and that can be computed directly. More detailed formulas for computing L_N log likelihood ratios are

$$\begin{gathered} L_N^{(2i-1)}(y^N, \hat{u}^{2i-2}) = \frac{1 + L_{N/2}^{(i)}(y^{N/2}, \hat{u}_o^{2i-2} \oplus \hat{u}_e^{2i-2}) L_{N/2}^{(i)}(y_{N/2+1}^{N/2}, \hat{u}_e^{2i-2})}{L_{N/2}^{(i)}(y^{N/2}, \hat{u}_o^{2i-2} \oplus \hat{u}_e^{2i-2}) + L_{N/2}^{(i)}(y_{N/2+1}^{N/2}, \hat{u}_e^{2i-2})} \\ L_N^{(2i)}(y^N, \hat{u}^{2i}) = L_{N/2}^{(i)}(y^{N/2}, \hat{u}_o^{2i-2} \oplus \hat{u}_e^{2i-2})^{1-2\hat{u}_{2i-1}} L_{N/2}^{(i)}(y_{N/2+1}^{N/2}, \hat{u}_e^{2i-2}) \end{gathered} . \tag{2.17}$$

Let $C(N)$ denote the complexity of the decoder for a length-N polar code. The N^{th} log likelihood ratio L_N is computed by using 2 $L_{N/2}$ log likelihood ratios with $O(N)$ computation complexity. Thus, the complexity of a length-N polar code is

$$C(N) = O(N) + 2C(N/2). \tag{2.18}$$

This equation implies $C(N)$ is of $O(N\text{log}N)$ complexity for successive cancellation decoding.

Belief Propagation Decoding

Polar codes can be seen as a generalization of Reed Muller (RM) codes [Hussami et al., 2009]. The BP decoding algorithm, also known as the sum product message passing algorithm [Pearl, 1988] can be used for RM codes by using the factor graphs proposed by Forney [Forney Jr, 2001], and it can be applied to polar codes as well.

Arıkan used the BP decoding to show the performance advantage of polar codes over the RM codes [Arıkan, 2009]. Korada indicated that the performance of the BP decoding for polar codes is halfway between the MAP and SC decoding algorithms [Korada, 2009]. As the SC decoding, the complexity of the BP decoding is also $O(N\log N)$.

Belief propagation is a message passing algorithm (MPA) that is performed on graphical representations. To explain by an example, we consider the parity check matrix of the (8,4) polar and RM codes given as

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

The corresponding Tanner graph of the H matrix is given in Figure 2.14. Tanner graph provides a representation of the H matrix and it is used for decoding. Tanner graph is a bi-partite graph, with two types of nodes; variable nodes, shown in Figure 2.14 as c_i and check nodes, shown as f_i.

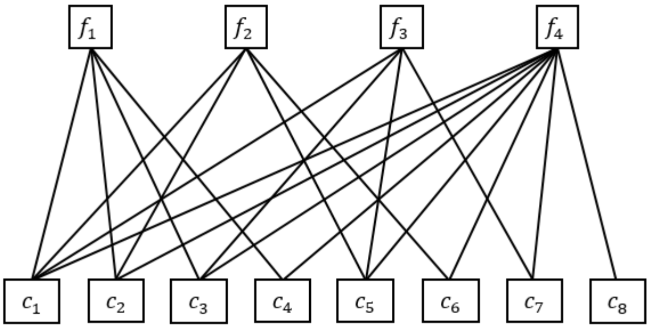


Figure 2.14: Tanner graph of the *H* matrix of the (8,4) polar and RM codes.

In the first step, all variable nodes send a message to the neighbor check nodes, The message is the belief of the variable node's being 1 or 0. In the second step, all check nodes send their response messages to each of their neighbor variable nodes. These messages are calculated by using the messages that are received from the neighbor variable nodes. In the third step, variable nodes calculate their messages that will be sent to check nodes by using the received messages from the neighbor check nodes.

Also all variable nodes update their estimated values by using the received messages. If the estimated word satisfies the parity check equation, $\mathbf{c}\mathrm{H}^{\mathrm{T}} = \mathbf{0}$, or the maximum number of iterations is performed, the algorithm is terminated; otherwise it goes back to second step.

For polar codes, there is a short cycle of size 4 (so the girth is 4) in this Tanner graph representation, corresponding to the path $c_1 \rightarrow f_1 \rightarrow c_2 \rightarrow f_2 \rightarrow c_1$. These short cycles are not desired since they degrade the decoding performance. Also high density parity check matrices with many 1's do not work for this algorithm. For f_4 in Figure 2.14, there are 8 neighbor variable nodes, so if more than one of them are incorrect, the decoding is prone to failure. Because of such reasons BP decoding of polar codes with this kind of representation is not convenient.

A suitable factor graph for BP decoding is the one proposed by Arıkan [Arıkan, 2009] and shown in Figure 2.15. This diagram is used for both encoding and decoding of polar codes.

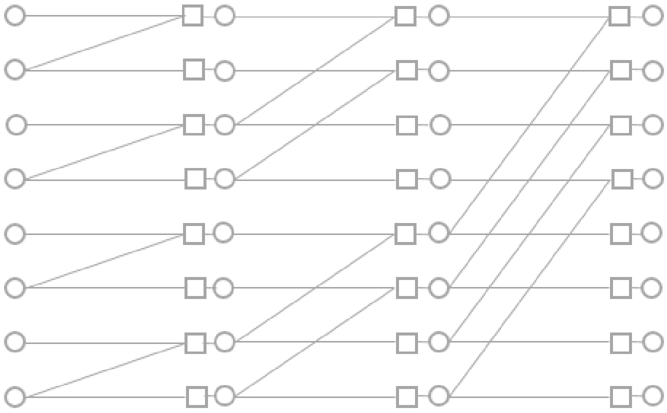


Figure 2.15: Factor graph representation of the *H* matrix of the (8,4) polar and RM codes.

With this representation one step message passing is divided into many stages. As explained above, messages are sent between neighbor nodes. With this representation, the number of variable nodes that are connected to each check node is decreased significantly. Every check node has 2 or 3 neighbor variable nodes. Another problem that is encountered for the Tanner graph representation is the size of the minimum cycle, whics is increased from 4 to 12 [Eslami and Pishro-Nik, 2010], by means of the factor graph with $\log_2 N$ stages.

More details about the belief propagation decoding algorithm that we implement is given in Chapter 3.

Successive Cancellation List Decoding

Successive cancellation list (SCL) decoding [Tal and Vardy, 2011] is a generalization of SC decoding that is proposed by Tal and Vardy. The main idea of list decoding is to keep L (list size) decoding paths at each stage and to select the most likely word from the list as output. As L increases, the performance improves but the complexity of this decoding becomes $O(LN\log N)$.

In Figure 2.16, the performance of the (2048,1024) polar code for each L value is given. Even for moderate L values, the performance is observed to approach to the ML bound.

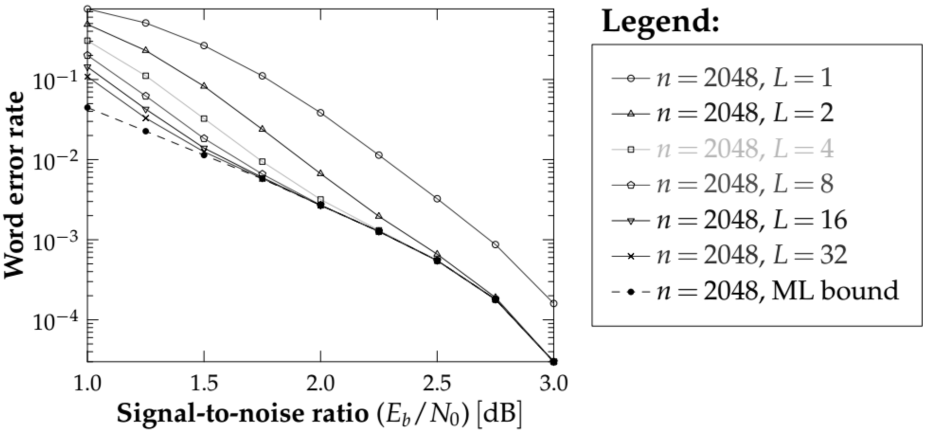


Figure 2.16: Word error rate of SCL decoding (reproduced from [Tal and Vardy, 2011]).

It is also possible to do better than this improvement with a small change on the polar code, while losing a small amount from rate. Tal and Vardy [Tal and Vardy, 2011] propose to add a genie while choosing the word from the list. For an (N, K) polar code there are K information bits. In the proposed method $K - R$ of them is used for data bits and R of them is used for the R-bit cyclic redundancy check (CRC) value of data bits. Before choosing the word from the list, if there is at least one path with a valid CRC, all other paths which have invalid CRC's are eliminated, and the choice is done on the remaining paths. If none of the paths has a correct CRC, most likely path is picked while it is known that at least one bit is incorrect.

In Figure 2.17 (or in Figure 1.1), which we reproduce from [Tal and Vardy, 2011], the performance comparison of the original polar code, different list decoding methods and WiMax LDPC code is given. As can be seen, the performance improves fascinatingly with a small loss in the rate caused by the CRC. The list size $L = 32$ and the CRC is 16 bits long in Figure 2.17.

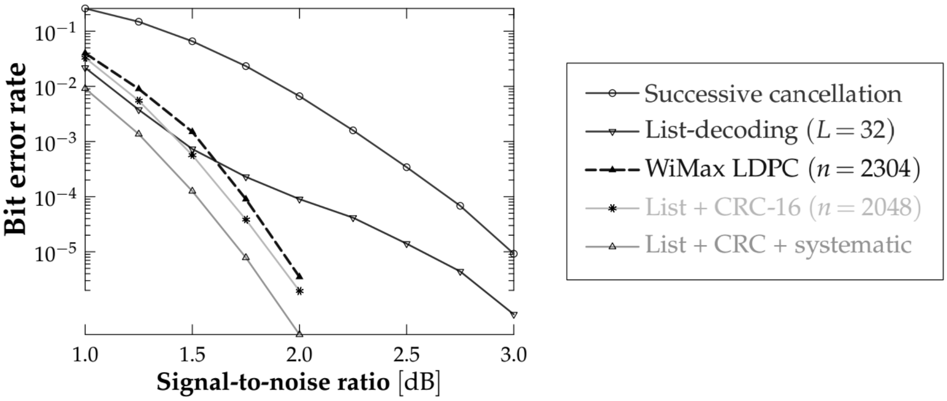


Figure 2.17: Performance comparison of decoding schemes of polar code and WiMax LDPC code (reproduced from [Tal and Vardy, 2011]).

CHAPTER 3

SIMULATION RESULTS

In this chapter, we present our simulation results related to the decoding of Reed-Muller, polar and adaptive polar codes by the belief propagation (BP) decoding algorithm. In Section 3.1, we describe the structure of the BP decoder that we implement, with emphasis on its discriminating feature of ‘to the leftmost with left/to the rightmost with right messages’. We discuss the choice of a decoding diagram that is also called a ‘factor graph’. In Section 3.2 we consider different factor graphs (diagrams) for the same code, together with the corresponding frozen variables. Section 3.3, discusses the choice of a proper value for the minimum sufficient number of iterations for the BP decoding of polar codes and Section 3.4 explains the correlation between bit channel capacities and the number of frozen variables. In Section 3.5, we compare the performance of the BP decoder for two extreme versions of the factor graphs. We then investigate in Section 3.6, how the performance of the BP decoding algorithm is affected when multiple factor graph decoding is utilized instead of a single factor graph, by simulating multiple factor graph decoding either with $\log_2 N$ cyclic diagrams as in [Korada, 2009], [Hussami et al., 2009], or with all possible $(\log_2 N)!$ diagrams. We also consider two choices of multiple factor graph sets: 1) with respect to the decreasing capacity sums (CS) of the information bit channels, 2) a genie-aided subset of the first set. In Section 3.7, we discuss the dependence of the performance on the number of frozen variables and capacity sums. Finally in Section 3.8, we present a brief comparison between the (128,64) RM and polar codes.

3.1 Description of the Decoding Structure Used in Simulations

In each simulation, after generating K random information bits and the corresponding codeword of size N for the given code, we pass the codeword through an erasure channel; i.e., erase each bit with a given erasure probability ϵ. K bits are generated randomly as 0 or 1. The N-bit codeword is the product of K bits and the generator matrix. The generated codeword is passed through a BEC(ϵ), by erasing or keeping each bit after comparing a random number uniformly distributed between 0 and 1 with the required erasure probability ϵ. If this random number is smaller than ϵ, the corresponding bit is erased.

