TECHNICAL UNIVERSITY OF CRETE
ELECTRICAL AND COMPUTER ENGINEERING DEPARTMENT
TELECOMMUNICATIONS DIVISION

Polar-code construction and decoding techniques

by

Magda Amiridi

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF
THE REQUIREMENTS FOR THE DIPLOMA OF
ELECTRICAL AND COMPUTER ENGINEERING

January 2018

THESIS COMMITTEE

Associate Professor George N. Karystinos, *Thesis Supervisor*
Associate Professor Aggelos Bletsas
Professor Athanasios Liavas

Abstract

Polar codes, recently invented by Arikan, are the first provably capacity-achieving codes for any binary input symmetric discrete memoryless channel with low encoding and decoding complexity. This thesis explores the practical implementation of polar codes which are complexity efficient and perform well for binary erasure channel (BEC) and binary symmetric channel (BSC). The explicit code construction is based on a characteristic called channel polarization which involves generating N extremal (perfect or completely noisy) channels from N independent uses of the same base channel. Information bits are sent over the noiseless channels while pilot bits, called frozen bits, are assigned to the noisy ones. Code design for BEC is based on the recursive relations presented in the original paper whereas for BSC we propose a heuristic and efficient algorithm and compare it to the method of recursive estimation of Bhattacharyya parameters of bit-channels. The encoding is implemented using a recursive butterfly structure with $\mathcal{O}(N \log N)$ complexity, where N is the block length of the code. Two main low complexity decoders are compared in terms of bit error rate: successive cancellation decoder proposed by Arikan having complexity $\mathcal{O}(N \log N)$ with susceptibility to error propagation and mediocre bit error rate performance at small or moderate code lengths and list decoder, proposed by Tal and Vardy, with complexity $\mathcal{O}(LN \log N)$ where L is the list size.

Acknowledgements

First of all, I would like to express my deepest gratitude to my thesis supervisor, Associate Professor George N. Karystinos, for all his guidance, support and patience throughout this work. It was his insightful and fascinating lectures on information and coding theory that attracted me into this field. I would also like to thank him for his efforts to improve my presentation and writing skills.

I would like to thank Associate Professor Aggelos Bletsas and Professor Athanasios Liavas for taking out time from their busy schedules and being a part of my thesis committee.

My sincere thanks also goes to Associate Professor Antonios Deligiannakis.

In addition, I would like to thank my buddies for being there for me and the wonderful time I spent with them during this journey. One person that stood out during these years is my closest friend and colleague Ioannis P. whose companionship made the past five years an unforgettable stage of my life full of memorable moments.

Finally, I would especially like to thank my family for their unconditional love and support and dedicate this thesis to my grandfather.

Table of Contents

List of Figures

Chapter 1

Introduction

1.1 Milestones in coding

To take the issue from the beginning, Shannon defined channel capacity of a noisy channel in 1948 as the highest rate at which a sender can reliably transmit data to a receiver, measuring the channel's quality. Two years later, Hamming introduced the first class of linear codes, error-correcting codes for which any linear combination of codewords is also a codeword as well as the Hamming distance. In 1954, Reed-Muller codes emerged and a year later Elias introduced the erasure channel, convolutional codes and also showed that random parity-check codes achieve capacity on the BEC. In 1959, BCH codes, generalization of the Hamming code for multiple-error correction became known. 1960 was the year 2 codes appeared: Low-density parity-check (LDPC) codes, introduced by Gallager, and Reed-Solomon codes.

More than 35 years passed before capacity was achieved in practice. In 1993, a new class of convolution codes called Turbo Codes emerged, whose performance in terms of BER were close to the Shannon limit. Furthermore, being impractical to implement when first developed by Gallager in 1963, LDPC codes' incredible potential remained undiscovered due to the computational demands of simulation until 1995 when they were rediscovered by McKay and Neal.

In 2008, polar codes, introduced by Erdal Arikan, are the first provably capacity achieving codes for binary-input discrete memoryless symmetric channels with low encoding and decoding complexity. Polar coding theory became a popular research area soon after and polar codes became a competitive can-

didate coding scheme in future wireless communication systems (5G system).