We perform decoding on a diagram that consists of $n = \log_2 N$ cascaded stages as shown in Figure 3.1 for $n = 3, N = 8$. Each stage of the diagram has N input and N output variable nodes, where the output of the i^{th} stage is the input of the $(i+1)^{\text{th}}$ stage. Hence, the number of all variable nodes is equal to $(n+1)N$ and operations among variable nodes are performed at nN check nodes.

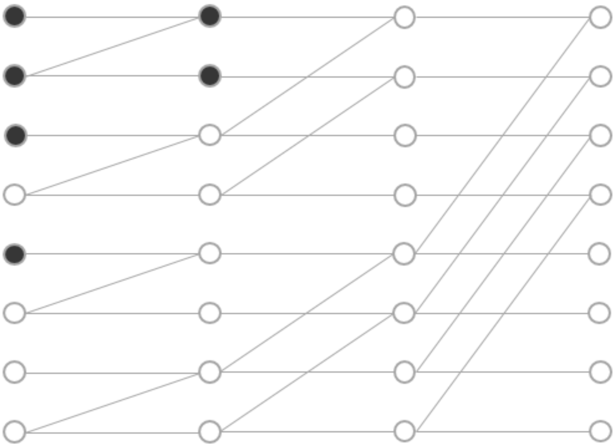


Figure 3.1: BP decoding diagram (or the factor graph) for an $N = 8$ code with $n = 3$ stages and $(n+1)N = 32$ variable nodes (where 6 filled nodes correspond to the frozen variables of the diagram, 2 of them are generated by the 4 input frozen bits of the (8,4) code).

Choosing a Diagram and Its Frozen Bits

For an (N, K) code with $N = 2^n$, there are $n!$ different valid diagrams. Each diagram that consists of *n* stages may have a different number of frozen variables, derived from the frozen bits at the input of the diagram.

After choosing a diagram corresponding to the given code, one should decide which of the input nodes are the information bits. The remaining bits are called frozen bits. Frozen bits of polar codes are chosen according to Arıkan's rule given by (3.1) and (3.2). The generator matrix $G_P(N, K)$ consists of rows chosen from $F^{\otimes n}$, where $F^{\otimes n}$ is the n^{th} Kronecker power of $F = \left(\begin{smallmatrix} 1 & 1 \\ 1 & 0 \end{smallmatrix}\right)$. The rows are chosen as follows:

(a) Calculate the $z_N = (z_{(N,1)}, z_{(N,2)}, ..., z_{(N,N)})$ vector according to:

$$z_{2k,j} = \begin{cases} 2z_{k,j} - z_{k,j}^2 & \text{if } 1 \leq j \leq k \\ z_{k,j-k}^2 & \text{if } k+1 \leq j \leq 2k \end{cases} \quad \text{where } z_{1,1} = 1/2 \tag{3.1}$$

(b) Form a permutation

$$\pi_N = (i_1, i_2, ..., i_N) \text{ where } z_{N,i_j} \leq z_{N,i_k} \ \forall j < k \tag{3.2}$$

(c) The rows of the generator matrix $G_P(N, K)$ are picked from the rows of $F^{\otimes n}$ with indices $i_1, i_2, ...i_K$. Remaining rows of $F^{\otimes n}$ correspond to the frozen bits.

3.1.1 Belief Propagation Decoding Algorithm

The belief propagation (BP) decoding algorithm that we have used in the simulations can be summarized by 6 steps. Xu, Che and Choi interpret the 3rd and 4th steps of the algorithm in the way that we do [Xu et al., 2015], and they claim that this interpretation differs from the common version of the algorithm in the 3rd and 4th steps. In our realization that is the same as the one in [Xu et al., 2015], each iteration starting from the output nodes, first proceeds to the leftmost of the factor graph only with left messages and then to the rightmost of the graph only with right messages, rather than simultaneously computing the left and right messages at each stage and then moving to the adjacent stage. The algorithm is composed of the following 6 steps:

1. Find the frozen variable nodes affected by frozen bits (the set of frozen variable nodes cover all frozen bits at the input and additional nodes at succeeding

stages). Assign 0 to all frozen variables and 0.5 to all remaining variable nodes of the diagram.

2. Set the output variable nodes of the last stage using 0's or 1's of the received word, and fill in 0.5 for erasures.

3. Shift messages (variable node values) starting from the rightmost of the diagram to the leftmost; i.e., from the output of the last stage to the input of the first stage,

4. Shift messages from left to right; i.e., from the input of the first stage to the output of the last stage.

5. If there is still an erasure at the output variable nodes, return to Step 3 until the received word is successfully reached at the rightmost variable set, or the maximum number of iterations are performed.

6. Stop.

Each step of the decoding algorithm can be further explained as follows:

Steps 1&2 - Frozen Variables & Setting the Received Word

A frozen variable is assigned to the value of 0 and remains as 0 till the end of the algorithm (observe the filled nodes in Figure 3.1). Figure 3.2 depicts a process unit (that is a detailed version of the z-shaped units in Figure 3.1 for $n = 1$, $N = 2$) of the BP algorithm, where variable nodes and check nodes are denoted by circles and rectangles respectively. The j^{th} and k^{th} output variables $v_O(i, j)$ and $v_O(i, k)$ of the process unit shown in Figure 3.2 can be frozen or not depending on the following four states of the j^{th} and k^{th} input variables $v_I(i, j)$ and $v_I(i, k)$.

(a) If $v_I(i, j)$ and $v_I(i, k)$ are frozen, then $v_O(i, j)$ and $v_O(i, k)$ are frozen.

(b) If only $v_I(i, k)$ is frozen, then only $v_O(i, k)$ is frozen.

(c) If only $v_I(i, j)$ is frozen, then none of $v_O(i, j)$ and $v_O(i, k)$ is frozen.

(d) If none of $v_I(i,j)$ and $v_I(i,k)$ is frozen, then none of $v_O(i,j)$ and $v_O(i,k)$ is frozen.

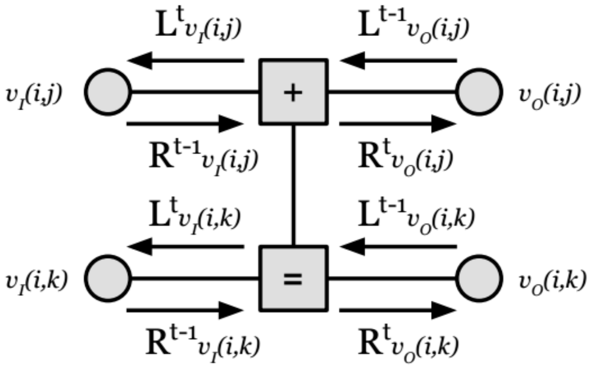


Figure 3.2: BP process unit (subscripts I and O indicate the input and output, i shows the stage, j and k show the rows of the diagram, L and R are the left and right messages, t is the iteration number).

0's and 1's of the received word are set to the rightmost variable nodes, with 1/2's for erasures.

Steps 3&4 – Shifting Messages from Right to Left & from Left to Right

Calling $p_x(0)$ the probability that *x* equals 0, and $p_x(1)$, the probability that *x* equals 1; in the description of operations going towards the left side of the diagram, $p_x(0) = L_v(0)$ and $p_x(1) = L_v(1)$; and the operations proceeding towards the right side of the diagram use the notation $p_x(0) = R_v(0)$ and $p_x(1) = R_v(1)$. Naturally, all these probabilities satisfy $p_x(0) + p_x(1) = 1$, $L_v(0) + L_v(1) = 1$ and $R_v(0) + R_v(1) = 1$.

Two operations $\otimes$ and $\odot$ between two probabilities are defined as:

$$\begin{aligned}(p_x \otimes p_y)(0) &= p_x(0)p_y(0) + p_x(1)p_y(1)\\ (p_x \otimes p_y)(1) &= p_x(0)p_y(1) + p_x(1)p_y(0)\\ (p_x \odot p_y)(0) &= p_x(0)p_y(0)\\ (p_x \odot p_y)(1) &= p_x(1)p_y(1)\end{aligned} \tag{3.3}$$

Denoting the present and past values of the variable nodes by the superscripts t and

$t-1$ respectively, operations on the BP process unit in Figure 3.2 towards left are defined by

$$\begin{aligned} L^t_{v_I(i,j)} &= L^{t-1}_{v_O(i,j)} \otimes [L^{t-1}_{v_O(i,k)} \odot R^{t-1}_{v_I(i,k)}] \\ L^t_{v_I(i,k)} &= [R^{t-1}_{v_I(i,j)} \otimes L^{t-1}_{v_O(i,j)}] \odot L^{t-1}_{v_O(i,k)} \end{aligned} \tag{3.4}$$

and those towards right are defined by:

$$\begin{aligned} R^t_{v_O(i,j)} &= R^{t-1}_{v_I(i,j)} \otimes [L^{t-1}_{v_O(i,k)} \odot R^{t-1}_{v_I(i,k)}] \\ R^t_{v_O(i,k)} &= [R^{t-1}_{v_I(i,j)} \otimes L^{t-1}_{v_O(i,j)}] \odot R^{t-1}_{v_I(i,k)} \end{aligned} \tag{3.5}$$

As the decoding starts, the received word is assigned to $v_O(n,j), j = 1, ..., N$, shown in Figure 3.3 (sketched for $N = 8$ and $n = 3$). Then left messages sent from $v_O(n,j)$ to check nodes are calculated. For a BEC, the received bit, $rw(i)$, is either 0, 1 or an erased bit E. Corresponding initial probabilities are taken as:

$$L^0_{v_O(n,j)} = \begin{cases} 0 & \text{if } rw(i) = 1 \\ 1 & \text{if } rw(i) = 0 \\ 0.5 & \text{if } rw(i) = E \end{cases} \tag{3.6}$$

Iterations start by going from the rightmost to the leftmost variables and using (3.4), then proceed by going from the leftmost to the rightmost variables and using (3.5). In the first iteration there are plenty of unknown variable messages, $L^0_{v_*(i,j)}$ and $R^0_{v_*(i,j)}$, and they are all taken as 0.5.

Frozen variables in the factor graph both speed up and fasten belief propagation decoding. While the number of frozen variables increases, the number of the decoding complexity decreases; because there are less sum and product operations while calculating the frozen variables.

The following equations (3.7), (3.8) and (3.9) can be used according to if $v_I(i,j)$ or $v_I(i,k)$ variables, that are given in Figure 3.2, are frozen.

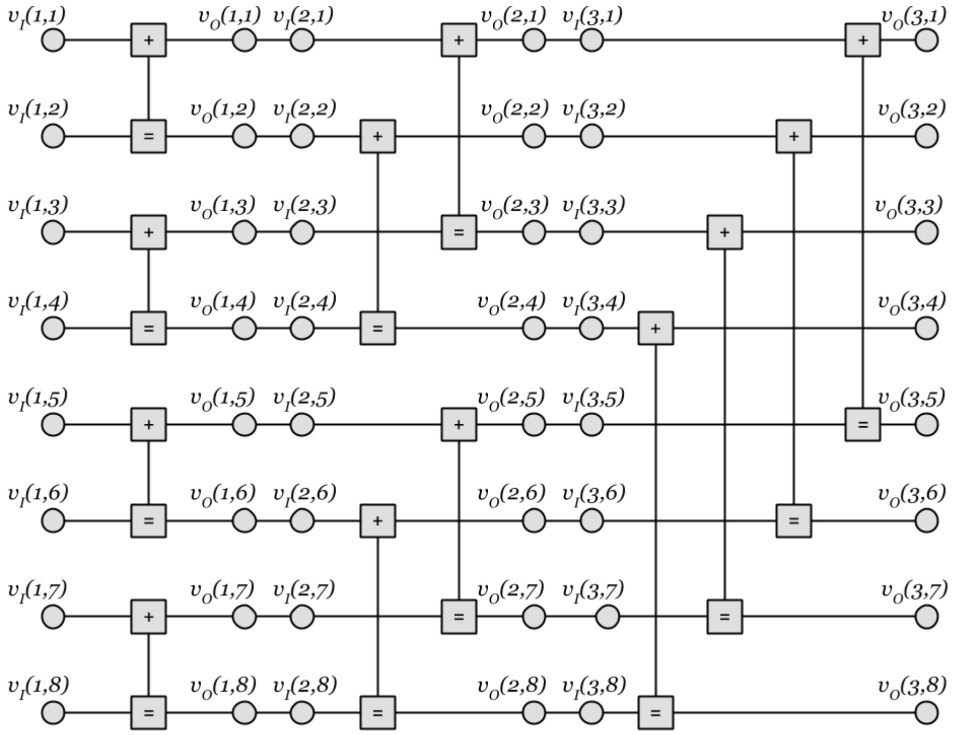

$v_I(1,1)$ $v_O(1,1)$ $v_I(2,1)$ $v_O(2,1)$ $v_I(3,1)$ $v_O(3,1)$
$v_I(1,2)$ $v_O(1,2)$ $v_I(2,2)$ $v_O(2,2)$ $v_I(3,2)$ $v_O(3,2)$
$v_I(1,3)$ $v_O(1,3)$ $v_I(2,3)$ $v_O(2,3)$ $v_I(3,3)$ $v_O(3,3)$
$v_I(1,4)$ $v_O(1,4)$ $v_I(2,4)$ $v_O(2,4)$ $v_I(3,4)$ $v_O(3,4)$
$v_I(1,5)$ $v_O(1,5)$ $v_I(2,5)$ $v_O(2,5)$ $v_I(3,5)$ $v_O(3,5)$
$v_I(1,6)$ $v_O(1,6)$ $v_I(2,6)$ $v_O(2,6)$ $v_I(3,6)$ $v_O(3,6)$
$v_I(1,7)$ $v_O(1,7)$ $v_I(2,7)$ $v_O(2,7)$ $v_I(3,7)$ $v_O(3,7)$
$v_I(1,8)$ $v_O(1,8)$ $v_I(2,8)$ $v_O(2,8)$ $v_I(3,8)$ $v_O(3,8)$
+
=