1.2 Discrete memoryless channels

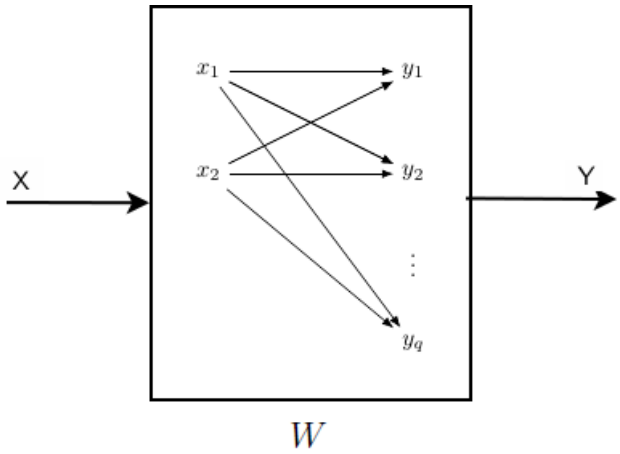


Figure 1.1: Binary-input discrete memoryless channel.

Considering a discrete memoryless channel with input alphabet $\mathcal{X}$={0,1} and output alphabet $\mathcal{Y}$, we define the symmetric capacity of the channel as

$$I(W) \triangleq \frac{1}{2} \sum_{y \in \mathcal{Y}} \sum_{x \in \mathcal{X}} W(y|x) \log_2 \frac{W(y|x)}{\frac{1}{2}W(y|0) + \frac{1}{2}W(y|1)} \,. \tag{1.1}$$

Equivalently, symmetric capacity can be reffered to as the mutual information between the input and the output of the channel, given uniformly distributed input. The symmetric capacity is a measure of the rate of information transmission. If the channel is symmetric (a discrete memoryless channel is symmetric if for all $\mathcal{X}$, the branches leaving the code symbol have the same set of probabilities, p_1, p_2,...,$p_{|Y|}$ and for all $\mathcal{Y}$, the branches entering the received symbol have the same set of probabilities p_1, p_2,...,$p_{|X|}$), then its symmetric capacity is equal to its Shannon capacity.

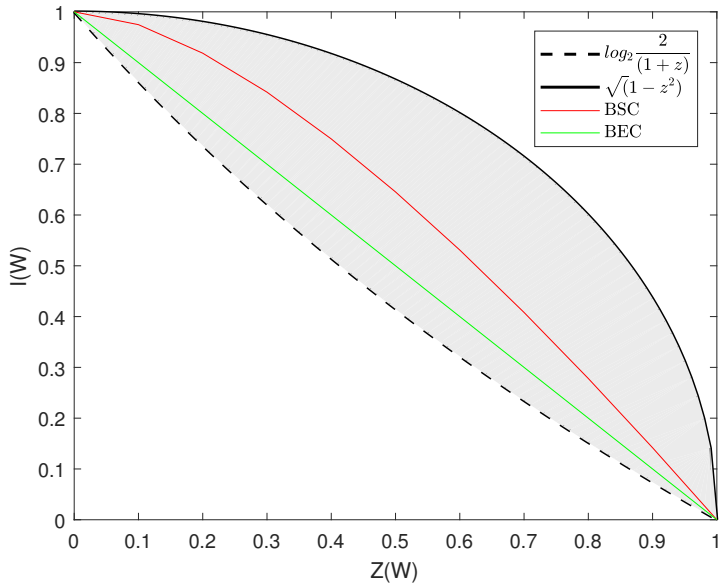


Figure 1.2: Relation between symmetric capacity and Bhatacharyya parameter for binary-input discrete memoryless channels.

Another important channel parameter is the Bhatacharyya parameter, which is denoted as

$$Z(W) \triangleq \sum_{y \in \mathcal{Y}} \sqrt{W(y|0)W(y|1)}\,. \tag{1.2}$$

The Bhatacharyya parameter is a measure of the reliability of information transmission. If we use W to transmit a bit, then $Z(W)$ is an upper bound on the error probability of maximum likelihood (ML) decision.

The symmetric capacity and Bhatacharyya parameter are related by

$$I(W) \geq \log_2 \frac{2}{1+Z(W)}, \tag{1.3}$$

$$I(W) \leq \sqrt{1-Z(W)^2}. \tag{1.4}$$

1.2.1 Binary erasure channel (BEC)

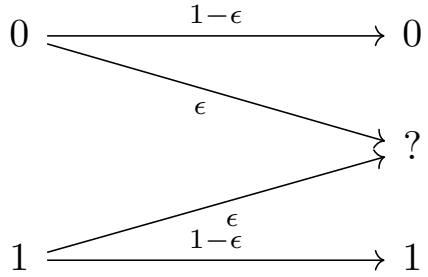


Figure 1.3: The binary erasure channel with erasure probability ϵ.

The binary erasure channel was introduced by Elias as a toy example in 1954. Erasure channels can be used to model data networks, where packets either arrive correctly or are lost due to buffer overfows or excessive delays. Erasure channels model situations where information may be lost but is never corrupted. A transmitted bit is either received correctly with probablity $1-\epsilon$ or known to be lost with probablity ϵ. Erasure occurs for each transmitted bit independently, thus the channel is memoryless.
The capacity of BEC(ϵ) is $C_{\mathrm{BEC}(\epsilon)} = 1-\epsilon$ and the Bhattacharyya parameter of such a channel is $Z_{\mathrm{BEC}(\epsilon)} = \epsilon$.

1.2.2 Binary symmetric channel (BSC)

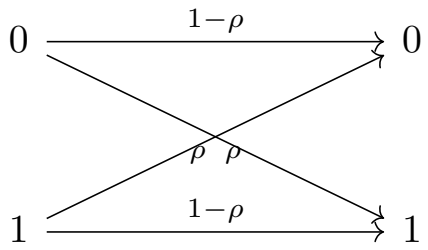


Figure 1.4: The binary symmetric channel with error probability ρ.

The binary symmetric channel is one of the simplest channels. The transmitted bit is received correctly with probability $1-\rho$ or "flipped" with crossover probability ρ.

If X is the transmitted random variable and Y is the received variable, then the conditional probabilities of the channel are

$\Pr(Y = 0|X = 0)$=$1-\rho$,
$\Pr(Y = 0|X = 1)$=ρ,
$\Pr(Y = 1|X = 0)$=ρ,
$\Pr(Y = 1|X = 1)$=$1-\rho$.

The capacity of BSC(ρ) is $C_{\mathrm{BSC}(\rho)}$=$1-\mathcal{H}(\rho)$, where $\mathcal{H}(\rho)$=$-\rho\log_2\rho-(1-\rho)\log_2(1-\rho)$, and the Bhattacharyya parameter of such a channel is $Z_{\mathrm{BSC}(\rho)}$ $= 2\sqrt{p(1-p)}$.

1.3 Definitions and preliminaries

Let N and K be positive integers with $K \leq N$. An (N, K) block code is a function from $\{0,1\}^K$ to $\{0,1\}^N$, i.e., its input is a vector of K bits and its output is a vector of N bits.

A specific polar code is fully defined by the four parameters (N, K, A, u_{A_C}).

- N: Block length, i.e., the total number of bits transmitted over the channel.
- K: The number of good channels. $K = \lfloor RN \rfloor$, where R is the rate that we choose to transmit.
- A: The information set $A \subset \{1, ..., N\}$, i.e., the set of indices which correspond to the good channels.
- u_{A_C}: The frozen bits, i.e., bits which have fixed values.

The transmission process works in the following way. To begin with, the information set A must be chosen according to the particular channel over

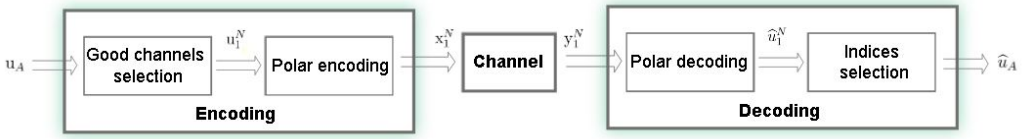


Figure 1.5: High-level transmission scheme.

which transmission takes place. Different channels yield different information sets. NR information bits are assigned to the good channels indicated by set A (vector u_A) and $N(1-R)$ frozen bits, typically taking a fixed zero value, are assigned to the rest of channels (vector u_{A_C}). The combined vector u_1^N of u_A and u_{A_C} is encoded in a recursive structure obtaining vector x_1^N, which is sent over the vector channel W^N. W^N corresponds to N uses of any binary-input discrete memoryless channel (B-DMC). The received vector y_1^N is decoded by the polar decoder, which produces an estimate of u_A, the vector $\hat{u}_A$.