Figure 3.3: BP decoding diagram that shows the input and output variables of each process unit in detail, for an $N = 8$ code with $n = 3$ stages.

If $v_I(i, j)$ and $v_I(i, k)$ are both frozen:

$$\begin{gathered} L^t_{v_I(i,j)}(1) = 0 \text{ and } L^t_{v_I(i,j)}(0) = 1 \\ L^t_{v_I(i,k)}(1) = 0 \text{ and } L^t_{v_I(i,k)}(0) = 1 \\ R^t_{v_O(i,j)}(1) = 0 \text{ and } R^t_{v_O(i,j)}(0) = 1 \\ R^t_{v_O(i,k)}(1) = 0 \text{ and } R^t_{v_O(i,k)}(0) = 1 \end{gathered} \tag{3.7}$$

If only $v_I(i + 1, j)$ is frozen:

$$\begin{gathered} L^t_{v_I(i,j)} = L^{t-1}_{v_O(i,j)} \\ L^t_{v_I(i,k)}(1) = 0 \text{ and } L^t_{v_I(i,k)}(0) = 1 \\ R^t_{v_O(i,j)} = R^{t-1}_{v_I(i,j)} \\ R^t_{v_O(i,k)}(1) = 0 \text{ and } R^t_{v_O(i,k)}(0) = 1 \end{gathered} \tag{3.8}$$

If only $v_I(i,j)$ is frozen:

$$\begin{aligned} L^t_{v_I(i,j)}(1) = 0 \text{ and } L^t_{v_I(i,j)}(0) = 1 \\ L^t_{v_I(i,k)} &= L^{t-1}_{v_O(i,j)} L^{t-1}_{v_O(i,k)} \\ R^t_{v_O(i,j)} &= L^{t-1}_{v_O(i,k)} R^{t-1}_{v_I(i,k)} \\ R^t_{v_O(i,k)} &= L^{t-1}_{v_O(i,j)} R^{t-1}_{v_I(i,k)} \end{aligned} \tag{3.9}$$

We will discuss in Section 3.7 that the number of frozen bits may have a significant effect also on the performance.

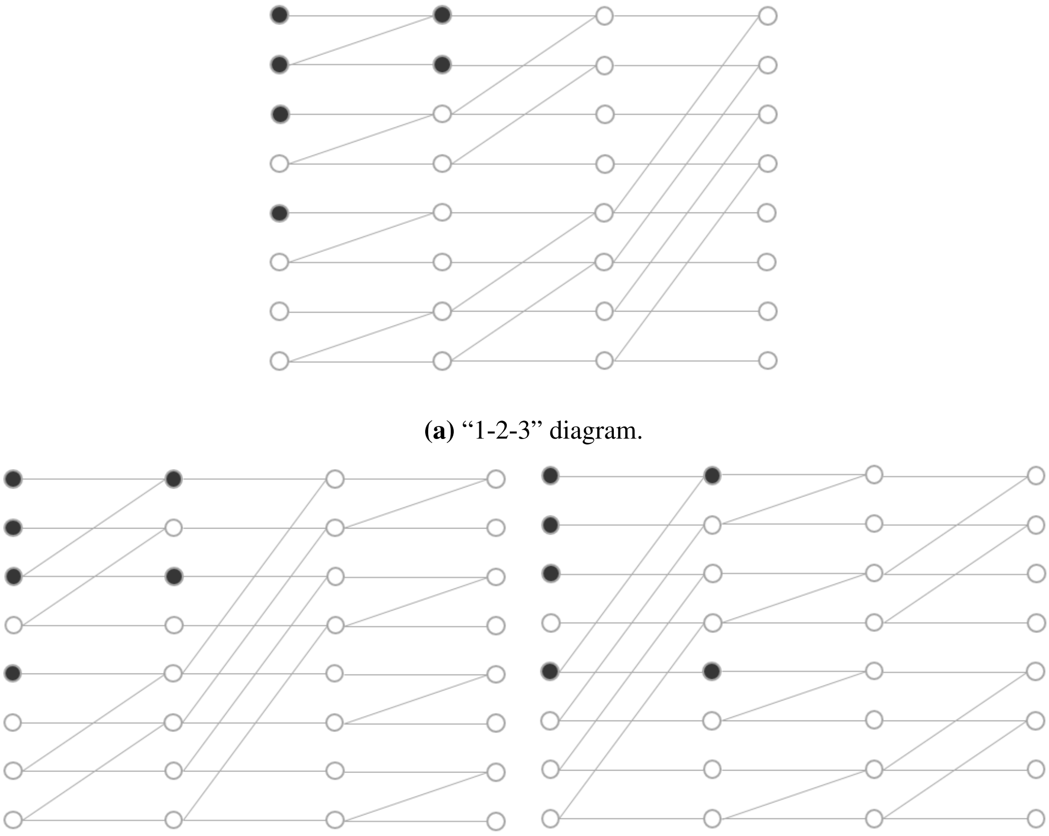
After each round iteration, which is composed of steps 3&4 in the BP decoding algorithm, with our interpretation that to the leftmost with only left messages and to the rightmost with only right messages, one arrives at $v_O(n,j)$ variables for $j = 1, \ldots, N$. These N variables form the decoded word at each iteration and it is checked to find out if there remains any erasures. If all erasures are filled with 0's and 1's and one obtains a codeword, decoding is finished. Sometimes a codeword cannot be obtained until the given maximum number of iterations are performed.

Simulations for determining the minimum sufficient number of round iterations are discussed in Section 3.3.

3.2 Decoder Structure and the Number of Frozen Variables for the BP Decoding Algorithm

BP decoding algorithm for an (N, K) code can be implemented by using the BP process unit depicted in Figure 3.2 in different decoder structures. There are $n = \log_2 N$ stages, and by changing the order of stages, one can form $n!$ different diagrams.

In Figure 3.4, we depict 3 of the $3! = 6$ possible factor graphs for a (8,4) polar code, and also indicate the corresponding frozen variables. Noting that Stage 1 in part (a) has adders for adjacent bits and Stage 3 has adders for the most distant bits, we call the corresponding regular decoder structure “1-2-3”; and name the permuted graphs in parts (b) and (c) as “2-3-1” and “3-1-2” respectively.



(a) "1-2-3" diagram.

(b) "2-3-1" diagram.

(c) "3-1-2" diagram.

Figure 3.4: Different decoding diagrams for the (8,4) polar code.

The frozen bits at the input are the same for all permutations; however, the frozen variables that they affect within the diagram are different as shown by filled dots in Figure 3.4. Since the frozen bits at the input stage are kept fixed, the total capacity of N channels differs from one diagram to another, and the decoder performance varies as will be discussed in Section 3.5.

In order to demonstrate the position and the number of the frozen variables more efficiently for larger values of N, we generate Table 3.1 that shows all initial node values of the factor graphs given in Figure 3.4 (and additionally, those of the remaining 3 graphs), where 0's indicate the frozen variables and 1's are used for the unfrozen bits. Each diagram is displayed by four columns, where the first column indicates the input and the fourth column shows the output variables. It can be seen in Figure 3.4 and Table 3.1 that the number of frozen variables is 6 and it remains the same for all 6 graphs of the (8,4) codes. However, as N increases, the number of frozen variables does not remain constant over possible factor graphs.

Table 3.1: An alternative way of displaying the frozen bits (0's in the table) corresponding to all factor graphs of the (8,4) polar code.

1-2-3					2-3-1					3-1-2			
0	0	1	1		0	0	1	1		0	0	1	1
0	0	1	1		0	1	1	1		0	1	1	1
0	1	1	1		0	0	1	1		0	1	1	1
1	1	1	1		1	1	1	1		1	1	1	1
0	1	1	1		0	1	1	1		0	0	1	1
1	1	1	1		1	1	1	1		1	1	1	1
1	1	1	1		1	1	1	1		1	1	1	1
1	1	1	1		1	1	1	1		1	1	1	1
1-3-2					2-1-3					3-2-1			
0	0	1	1		0	0	1	1		0	0	1	1
0	0	1	1		0	1	1	1		0	1	1	1
0	1	1	1		0	0	1	1		0	1	1	1
1	1	1	1		1	1	1	1		1	1	1	1
0	1	1	1		0	1	1	1		0	0	1	1
1	1	1	1		1	1	1	1		1	1	1	1
1	1	1	1		1	1	1	1		1	1	1	1
1	1	1	1		1	1	1	1		1	1	1	1

For instance, considering the (32,16) polar code, there are $n = 5$ stages, so one can form $5! = 120$ different diagrams. Using the same definition for stages (i.e.; Stage 1 has adders for adjacent nodes, Stage 5 has adders for the most distant nodes), we have arbitrarily chosen the reference diagram as "5-4-3-2-1", and adjusted the input frozen bits with respect to this reference diagram. In accordance with Korada's multiple factor graph decoding idea [Hussami et al., 2009], we have tried every permutation and found the number of frozen variables for each of the 120 diagrams, by keeping the 16 frozen bits at the input fixed. In Figure 3.5 we plot the distribution of the 120 diagrams for the (32,16) polar code versus the number of frozen variables; i.e., the number of diagrams corresponding to each possible number of frozen variables, which is found to vary between 24 and 44 in steps of 4.

Figure 3.5 shows that there are 12 diagrams of the (32, 16) code with 44 frozen variables. The stage orders corresponding to these 12 diagrams are given in Table 3.2. In Section 3.4, it is stated that all of these 12 diagrams, with the fixed input frozen bits chosen according to the "5-4-3-2-1" reference diagram, have the information bit channels with the 16 highest capacities; therefore, all 12 these diagrams correspond

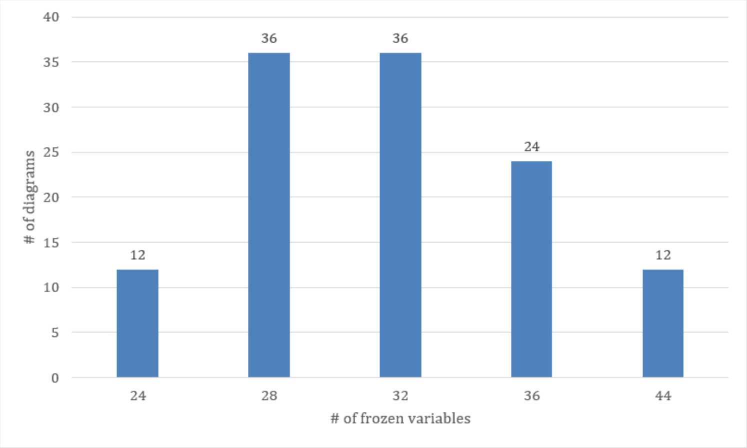


Figure 3.5: Distribution of 120 diagrams corresponding to the given number of frozen variables for the (32,16) polar code.

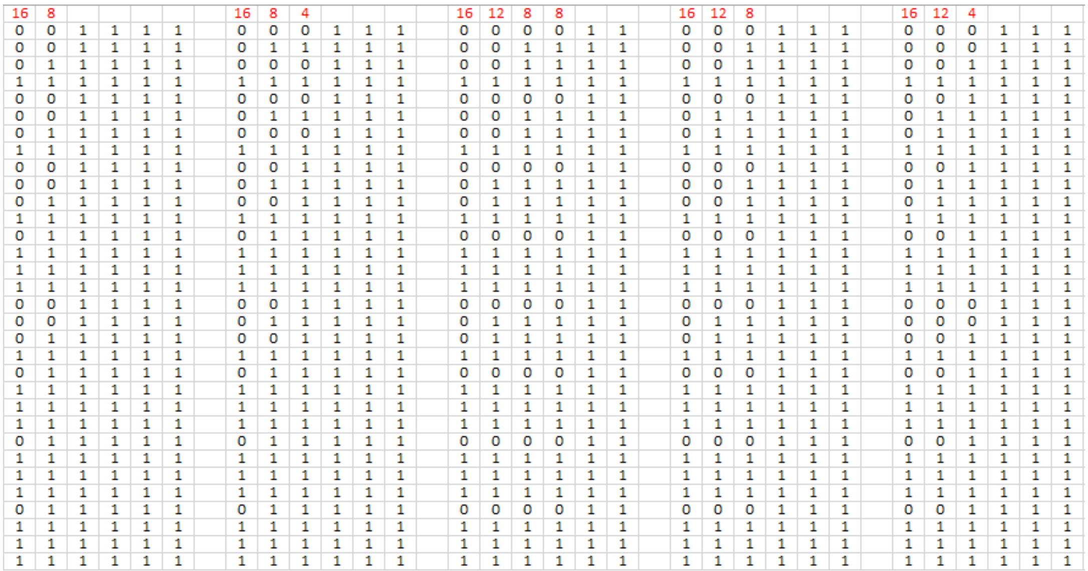
to (32, 16) polar codes with properly chosen frozen bits.

Table 3.2: For the (32,16) polar code, stage orders of the 12 diagrams having 44 frozen variables, whose input frozen bits are adjusted according to the "5-4-3-2-1" diagram.

Diagram	**Stage order**				
1	3	4	5	2	1
2	3	4	5	1	2
3	3	5	4	1	2
4	3	5	4	2	1
5	4	3	5	1	2
6	4	3	5	2	1
7	4	5	3	1	2
8	4	5	3	2	1
9	5	3	4	2	1
10	5	3	4	1	2
11	5	4	3	2	1
12	5	4	3	1	2

In Figure 3.6, we show the initial node values for the 5 cyclic permutations of 5 stages. Zeros are the frozen variables and ones are the information variables. First diagram's stage order is 1-2-3-4-5, and the successive ones are its cyclic rotations. The numbers in the first row show the number of frozen variables for each stage. It can be seen that the third diagram contains the maximum number of 44 frozen variables $(16+12+8+8)$. The corresponding stage order "3-4-5-1-2" is the second one among the 12 diagrams given in Table 3.2.

As will be observed in Section 3.7, a diagram usually has better performance than the



16	8					16	8	4				16	12	8	8			16	12	8				16	12	4			
0	0	1	1	1	1	0	0	0	1	1	1	0	0	0	0	1	1	0	0	0	1	1	1	0	0	0	1	1	1
0	0	1	1	1	1	0	1	1	1	1	1	0	0	1	1	1	1	0	0	1	1	1	1	0	0	0	1	1	1
0	1	1	1	1	1	0	0	0	1	1	1	0	0	1	1	1	1	0	0	1	1	1	1	0	0	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0	0	1	1	1	1	0	0	0	1	1	1	0	0	0	0	1	1	0	0	0	1	1	1	0	0	1	1	1	1
0	0	1	1	1	1	0	1	1	1	1	1	0	0	1	1	1	1	0	1	1	1	1	1	0	1	1	1	1	1
0	1	1	1	1	1	0	0	0	1	1	1	0	0	1	1	1	1	0	1	1	1	1	1	0	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0	0	1	1	1	1	0	0	1	1	1	1	0	0	0	0	1	1	0	0	0	1	1	1	0	0	1	1	1	1
0	0	1	1	1	1	0	1	1	1	1	1	0	1	1	1	1	1	0	0	1	1	1	1	0	1	1	1	1	1
0	1	1	1	1	1	0	0	1	1	1	1	0	1	1	1	1	1	0	0	1	1	1	1	0	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0	1	1	1	1	1	0	1	1	1	1	1	0	0	0	0	1	1	0	0	0	1	1	1	0	0	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0	0	1	1	1	1	0	0	1	1	1	1	0	0	0	0	1	1	0	0	0	1	1	1	0	0	0	1	1	1
0	0	1	1	1	1	0	1	1	1	1	1	0	1	1	1	1	1	0	1	1	1	1	1	0	0	0	1	1	1
0	1	1	1	1	1	0	0	1	1	1	1	0	1	1	1	1	1	0	1	1	1	1	1	0	0	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0	1	1	1	1	1	0	1	1	1	1	1	0	0	0	0	1	1	0	0	0	1	1	1	0	0	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0	1	1	1	1	1	0	1	1	1	1	1	0	0	0	0	1	1	0	0	0	1	1	1	0	0	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0	1	1	1	1	1	0	1	1	1	1	1	0	0	0	0	1	1	0	0	0	1	1	1	0	0	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Figure 3.6: Frozen variables (shown by 0's) of the 5 cyclic diagrams for the (32,16) polar code, corresponding to stage permutations 1-2-3-4-5, 2-3-4-5-1, 3-4-5-1-2, 4-5-1-2-3 and 5-1-2-3-4 respectively.

remaining diagrams with smaller number of frozen variables. However, before measuring the performance by simulations, one should decide on the optimum number of iterations required for BP decoding; which is the question that we try to answer in Section 3.3.

3.3 Choice of the Number of Iterations for the BP Decoding Algorithm

Considering the 3rd and the 4th steps of the BP decoding algorithm given in Section 3.1.1 as a single round-iteration, we search for the optimum number of iterations needed for BP decoding over a BEC(ϵ) with erasure probability ϵ. The number of required iterations for a given code depends on the other parameters of the simulation, and in order to determine the minimum sufficient number of iterations, we perform an experiment as follows: For each given erasure probability up to $\epsilon = 0.5$, we transmit 1,000 codewords for rate $1/2$ codes, where $N = 32, 128$ and 512, using factor graphs similar to Figure 3.1; i.e., having the stage order 1-...-5, which is chosen because it contains the minimum number of frozen variables (24 for the (32,16) code, as shown in the first 6 columns of Figure 3.6). We then repeat the experiment for the reference stage order 5-...-1 that has the maximum number of 44 frozen variables for the (32,16) code.

Table 3.3 shows the number of decoded codewords versus the iteration number for the (32,16) code, having the stage order 1-...-5; where the last row is the number of undecoded words after the decoding algorithm has reached the maximum number of round iterations. Table 3.4 presents similar information, for the stage order 5-...-1.

Table 3.3: Number of decoded codewords out of 1,000 received words of the (32,16) polar code after each round iteration for the stage order 1-...-5 over a BEC(ϵ).

ϵ Iter. number	0.05	0.15	0.25	0.35	0.45
1	901	688	247	43	4
2	99	283	489	308	84
3	0	19	136	196	113
4	0	2	30	77	63
5	0	0	6	16	24
6	0	0	1	5	7
7	0	0	0	0	0
8	0	0	0	0	1
9	0	0	0	0	0
10	0	0	0	0	0
.	.	.	.	.	.
.	.	.	.	.	.
.	.	.	.	.	.
30	0	0	0	0	0
Number of undecoded words	0	8	91	355	703

From Table 3.3 one observes that all erasures of the 901 received words are corrected at the first iteration for an $\epsilon = 0.05$ and remaining 99 words are corrected at the second iteration. So, there remains no word that is not corrected. If $\epsilon = 0.15$, 688 codewords are decoded in the first iteration, 283 in the second iteration, 19 in the third iteration and 2 in the fourth iteration. No received word is corrected after the fourth iteration and there remains 8 undecoded words after 30 round iterations. That means iterations after the fourth one are useless for $\epsilon = 0.15$.

For the (32,16) polar code, and erasure probabilities up to 0.45, we observe that the maximum value of iterations at which a codeword can be decoded by the diagram with stage order 1-2-3-4-5 is 8. So, at all considered ϵ's, setting the maximum number of round iterations to 10 seems to be sufficient.

On the other hand, if the same experiment is repeated for the diagram with the reference stage order 5-...-1, we obtain the iteration numbers given in Table 3.4, which are definitely smaller than those given in Table 3.3. All erasures of the 988 received words are corrected at the first iteration for $\epsilon = 0.05$ and remaining 12 words are corrected at the second iteration. So, there remains no word that is not corrected. If $\epsilon = 0.15$, 915 received words are decoded in the first iteration, and 82 received words at the second iteration. No received word is corrected after the second iteration and there remains 3 undecoded words after 30 round iterations. It is clear that with the stage order 5-...-1, the received words are decoded at earlier iterations than the stage order 1-...-5. Similar observations are valid for other ϵ values ($\epsilon = 0.25, 0.35, 0.45$). Hence, one can say that the BP decoder of (32,16) code with the stage order 5-...-1, needs smaller number of iterations than the stage order 1-...-5; and it is also more successful in terms of the number of remaining undecoded words (last rows of Table 3.3 and 3.4).

Table 3.4: Number of decoded codewords out of 1,000 received words of the (32,16) polar code after each round iteration for the stage order *n*-...-1 over a BEC(ϵ).

ϵ / Iter. number	0.05	0.15	0.25	0.35	0.45
1	988	915	637	276	63
2	12	82	305	432	241
3	0	0	17	69	120
4	0	0	2	9	19
5	0	0	0	2	8
6	0	0	0	0	0
7	0	0	0	0	0
8	0	0	0	0	0
9	0	0	0	0	0
10	0	0	0	0	0
.	.	.	.	.	.
.	.	.	.	.	.
.	.	.	.	.	.
30	0	0	0	0	0
Number of undecoded words	0	3	39	212	549

In Appendix A, Tables A.1 and A.2 give similar information for the (128, 64) codes, and Tables A.3 and A.4 demonstrate the required number of iterations for the (512,256) codes. As in the case of the (32,16) code, the stage order 1-...-n in Table A.1 and A.3 needs larger number of iterations and performs worse than the reference stage order *n*-...-1 in Table A.2 and A.4.

To conclude this section, we collect the required number of BP iterations for the mentioned three codes ($N = 32, 128, 512$) and two stage orders, 1-...-*n* and *n*-...-1, in Table 3.5. Blank entry for the (512,256) polar code and $\epsilon = 0.45$ shows that there is no word decoded by the 1-...-*n* diagram at any iteration. We can say that for the reference stage order *n*-...-1, it is sufficient to use 5, 10 and 15 round iterations for the BP decoding of the (32,16), (128,64) and (512,256) polar codes, respectively.

Table 3.5: Number of round iterations needed for BP decoding of rate 1/2 polar codes over a BEC(ϵ).

Stage order of the diagram	Erasure probability ϵ	(32, 16) polar code	(128, 64) polar code	(512, 256) polar code
1-...-*n*	0.25	6	10	15
n-...-1		4	5	5
1-...-*n*	0.35	6	13	20
n-...-1		5	8	9
1-...-*n*	0.45	8	12	-
n-...-1		5	9	15

3.4 Dependence of Bit Channel Capacities on the Decoder Structure

As mentioned in Chapter 2, every bit channel has its own capacity and this depends on how it is connected to the original channel. In this section, we give a small example for a polar code with $N = 4$. In Figure 3.7, the erasure probability of the original binary erasure channels is $\epsilon = 0.5$, so the channel capacity is $1 - 0.5 = 0.5$. By calculating the capacity of each channel stage by stage, the capacities of bit channels are computed as in Figure 3.7.

Now, if the stage order is changed as in Figure 3.8, the capacities on every stage change and the computed capacities of bit channels are as shown in Figure 3.8.