An important characteristic of polar codes is their non universality. Different polar codes are generated depending on the specified value of erasure probability ϵ for BEC or crossover probability ρ for BSC.

Chapter 2

Channel Polarization: The key-idea behind Polar Codes

2.1 Basic channel transformation

Combining two independent copies of W as shown in Figure (2.1) creates a new synthetic channel W_2, with capacity $2I(W)$. The channel W_2 is then decomposed into two synthesized channels: $\mathrm{W}_2^{(1)} : u_1 \to (y_1, y_2)$ (which we call degraded, for reasons that will be explained shortly) and $\mathrm{W}_2^{(2)} : u_2 \to (y_1, y_2, u_1)$ (which we call upgraded).

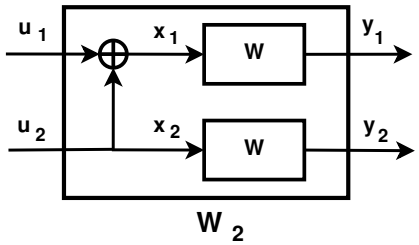


Figure 2.1: Channel W_2.

To calculate the symmetric capacity and the Bhattacharyya parameter of the upgraded and degraded channels, we must first calculate the transition probabilities of the two synthesized bit-channels.

For the first bit-channel, we have

$$\begin{aligned} p(y_1, y_2|u_1) &= \sum_{u_2} p(y_1, y_2|u_1, u_2)p(u_2|u_1) \\ &= \sum_{u_2} p(y_1, y_2|u_1, u_2)p(u_2) \end{aligned}$$

$$
\begin{aligned}
&= \sum_{u_2} p(y_1, y_2|u_1, u_2)\frac{1}{2} \\
&= \frac{1}{2}p(y_1, y_2|u_1, u_2 = 0) + \frac{1}{2}p(y_1, y_2|u_1, u_2 = 1) \\
&= \frac{1}{2}p(y_1, y_2|x_1, x_2 = 0) + \frac{1}{2}p(y_1, y_2|x_1, x_2 = 1) \\
&= \frac{1}{2}p(y_1|x_1)p(y_2|x_2 = 0) + \frac{1}{2}p(y_1|x_1)p(y_2|x_2 = 1) \\
&= \sum_{u_2} \frac{1}{2}p(y_1|u_1 \oplus u_2)p(y_2|u_2). \qquad (2.1)
\end{aligned}
$$

$$
I(u_1; y_1, y_2) = \sum_{y_1, y_2, u_1} p(y_1, y_2|u_1)p(u_1) \log_2 \frac{p(y_1, y_2|u_1)}{\frac{1}{2}p(y_1, y_2|u_1 = 0) + \frac{1}{2}p(y_1, y_2|u_1 = 1)}\,. \qquad (2.2)
$$

$$
Z(W') = \sum_{y_1, y_2} \sqrt{p(y_1, y_2|u_1 = 0)p(y_1, y_2|u_1 = 1)}\,. \qquad (2.3)
$$

For the second bit-channel, we have

$$
\begin{aligned}
p(y_1, y_2, u_1|u_2) &= p(y_1, y_2|u_1, u_2)p(u_2|u_1) \\
&= p(y_1, y_2|u_1, u_2)\frac{1}{2} \\
&= \frac{1}{2}p(y_1, y_2|x_1, x_2) \\
&= \frac{1}{2}p(y_1|x_1)p(y_2|x_2) \\
&= \frac{1}{2}p(y_1|u_1 \oplus u_2)p(y_2|u_2). \qquad (2.4)
\end{aligned}
$$

$$
I(u_2; y_1, y_2, u_1) = \sum_{y_1, y_2, u_1, u_2} p(y_1, y_2, u_1|u_2) \log_2 \frac{p(y_1, y_2, u_1|u_2)}{\frac{1}{2}p(y_1, y_2, u_1|u_2 = 0) + \frac{1}{2}p(y_1, y_2, u_1|u_2)}\,. \qquad (2.5)
$$