One can observe that the bit channel capacities from the input bits to the N output bits are different in Figures 3.7 and 3.8, where the 2$^{\text{nd}}$ and the 3$^{\text{rd}}$ bit channels' capacities

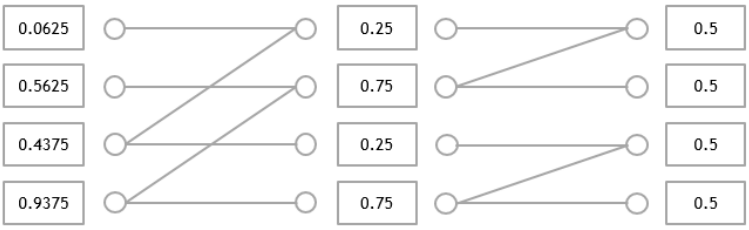


Figure 3.7: Capacities of 2-1 stage order for a BEC(0.5).

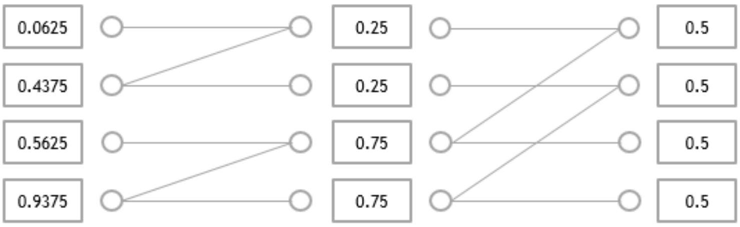


Figure 3.8: Capacities of 1-2 stage order for a BEC(0.5).

are switched. If one chooses the 2^{nd} and the 4^{th} inputs to send the data bits, according to Figure 3.7 the capacity sum (CS) of the used channels is $0.5625 + 0.9375 = 1.5$ and according to Figure 3.8, CS$= 0.4375 + 0.9375 = 1.375$. Hence, the sum of the transmitted bit channel capacities is changed according to the chosen stage order. In other words, to freeze the first and the second input bits is rational for the 2-1 diagram in Figure 3.7, but it is irrational for the 1-2 diagram in Figure 3.8. Since to freeze a high-capacity channel is against the polar coding philosophy of transmission over high-capacity bit channels, one can not say that the "1-2 diagram with frozen 1^{st} and 3^{th} bits" is a suitably chosen graph for a polar code. On the other hand, freezing the 1^{st} and the 2^{nd} bits of the 1-2 diagram in Figure 3.8 would make it completely identical to the polar code given in Figure 3.7; i.e., to the 2-1 diagram with frozen 1^{st} and 3^{th} bits.

[Korada, 2009] and [Hussami et al., 2009] consider multiple factor graph decoding with $\log_2 N$ cyclic diagrams and report an improvement over single factor graph decoding. By the reasoning given above, keeping the input frozen bits (adjusted for a specific diagram) fixed for all diagrams is expected to weaken the performance on the average. However, multiple factor graph decoding chooses the best path among many decoders; therefore, it promises an improvement at the expense of increased complexity. Our aim is to investigate whether it is feasible to increase the number of factor graphs to $(\log_2 N)!$ diagrams instead of the $\log_2 N$ cyclically rotated diagrams used in [Korada, 2009] and [Hussami et al., 2009].

Before presenting our simulation results with $(\log_2 N)!$ factor graphs in Section 3.6, some empirical evidence on the correlation of the capacity sums of the transmission bit channels and the number of frozen variables may be useful.

As an example, for the (32,16) code, there are $5! = 120$ different diagrams. These 120 diagrams all having the same 16 frozen bits (chosen with respect to the 5-4-3-2-1 diagram) can be gathered into 10 sets, each containing 12 diagrams with equal CS (capacity sum) values found over a BEC(0.3125); as tabulated in Table 3.6. For each number of frozen variables, FV, the last row of Table 3.6 indicates the average of the corresponding CS values, which is observed to decrease as FV decreases.

Table 3.6: Equal capacity sets of the (32,16) polar codes over a BEC(0.3125).

Set	Capacity Sum (CS)	Number of diagrams in the set with the number of frozen variables (FV) equal to				
		44	36	32	28	24
1	15.82	12				
2	15.68		12			
3	15.57			12		
4	15.50		12			
5	15.46				12	
6	15.36			12		
7	15.23				12	
8	15.16			12		
9	15.05				12	
10	14.94					12
Average Capacity for fixed FV		15.82	15.59	15.36	15.25	14.94

In another example, we observe similar positive correlation between the CS and FV values of the (64,32) code (see Appendix B) over a BEC(0.34375), as emphasized in Section 3.7. There is empirical as well as theoretical evidence indicating that both CS and FV values of a factor graph are the main factors that affect the performance of the BP decoder.

3.5 Performance Dependence of the BP Decoding Algorithm on the Decoder Structure - Stage Order "1-...-*n*" versus "*n*-...-1"

In this section, our aim is to find the performance difference between two extreme kinds of factor graphs, namely those with stage orders 1-2-...-*n* and *n*-...-2-1. As the measure of performance, we compute the remaining bit erasure probabilities after decoding.

Decodings with both 1-2-...-*n* and *n*-...-2-1 stage orders are simulated under the same conditions. First we generate a codeword and pass that codeword through a given BEC(ϵ). For both decodings, the codewords and erased bits are the same. Then the BP algorithm tries to decode the erased word.

We compute the bit erasure probabilities for a (32,16) polar code over 10,000 trials (erased codewords) using 10 round iterations at most, that are given in Table 3.7. The 5-...-2-1 diagram has 44 frozen variables, while the 1-2-...-5 diagram has 24 frozen variables. The capacity sum of the 16 unfrozen channels is equal to 15.82 for the 5-...-2-1 and 14.94 for the 1-2-...-5 diagrams for a BEC(0.3125).

Table 3.7: Decoded bit erasure probabilities of a (32,16) codes on a BEC(ϵ) over 10,000 trials and up to 10 iterations of the BP algorithm.

Code Type	$\epsilon = 0.05$	$\epsilon = 0.15$	$\epsilon = 0.25$	$\epsilon = 0.35$	$\epsilon = 0.45$
Polar (1-2-...-5)	0.00000	0.00217	0.02690	0.12323	0.30915
Polar (5-...-2-1)	0.00000	0.00069	0.00798	0.05584	0.19930

Table 3.8 gives the bit erasure probability performance of the (128,64) polar code over 2,000 trials using up to 15 round iterations at the BP decoder. The 7-...-2-1 diagram has 274 frozen variables while 1-2-...-7 diagram has 162 frozen variables. The capacity sum of the 64 unfrozen channels is equal to 61.66 for the 7-...-2-1 and 51.58 for the 1-2-...-7 diagrams for a BEC(0.45).

Table 3.8: Decoded bit erasure probabilities of a(128,64) codes on a BEC(ϵ) over 2,000 trials and up to 15 iterations of the BP algorithm.

Code Type	$\epsilon = 0.05$	$\epsilon = 0.15$	$\epsilon = 0.25$	$\epsilon = 0.35$	$\epsilon = 0.45$
Polar (1-2-...-7)	0.00000	0.00151	0.05344	0.25140	0.43261
Polar (7-...-2-1)	0.00000	0.00000	0.00097	0.02096	0.19086

Finally, Table 3.9 gives the bit erasure probability performance of the (512, 256) polar code over 1,000 trials and up to 20 BP iterations. The 9-...-2-1 diagram has 1422 frozen variables, while the 1-2-...-9 diagram has 666 frozen variables. The capacity sum of the 256 unfrozen channels is equal to 251.68 for the 9-...-2-1 and 192.33 for the 1-2-...-9 diagrams for a BEC(0.45).

Table 3.9: Decoded bit erasure probabilities of a (512,256) codes on a BEC(ϵ) over 1,000 trials and up to 20 iterations of the BP algorithm.

Code Type	$\epsilon = 0.05$	$\epsilon = 0.15$	$\epsilon = 0.25$	$\epsilon = 0.35$	$\epsilon = 0.45$
Polar (1-2-...-9)	0.00000	0.00347	0.11889	0.33977	0.45005
Polar (9-...-2-1)	0.00000	0.00000	0.00003	0.00313	0.18507

Tables 3.7, 3.8 and 3.9 show that there is a significant difference between 1-2-...-*n* and *n*-...-2-1 diagrams diagrams, since the frozen bits of the input are adjusted according to the latter stage order. Superior performance of the latter diagram is related both to the higher capacity sum (CS) and the higher number of frozen variables (FV).

Bit erasure probabilities (BEPs) found in the simulations described above are sketched in Figures 3.9, 3.10 and 3.11 separately for the (32,16), (128, 64) and (512, 256) codes respectively, and in Figure 3.12 all together. The best case to worst case BEP ratio at a specific ϵ; i.e., "BEP for the *n*-...-1 stage order / BEP for the 1-...-*n* stage order" increases rapidly as N gets large, for instance, for a BEC(0.25), this ratio is $0.02690/0.00798 = 3.37$, $0.05344/0.00097 = 55$, and $0.11889/0.00003 = 3963$ for the (32,16), (128,64) and (512,256) codes respectively.

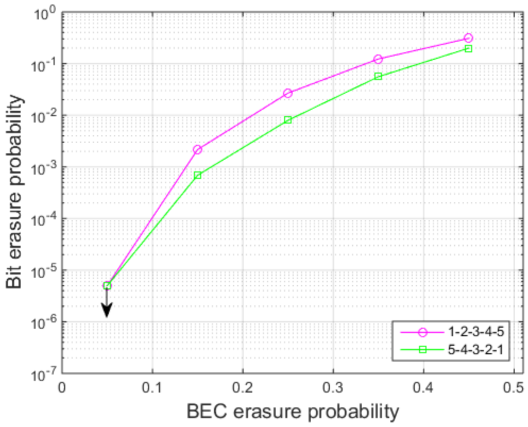


Figure 3.9: Decoded bit erasure probabilities of (32,16) codes over a BEC(ϵ).

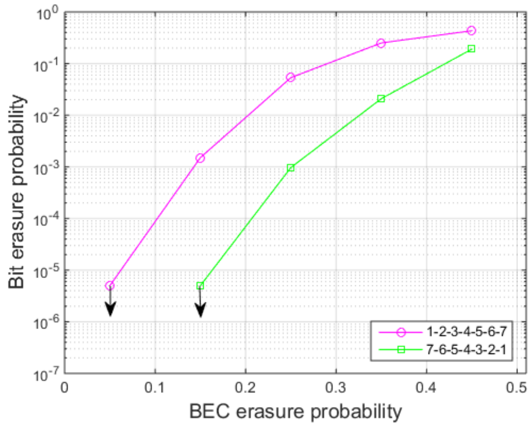


Figure 3.10: Decoded bit erasure probabilities of (128,64) over a BEC(ϵ) codes.

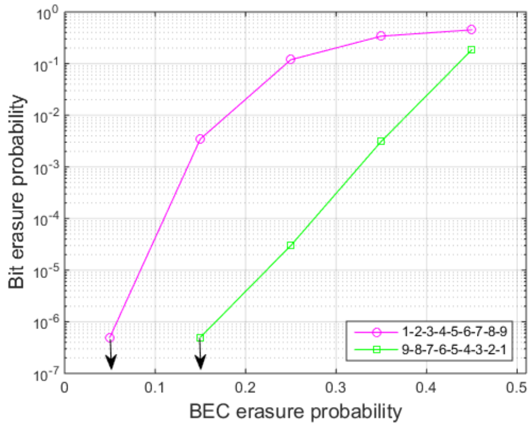


Figure 3.11: Decoded bit erasure probabilities of (512,256) over a BEC(ϵ) codes.

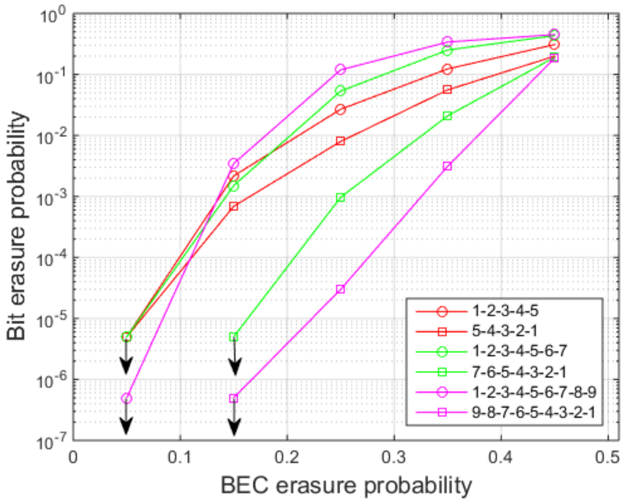


Figure 3.12: Decoded bit erasure probabilities of (32,16), (128,64), (512,256) codes over a BEC(ϵ).