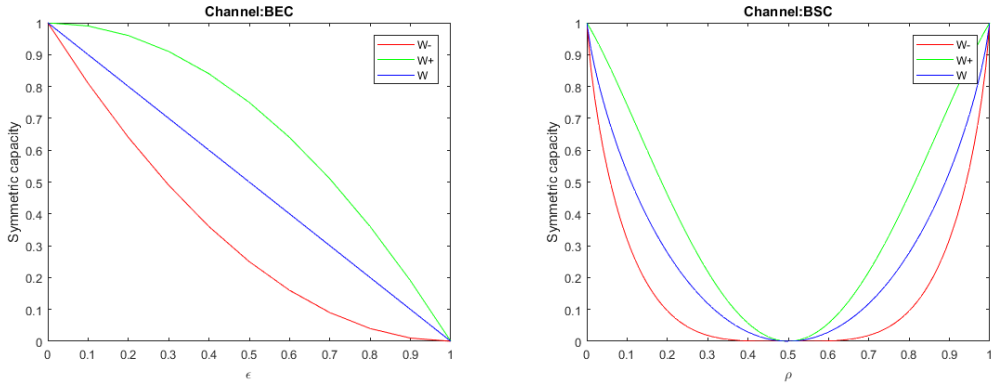


Figure 2.2: Symmetric capacity of BEC and BSC before and after the basic polarization step.

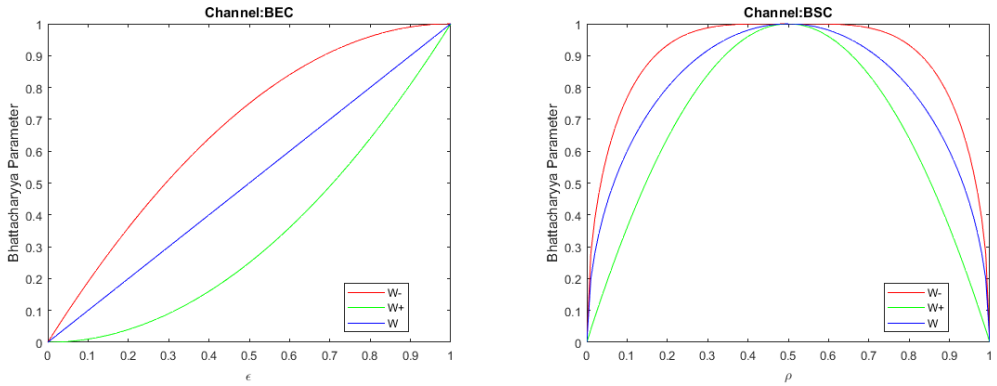


Figure 2.3: Bhattacharyya parameter of BEC and BSC before and after the basic polarization step.

$$Z(W'') = \sum_{y_1,y_2,u_1} \sqrt{p(y_1,y_2,u_1|u_2=0)p(y_1,y_2,u_1|u_2)}\,. \tag{2.6}$$

In the case W is a BEC with erasure probability ϵ, the capacities of channels W' and W'' are equal to the capacities of BECs with erasure probabilities 2ϵ-ϵ^2 and ϵ^2, respectively. Thus $I(W') = 1 - (2\epsilon\text{-}\epsilon^2)$ and $I(W'') = 1 - \epsilon^2$.

The two bit channels created after the basic polarization step are defined as upgraded and degrated channels with respect to the original one in terms of

symmetric capacity which is illustrated in Figure (2.2) for BEC and BSC. Figure (2.3) demonstrates the Bhattacharyya parameters before and after the basic polarization step. We observe that a degraded channel has larger Bhattacharrya parameter and smaller symmetric capacity than the original channel. The opposite applies to the upgraded channel.

The channel polarization is a capacity preserving operation, which is mathematically expressed as $I(W') + I(W'') = 2I(W)$.