3.6 Multiple Factor Graph BP Decoding with log$_2N$ and (log$_2N$)! Factor Graphs, and Two Proposals: Highest CS and Genie-Chosen Factor Graph Sets

In accordance with Korada's multiple factor graph decoding idea [Hussami et al., 2009], we have tried every permutation and found the number of frozen variables and capacity sum (CS) for each of the 720 diagrams, by keeping the 32 frozen bits at the input fixed.

Simulations are performed for a (64,32) polar code over a BEC(0.34375). 1,000 codewords are passed through the channel, by fixing the number of erasures to 22 for each word. The same channel realizations are used for every diagram to minimize fluctuations that can be caused by erasure patterns.

Single Factor Graph BP Decoding

Our simulations show that the best performance of 76 undecoded words is achieved by 3 separate diagrams, which have the "6-5-4-3-2-1", "6-4-5-3-2-1" and "4-6-5-3-2-1" stage orders. These 3 diagrams contain 88, 88 and 84 frozen variables and their CS values are 31.699, 31.699 and 31.613 respectively. The worst performance belongs to "1-3-2-5-6-4" stage order with 434 undecoded words. The corresponding diagram has 52 frozen variables and a CS value of 28.667. The complexity corresponding to the single factor graph BP decoding is $O(N(\log N))$.

Multiple Factor Graph BP Decoding with log$_2N$ Factor Graphs

In Table 3.10, the performances of 6 cyclic diagrams derived from the reference stage order 6-...-1 are seen. Decoding with cyclic factor graphs is proposed by Hussami and Korada in [Hussami et al., 2009] and [Korada, 2009]. "6-5-4-3-2-1" stage order has the best performance with 76 undecoded words out of 1,000 erased words. "2-1-6-5-4-3" stage order has the worst performance with 292 undecoded words. Since the multiple factor graph decoder chooses the best result among 6 cyclic diagrams in each trial, there remains only 53 undecoded words out of 1,000; i.e., 23 more words are decoded in addition to the performance of the 6-5-4-3-2-1 diagram.

The complexity of this method is $O(N(\log N)^2)$

Table 3.10: $n = 6$ cyclic diagrams and their performances for the (64,32) code over 1,000 codewords each having 22 erasures.

Diagram	CS	Number of frozen variables	Number of undecoded words	Number of remaining undecoded words
6-5-4-3-2-1	31.699	88	76	76
5-4-3-2-1-6	30.064	80	99	75
4-3-2-1-6-5	29.525	68	172	70
3-2-1-6-5-4	29.267	60	298	64
2-1-6-5-4-3	29.828	56	292	55
1-6-5-4-3-2	30.412	68	261	53

Multiple Factor Graph BP Decoding with $(\log_2 N)!$ Factor Graphs

There are $6! = 720$ different stage orders for $N = 64$. If the BP decoder performs on every stage order corresponding to all factor graphs, and the best result is chosen at each trial, the number of undecoded words can be decreased to 46. The complexity of this method is $O(N\log N(\log N)!)$, and this kind of complexity is not desired for large block length codes.

Multiple Factor Graph BP Decoding with a Diagram from Each CS Set

For the (64,32) code and over BEC(0.34375), there are 180 different CS values in 720 different diagrams, with each CS value shared by 4 different diagrams (see Appendix B). Each diagram with the same CS value also has the same number of frozen variables. Thus, the performances of the 4 diagrams within the equal-CS set are very similar. If one randomly picks up 1 out of these 4 diagrams with equal CS values, and uses multiple factor graph decoding; i.e., chooses the best of 180 results at each trial, again there remains 46 undecoded words; that is the performance of the multiple factor graph decoder with 180 factor graphs is the same as that of the decoder with 720 factor graphs. The complexity is reduced 4 times with respect to $(\log_2 N)!$ diagrams. However as the block length grows, the CS values start to differ a lot; i.e., for $N = 512$ every stage order has a different CS value. So, grouping diagrams according to their CS values may work for large block lengths, only when equal-CS sets are replaced by close valued-CS sets with CS varying in small intervals.

Multiple Factor Graph Decoding with the Best Diagrams

Another method to improve the performance can be choosing the best diagrams. For the (64,32) code, if one chooses the 19 diagrams with the highest CS values, the performance can be as good as in the case of $(\log_2 N)!$ multiple factor graphs and again, there remains 46 undecoded words. The chosen diagrams having the highest CS values are given in Table 3.11. The complexity of this method is 19 times as much as the single factor graph case, which ends up with 76 undecoded words.

Table 3.11: 19 diagrams with the best 19 CS values for the (64,32) code over 1,000 codewords each having 22 erasures.

Diagram	CS	Number of frozen variables	Number of undecoded words	Number of remaining undecoded words
6-5-4-3-2-1	31.699	88	76	76
4-6-5-3-2-1	31.613	84	76	76
6-4-3-5-2-1	31.560	88	83	74
4-5-6-2-3-1	31.500	80	84	69
6-4-5-2-1-3	31.485	88	84	59
4-6-3-5-2-1	31.459	84	83	59
6-2-4-5-3-1	31.447	84	100	56
4-6-5-3-1-2	31.384	84	83	51
6-4-3-2-5-1	31.355	88	91	50
6-4-3-5-1-2	31.331	88	87	49
6-4-5-1-3-2	31.280	88	101	47
2-6-4-5-3-1	31.260	76	115	47
6-2-4-3-5-1	31.246	84	109	47
4-6-3-5-1-2	31.241	84	88	47
4-5-6-2-1-3	31.236	80	87	47
4-6-3-2-5-1	31.228	84	92	47
4-5-3-6-2-1	31.207	80	82	47
4-2-6-5-3-1	31.198	76	100	47
6-2-4-5-1-3	31.183	84	102	46

Multiple Factor Graph Decoding with Genie-Chosen Diagrams

If one can add a genie that uses some pre-trained criteria, while choosing the diagrams to be used for decoding. The maximum performance can be achieved with lower complexity; such that with 5 diagrams, the number of undecoded words is decreased to 46. The chosen diagrams are shown in Table 3.12. The complexity of this method is similar to Korada's cyclic factor graph method, while it has a better performance..

Table 3.12: 5 diagrams that give the best performance for the (64,32) code over 1,000 codewords each having 22 erasures.

Diagram	CS	Number of frozen variables	Number of undecoded words	Number of remaining undecoded words
6-2-4-5-3-1	31.447	84	100	100
6-4-3-2-5-1	31.355	88	91	75
6-4-3-5-1-2	31.331	88	87	53
6-4-5-1-3-2	31.280	88	101	47
6-2-4-5-1-3	31.183	84	102	46

In Table 3.13, we summarize all simulation results found above for a (64,32) code.

Table 3.13: The number of undecoded words out of 1,000 words (each with 22 erasures) in each simulation, with multiple factor graph BP decoding for the (64,32) code.

Number of undecoded words						
Method	6-5-4-3-2-1 diagram	n cyclic diagrams	$n!$ diagrams	$180 = n!/4$ diagrams from all CS sets	19 (or 13) diagrams from highest CS's	5 (or 4) chosen diagrams
First simulation	76	53	46	46	46	46
Second simulation	90	54	44	45	45	45

Repeated Simulations with a Different Set of 1,000 Erased Words

A different set of 1,000 erasure patterns are generated to have an idea about the reliability of the results given in Table 3.13. Again, 1,000 codewords of the (64,32) polar code with 22 erasures in each codeword are fed to the BP decoder. For every diagram the same 1,000 erased words are used, and frozen bits are chosen according to the 6-5-4-3-2-1 diagram. In the last row of Table 3.13 the number of undecoded words are given for each case considered above. Korada's cyclic factor graphs decrease the number of undecoded words from 90 to 54. If all 720 diagrams are used, there remains 44 undecoded words. If one chooses a random diagram from all CS values, the number of undecoded words decreases to 45. This performance can be achieved by 13 diagrams chosen from the largest CS values. Also, 4 diagrams that are chosen on empirical evidence rather than systematically among 13 diagrams with the highest CS, decrease the number of undecoded words to 45 as well. These 4 diagrams are given in Table 3.14.

Table 3.14: 4 diagrams that give the best performance for the (64,32) code over 1,000 codewords each having 22 erasures.

Diagram	CS	Number of frozen variables	Number of undecoded words	Number of remaining undecoded words
6-4-3-5-2-1	31.560	88	94	94
6-4-5-2-1-3	31.485	88	92	59
4-6-5-3-1-2	31.384	84	100	53
6-2-4-3-5-1	31.246	84	113	45

3.7 Performance Dependence on the Number of Frozen Variables and Capacity Sum

In order to understand the performance dependence of the single factor graph BP decoder on the number of frozen variables (FV) and the capacity sum (CS) of the unfrozen paths, Figures 3.13 and 3.14 are plotted for 1,000 codewords of the (64,32) code over BEC(0.34375).

There are 10 different values (88, 84, 80, ..., 52) for the number of frozen variables corresponding to 720 diagrams for $N = 64$ (see Table B.2). Each given number of frozen variables is possessed by a different number of diagrams. So, there are the best, worst and average performances among the diagrams with the same number of frozen variables. Figure 3.13 gives the performance dependence on the number of frozen variables. As expected, the number of undecoded words decreases while the number of frozen variables increases.

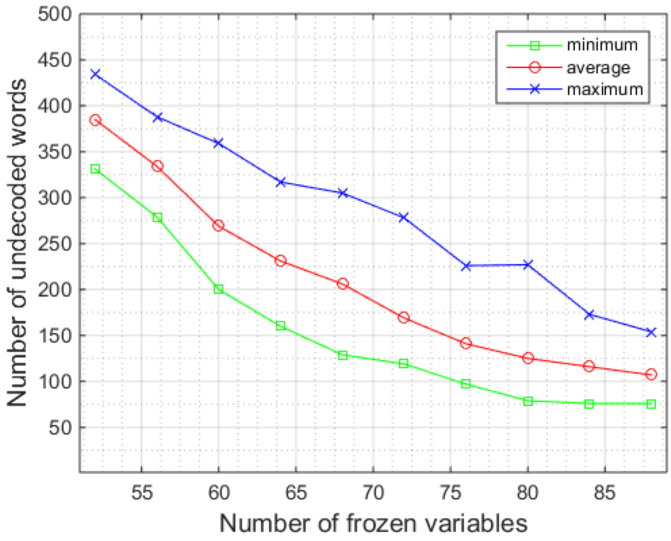


Figure 3.13: Minimum, maximum and average number of undecoded words for a (64,32) code over a BEC(0.34375).

For the (64,32) code, there are 180 different CS values, and each equal-CS value set contains 4 diagrams. Also, the number of frozen variables for all 4 diagrams in the equal-CS value set is the same. Thus, the performances are very close for those 4 diagrams. In Figure 3.14, the average number of undecoded words is depicted. Even though there are jumps in the figure, as the CS increases the average number of undecoded words decreases as expected. "6-5-4-3-2-1" diagram has the largest CS and the number of frozen variables, so it has the best performance corresponding to the rightmost point of the curve.

In our simulations for $N = 32, 64, 128, 512$ and 1024, we have encountered many instances, where the depth of the frozen variables on the factor graph has no effect on the performance. For $N = 32$, the superior performance of one factor graph over another can always be explained by either the higher number of frozen variables (FV), or higher sum of capacities (CS) corresponding to all information bit channels. However, for $N = 64$, one may confront a few odd cases, where neither CS, nor FV can explain the performance difference. The depth of the frozen bits on the graph also doesn't give any consistent explanation in such cases. Anyhow, these odd instances are quite rare, and the performance of a given factor graph is pretty much dependent on its CS and FV values on the average. Figure 3.15 depicts all possible CS values for each FV. The number of different CS values for each specific FV is as given in

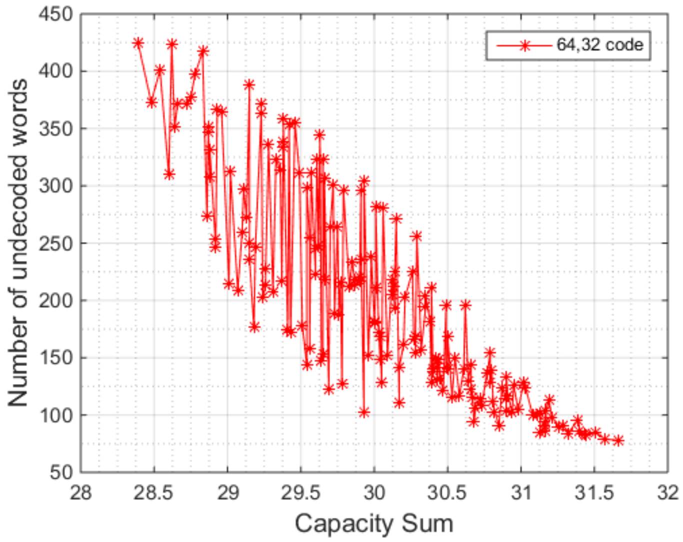


Figure 3.14: Average number of undecoded words for a (64,32) code versus the capacity sum over a BEC(0.34375).

Appendix B. In Figure 3.16, we plot the average CS values versus FV; and see almost linear correlation between these parameters.

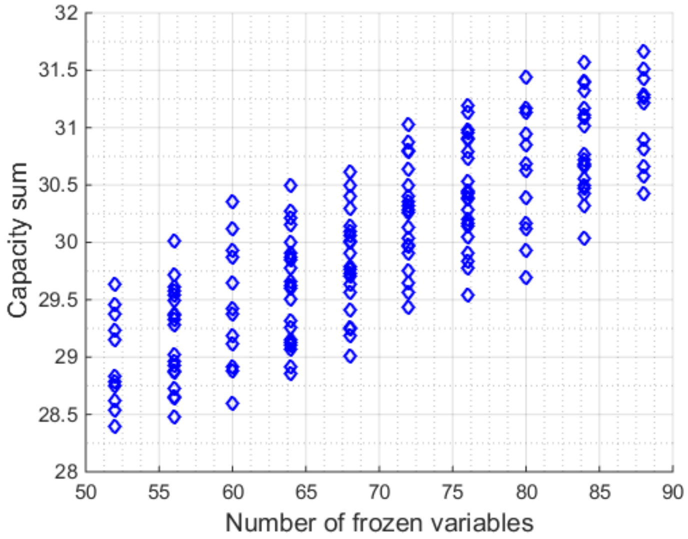


Figure 3.15: Capacity sums over a BEC(0.34375) versus number of frozen variables for all stage orders of the (64,32) code.

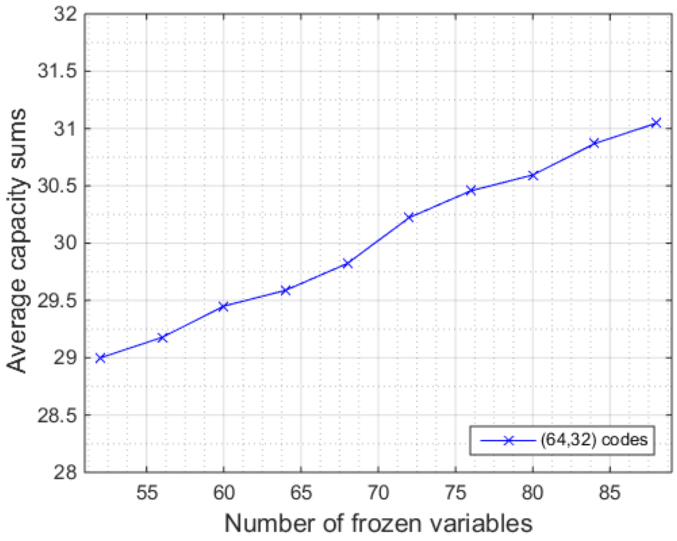


Figure 3.16: Average capacity sums over a BEC(0.34375) versus number of frozen variables for all stage orders of the (64,32) code.

3.8 Performance Comparison of Polar and Reed Muller Codes

Polar and Reed Muller codes can be characterized by the same factor graph representation. The only difference between these two codes are the chosen channels for sending the information and frozen bits.

Polar codes can be decoded with different factor graphs. As explained in Section 3.2, the number of frozen variables and the depth of the frozen variables in the factor graph are dependent on the stage order. Thus, BP decoding with different factor graphs has differences in performance.

Reed Muller codes also can be decoded with different factor graphs, but every one of the factor graphs has the same number of frozen variables; such that, two different factor graphs have identically the same frozen variables, i.e., the places of the frozen variables are the same. Thus, RM codes can be decoded on any factor graph. Decoding with all factor graphs gives almost the same performance, while there may be small performance differences caused by the special erasure patterns.

In Table 3.15, the number of frozen variables, the depth of the frozen variables and the performance are given for the (128,64) polar and Reed Muller codes. 1,000 codewords are transmitted over BEC(0.35). The same codeword and the same erasure pattern is used for every diagram to minimize experimental fluctuations.

(128,64) Reed Muller code has 140 frozen variables, depth of 4 in the 7-stage graph and 62.15 capacity sum value of chosen information bit channels. If compared with the polar code, which has 210 frozen variables, depth 5 and CS 63.70, RM code has greater number of undecoded words; and it has worse performance. Generally speaking, polar codes have larger number of frozen variables and better performance than RM codes.

Table 3.15: Performance of BP decoding (128,64) polar code with different factor graphs and (128,64) RM code over a BEC(0.35).

	Number of frozen variables	Depth of the frozen variables	Capacity sum	Number of Undecoded words
7-6-5-4-3-2-1	210	5	63.70	163
7-6-4-5-3-2-1	210	5	63.70	162
Reed Muller	140	4	62.15	579

CHAPTER 4

CONCLUSION

In this thesis, we simulate BP decoding of polar codes with block length N, using all possible factor graphs with $\log_2 N$ stages over binary erasure channels. We are interested in the relation between parameters like CS and FV, and their effects on the decoding performance. We define the capacity sum (CS) as the sum of all synthesized channel capacities used for the transmission of the information bits (so capacities of the $N - K$ frozen channels are excluded) and the number of frozen variables (FV) on the factor graph is defined as the sum of input frozen bits and those they generate in the inner stages of the graph. We empirically detect a strong positive correlation between CS and FV, as tabulated in Table 3.6 for the (32,16) code and plotted in Figure 3.16 for the (64,32) code. We also observe that an increase in CS and/or FV improves the performance of the BP decoder.

As for the BP decoding, we have considered the message directions in rather an unusual way as argued by [Xu et al., 2015], who have preferred the same unusual way. Starting from the encoder output, we proceed with only the left messages to the leftmost of the factor graph, and after reaching there, we advance to the right with only the right messages, until we reach to the rightmost end of the graph and complete the cycle, also called a round iteration. In terms of the required round iterations, we have found smaller iteration numbers than those seldomly declared in the literature (Arıkan uses 60 iterations, whereas we find that 5-15 round iterations are sufficient for similar codes). Although the statement of the algorithmic steps are the same everywhere, [Xu et al., 2015] assert that common application of the BP algorithm on factor graphs with $\log_2 N$ stages is to compute all left and right messages simultaneously and use them before leaving any stage for the adjacent one. Our literature survey does not make this detail clear. On the other hand, [Xu et al., 2015] pronounce an advantage that we can agree on the assumption that other BP decoders use more iterations (all we know is the 60 iterations mentioned above), and such a benefit speeds up the algorithm. Another detail that we discover from the literature is that Korada uses the BP algorithm in the direction of Z's that form the factor graph, starting from the bottom. Anyway, our BP decoding results for similar codes over the BEC are exactly the same as in [Korada, 2009], showing that our decoders are equivalent.

There are $(\log_2 N)!$ factor graph representations and among all possible factor graphs, if one chooses one graph and adjusts the input frozen bits of the polar encoder by computing all synthesized channel capacities and then choosing the maximum K capacities among N [Arıkan, 2009], the corresponding $\mathrm{CS}_{\mathrm{max}}$ needs to be the maximum in the set of all possible CS's for the given N and K. Now, if the factor graph is changed by keeping the frozen bits fixed, CS decreases and the resulting decoder structure is not optimum for polar code anymore. Still, different factor graphs are used in the literature within the context of multiple factor graph decoding with parallel paths [Korada, 2009]. This idea is useful, because as shown in Section 3.6, the performance of the BP decoder improves with multiple factor graph BP decoding. For all of these factor graphs, in addition to different values of CS, the number of frozen variables and the depth of the frozen variables in the factor graph are dependent on the stage order. Thus, BP decoding with different factor graphs has differences in performance.

In the light of the results given in Section 3.6, we do not recommend the use of all $(\log_2 N)!$ multiple factor graphs, since it does not bring sufficient improvement to account for the increased complexity from $O(N\log N)$ of single factor graph decoding to $O(N\log N(\log N)!)$. Also, $\log_2 N$ cyclic factor graphs suggested by Korada does not give the maximum performance improvement; because there are factor graphs in the cyclic cluster with bad performance. "3-2-1-6-5-4", "2-1-6-5-4-3" and "1-6-5-4-3-2" cyclic stage orders have small effects on decreasing the number of undecoded words, whenever the frozen bits are adjusted according to the “6-5-4-3-2-1” diagram.

Pre-trained choice of suitable factor graphs seems to have some practical significance.

On the contrary of Korada's cyclic idea, the stage orders with large CS and FV values are used for BP decoding, whose multiple factor graphs are pre-chosen according to some criteria. For our simulations given in Section 3.6, 19 (or 13) factor graphs which are chosen according their CS values, i.e., the highest 19 (or 13) CS values, brings improvement as much as $(\log_2 N)!$ multiple factor graphs. Moreover, 5 (or 4) factor graphs chosen from 19 (or 13) factor graphs gives the same performance. By some clever choice of factor graphs as explained, frame error ratio (FER) can be halved; i.e., reduced from say $90/1000$ to $45/1000$ at an erasure probability of $\epsilon = 0.34375$. The computational complexity increases at reasonable costs, such that, 5 (or 4) times the single factor graph decoding.

It is a future work to choose factor graphs that are used for multiple factor graph decoding and to theoretically derive the relation between CS and FV parameters.

REFERENCES

[Arıkan, 2008] Arıkan, E. (2008). A performance comparison of polar codes and reed-muller codes. *IEEE Commun. Lett*, 12(6):447–449.

[Arıkan, 2009] Arıkan, E. (2009). Channel polarization: A method for constructing capacity-achieving codes for symmetric binary-input memoryless channels. *Information Theory, IEEE Transactions on*, 55(7):3051–3073.

[Arıkan et al., 2015] Arıkan, E., Lentmaier, M., Montorsi, G., Sayir, J., et al. (2015). Challenges and some new directions in channel coding. *arXiv preprint arXiv:1504.03916*.

[Bahl et al., 1974] Bahl, L., Cocke, J., Jelinek, F., and Raviv, J. (1974). 284 IEEE Transacttons on Information Theory, March 1974.

[Berrou and Glavieux, 1996] Berrou, C. and Glavieux, A. (1996). Near optimum error correcting coding and decoding: Turbo-codes. *Communications, IEEE Transactions on*, 44(10):1261–1271.

[Bose and Ray-Chaudhuri, 1960] Bose, R. C. and Ray-Chaudhuri, D. K. (1960). On a class of error correcting binary group codes. *Information and Control*, 3(1):68–79.

[Elias, 1955] Elias, P. (1955). Coding for noisy channels. In *Proceedings of the Institute of Radio Engineers*, volume 43, pages 356–356.

[Eslami and Pishro-Nik, 2010] Eslami, A. and Pishro-Nik, H. (2010). On bit error rate performance of polar codes in finite regime. In *Communication, Control, and Computing (Allerton), 2010 48th Annual Allerton Conference on*, pages 188–194. IEEE.

[Eslami and Pishro-Nik, 2013] Eslami, A. and Pishro-Nik, H. (2013). On finite-length performance of polar codes: stopping sets, error floor, and concatenated design. *Communications, IEEE Transactions on*, 61(3):919–929.

[Forney Jr, 2001] Forney Jr, G. D. (2001). Codes on graphs: normal realizations. *Information Theory, IEEE Transactions on*, 47(2):520–548.

[Gallager, 1962] Gallager, R. G. (1962). Low-density parity-check codes. *Information Theory, IRE Transactions on*, 8(1):21–28.

[Golay, 1949] Golay, M. J. (1949). Notes on digital coding. *Proceedings of the Institute of Radio Engineers*, 37(6):657–657.

[Guo et al., 2014] Guo, J., Qin, M., Guillen i Fabregas, A., and Siegel, P. H. (2014). Enhanced belief propagation decoding of polar codes through concatenation. In *Information Theory (ISIT), 2014 IEEE International Symposium on*, pages 2987–2991. IEEE.

[Hocquenghem, 1959] Hocquenghem, A. (1959). Codes correcteurs d'erreurs. *Chiffres (paris)*, 2(147-156):116.

[Hussami et al., 2009] Hussami, N., Korada, S. B., and Urbanke, R. (2009). Performance of polar codes for channel and source coding. In *Information Theory, 2009. ISIT 2009. IEEE International Symposium on*, pages 1488–1492. IEEE.

[Korada, 2009] Korada, S. B. (2009). *Polar codes for channel and source coding*. PhD thesis, École Polytechnique Fédérale de Lausanne.

[MacKay and Neal, 1995] MacKay, D. J. and Neal, R. M. (1995). Good codes based on very sparse matrices. In *Cryptography and Coding*, pages 100–111. Springer.

[Muller, 1954] Muller, D. E. (1954). Application of boolean algebra to switching circuit design and to error detection. *Electronic Computers, Transactions of the IRE Professional Group on*, (3):6–12.

[Pamuk, 2011] Pamuk, A. (2011). An FPGA implementation architecture for decoding of polar codes. In *Wireless Communication Systems (ISWCS), 2011 8th International Symposium on*, pages 437–441. IEEE.

[Pearl, 1988] Pearl, J. (1988). *Probabilistic reasoning in intelligent systems: Networks of plausible reasoning*. Morgan Kaufmann Publishers, Los Altos.

[Reed, 1954] Reed, I. (1954). A class of multiple-error-correcting codes and the decoding scheme. *Transactions of the IRE Professional Group on Information Theory*, 4(4):38–49.

[Reed and Solomon, 1960] Reed, I. S. and Solomon, G. (1960). Polynomial codes over certain finite fields. *Journal of the Society for Industrial and Applied Mathematics*, 8(2):300–304.

[Shannon, 1948] Shannon, C. (1948). A mathematical theory of communication. *Bell System Technical Journal*, 27:379–423 and 623–656.

[Tal and Vardy, 2011] Tal, I. and Vardy, A. (2011). List decoding of polar codes. In *Information Theory Proceedings (ISIT), 2011 IEEE International Symposium on*, pages 1–5. IEEE.

[Viterbi, 1967] Viterbi, A. J. (1967). Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *Information Theory, IEEE Transactions on*, 13(2):260–269.

[Wiberg, 1996] Wiberg, N. (1996). *Codes and decoding on general graphs*. Citeseer.

[Wiberg et al., 1995] Wiberg, N., Loeliger, H.-A., and Kötter, R. (1995). Codes and iterative decoding on general graphs. *European Transactions on Telecommunications*, 6(5):513–525.

[Xu et al., 2015] Xu, J., Che, T., and Choi, G. (2015). Xj-bp: Express journey belief propagation decoding for polar codes. *arXiv preprint arXiv:1504.06025*.

[Yuan and Parhi, 2014a] Yuan, B. and Parhi, K. (2014a). Early stopping criteria for energy-efficient low-latency belief-propagation polar code decoders. *Signal Processing, IEEE Transactions on*, 62(24):6496–6506.

[Yuan and Parhi, 2014b] Yuan, B. and Parhi, K. K. (2014b). Algorithm and architecture for hybrid decoding of polar codes. *arXiv preprint arXiv:1411.7286*.

APPENDIX A

REQUIRED NUMBER OF ROUND ITERATIONS FOR THE BP DECODER FOR *N*=128 & 512

In Table A.1, we observe that the maximum value of iterations at which a codeword is decoded is 13 at $\epsilon = 0.35$. So, for the (128,64) polar code, the maximum number of round iterations in the BP decoding algorithm can be fixed to 15 for the stage order 1-...-*n*. And from Table A.2, we can observe that 10 is sufficient for the maximum number of round iterations for the stage order *n*-...-1.

For the (512, 256) polar code and erasure probabilities up to 0.35, Table A.3 shows that there is no decoded codeword after the 20^{th} iteration for the stage order 1-...-*n*. So, the maximum number of round iterations can be chosen as 20 for this case. However, for the stage order *n*-...-1 given in Table A.4, there is no decoding after 9^{th} iteration up to $\epsilon = 0.35$. Thus the maximum number of round iterations can be fixed to 10 which is half of the chosen maximum number of round iterations value of the stage order 1-...-*n*.

For $\epsilon = 0.45$, the stage order 1-...-*n* can not decode any word at any iteration. So, it is not possible to make comparison with the stage order *n*-...-1.

Table A.1: Number of decoded codewords out of 1,000 received words of the (128,64) polar code after each round iteration for the stage order 1-...-*n* over a BEC(ϵ).

ϵ Iter. number	0.05	0.15	0.25	0.35	0.45
1	871	24	0	0	0
2	128	588	29	0	0
3	1	323	200	5	0
4	0	48	263	21	0
5	0	7	146	46	0
6	0	1	62	45	0
7	0	2	28	41	1
8	0	0	15	24	2
9	0	0	3	19	2
10	0	0	2	13	1
11	0	0	0	5	2
12	0	0	0	5	2
13	0	0	0	2	0
14	0	0	0	0	0
15	0	0	0	0	0
.	.	.	.	.	.
.	.	.	.	.	.
.	.	.	.	.	.
30	0	0	0	0	0
Number of undecoded words	0	7	252	774	990

Table A.2: Number of decoded codewords out of 1,000 received words of the (128,64) polar code after each round iteration for the stage order *n*-...-1 over a BEC(ϵ).

ϵ Iter. number	0.05	0.15	0.25	0.35	0.45
1	989	495	31	0	0
2	11	488	584	78	0
3	0	16	344	406	25
4	0	0	28	275	91
5	0	0	5	70	104
6	0	0	0	14	44
7	0	0	0	6	16
8	0	0	0	3	3
9	0	0	0	0	2
10	0	0	0	0	0
11	0	0	0	0	0
12	0	0	0	0	0
13	0	0	0	0	0
14	0	0	0	0	0
15	0	0	0	0	0
.	.	.	.	.	.
.	.	.	.	.	.
.	.	.	.	.	.
30	0	0	0	0	0
Number of undecoded words	0	1	8	148	715

Table A.3: Number of decoded codewords out of 1,000 received words of the (512,256) polar code after each round iteration for the stage order 1-...-*n* over a BEC(ϵ).

ϵ Iter. number	0.05	0.15	0.25	0.35	0.45
1	65	0	0	0	0
2	891	0	0	0	0
3	44	75	0	0	0
4	0	424	0	0	0
5	0	330	2	0	0
6	0	105	32	0	0
7	0	33	55	0	0
8	0	6	74	0	0
9	0	1	53	0	0
10	0	2	51	0	0
11	0	0	32	0	0
12	0	0	29	0	0
13	0	0	16	1	0
14	0	0	9	0	0
15	0	0	3	2	0
16	0	0	0	0	0
17	0	0	0	0	0
18	0	0	0	0	0
19	0	0	0	1	0
20	0	0	0	1	0
21	0	0	0	0	0
22	0	0	0	0	0
.	.	.	.	.	.
.	.	.	.	.	.
.	.	.	.	.	.
30	0	0	0	0	0
Number of undecoded words	0	24	644	995	1000

Table A.4: Number of decoded codewords out of 1,000 received words of the (512,256) polar code after each round iteration for the stage order *n*-...-1 over a BEC(ϵ).

ϵ Iter. number	0.05	0.15	0.25	0.35	0.45
1	961	19	0	0	0
2	39	893	64	0	0
3	0	87	798	17	0
4	0	1	128	359	0
5	0	0	7	431	2
6	0	0	0	100	15
7	0	0	0	18	38
8	0	0	0	9	47
9	0	0	0	2	9
10	0	0	0	0	20
11	0	0	0	0	13
12	0	0	0	0	4
13	0	0	0	0	2
14	0	0	0	0	0
15	0	0	0	0	1
16	0	0	0	0	0
17	0	0	0	0	0
18	0	0	0	0	0
19	0	0	0	0	0
20	0	0	0	0	0
21	0	0	0	0	0
22	0	0	0	0	0
.	.	.	.	.	.
.	.	.	.	.	.
.	.	.	.	.	.
30	0	0	0	0	0
Number of undecoded words	0	0	3	64	849

APPENDIX B

CORRELATION BETWEEN THE NUMBER OF FROZEN VARIABLES AND THE CAPACITY SUMS

In Section 3.4, we have given an example to show the positive correlation between the number of frozen variables (FV) and the corresponding capacity sums (CS) of the unfrozen paths of factor graphs, considering $5! = 120$ different diagrams of the (32,16) code. These 120 diagrams gather into 10 sets, each containing 12 diagrams with equal CS values found over a BEC(0.3125); as tabulated in Table 3.6. We observe that as the number of FV decreases, the corresponding average CS value also decreases.

In order to ease the presentation of the distribution of CS and FV values for a (64,32) code, we first summarize the distribution of equal-CS sets given in Table 3.6 by a simpler table for the (32,16) code, Table B.1.

Table B.1: Distribution of equal-CS sets (each having 12 elements) for the (32,16) code over a BEC(0.3125), expressed differently from Table 3.6.

Number of Frozen Variables	Number of Diagrams	Number of Equal-CS Sets (each having 12 elements)	Maximum Capacity Sum	Minimum Capacity Sum
44	12	1	15.82	15.82
36	24	2	15.68	15.50
32	36	3	15.57	15.16
28	36	3	15.46	15.05
24	12	1	14.94	14.94

We can then proceed with a similar table for the (64,32) code, namely Table B.2. Close inspection of Table again indicates positive correlation between the CS and FV values of the (64,32) codes over a BEC(0.34375), similar to the case of the (32,16) code over BEC(0.3125). Total number of $6! = 720$ factor graphs are grouped into 4-element sets with equal CS; hence, there are 180 different CS values which vary in the interval [28.57,31.70]. The number of frozen variables, FV, takes 10 different values in the interval [52,88] with steps of 4.

Table B.2: Distribution of equal-CS sets (each having 4 elements) for the (64,32) polar code over a BEC(0.34375).

Number of Frozen Variables	Number of Diagrams	Number of Equal-CS Sets (each having 4 elements)	Maximum Capacity Sum	Minimum Capacity Sum
88	48	12	31.70	30.53
84	84	21	31.61	30.13
80	48	12	31.50	29.82
76	96	24	31.26	29.68
72	84	21	31.11	29.58
68	84	21	30.69	29.16
64	96	24	30.59	29.01
60	48	12	30.44	28.77
56	84	21	30.11	28.66
52	48	12	29.74	28.57

In Table B.3 correlation between the CS and FV values of the (128,64) codes over a BEC(0.35) is given. Total number of $7! = 5040$ factor graphs are grouped into 2-element sets with equal CS which vary in the interval [55.88,63.77]. The number of frozen variables takes 28 different values in the interval [98,210] with steps of 4 except 102. However, for RM code every factor graphs have 140 frozen variables and 62.15 capacity sum.

As N gets larger, the number of different CS values increases and the size of equal-CS sets rapidly reduces to 1, instead of the 12 elements for the (32,16) and 4 elements for the (64,32) code.

Table B.3: Distribution of equal-CS sets (each having 2 elements) for the (128,64) polar code over a BEC(0.35).

Number of Frozen Variables	Number of diagrams	Number of Equal-CS Sets (each having 2 elements)	Maximum Capacity Sum	Minimum Capacity Sum
210	72	36	63.70	60.49
206	72	36	63.63	60.38
202	120	60	63.57	60.15
198	96	48	63.51	60.41
194	192	96	63.45	60.12
190	120	60	63.37	59.96
186	264	132	63.15	59.29
182	168	84	62.83	58.77
178	192	96	62.45	58.63
174	144	72	62.33	58.13
170	336	168	62.11	58.12
166	168	84	61.63	58.10
162	288	144	61.77	57.68
158	120	60	61.71	57.42
154	288	144	61.58	57.48
150	48	24	61.01	57.39
146	456	228	61.42	57.25
142	120	60	61.30	57.29
138	384	192	61.07	56.78
134	96	48	61.13	56.95
130	312	156	60.86	56.47
126	48	24	57.97	56.51
122	288	144	59.82	56.24
118	144	72	59.56	56.62
114	96	48	58.98	56.63
110	96	48	59.03	56.24
106	192	96	58.94	55.95
98	120	60	57.70	55.88