Proof

$I(\mathrm{W}_2^{(1)}) = I(u_1; y_1, y_2)$
$I(\mathrm{W}_2^{(2)}) = I(u_2; y_1, y_2, u_1)$

Since u_1 and u_2 are independent, $I(u_2; y_1, y_2, u_1)$ equals $I(u_2; y_1, y_2|u_1)$. So, by the chain rule and due to the one-to-one relation between (x_1, x_2) and (u_1, u_2), we have

$$\begin{aligned} I(W') + I(W'') &= I(u_1, u_2; y_1, y_2) \\ &= I(x_1, x_2; y_1, y_2) \\ &= I(x_1; y_1) + I(x_2; y_2) \\ &= 2I(W). \end{aligned} \tag{2.7}$$

After polarization, capacity is conserved but redistributed unevenly. The synthesized bit-channels, the upgraded and the degraded one, have lower or higher, respectively, capacity compared to the original channel. For the basic polarization step, this property of capacity extremization is mathematically expressed as

$$I(W') \leqslant I(W) \leqslant I(W'') \tag{2.8}$$

Equality holds iff $I(W)$ equals 0 or 1.

Proof

$$\begin{aligned} I(W'') &= I(u_2; y_1, y_2, u_1) \\ &= I(u_2; y_2) + I(u_2; y_1, u_1|y_2) \\ &= I(W) + I(u_2; y_1, u_1|y_2) \end{aligned} \tag{2.9}$$

This shows that I $(W'') \geqslant I(W)$.

Combining this and the capacity conservation property which was proved previously, namely $I(W') + I(W'') = 2I(W)$, it is implied that $I(W') \leqslant 2I(W)$.

Regarding the Bhattacharyya parameter of the synthesized channels, Arikan [2] proved the following relations in case the base channel is B−DMC

$$Z(W'') = Z(W)^2, \tag{2.10}$$

$$Z(W') \leq 2Z(W) - Z(W)^2, \tag{2.11}$$

$$Z(W') \geq Z(W) \geq Z(W''). \tag{2.12}$$

Equality holds in (2.11) iff W is a BEC.

2.2 General channel transformation

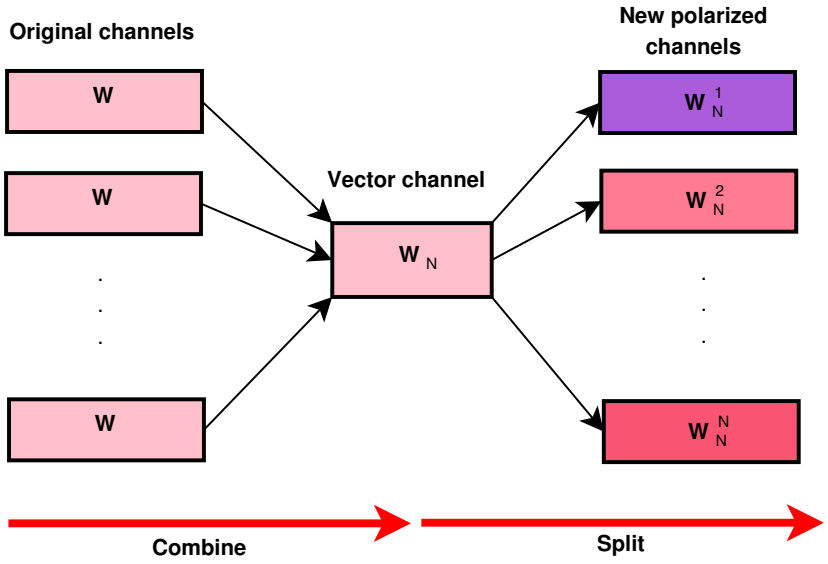


Figure 2.4: Channel combining and splitting.

The concept of polarization lies among two processes: **channel combining** and **channel splitting**. N independent copies of the original channel W are combined to form a vector channel W_N in a recursive way and then the new super channel W_N is split back into N virtual channels $\mathrm{W}_N^{(i)}$, $\{1 \leq i \leq N\}$ which are called i-th bit-channels of W. 0th level of recursion $(n = 0, N = 1)$ is simply using the channel W by itself, $W_1 \stackrel{\circ}{=} W$ and the 1st level of recursion $(n = 1, N = 2)$ is the basic polar transformation that was discussed in section (2.1).

For example, we use two independent copies of W_2 to create the vector channel W_4 Figure (2.5). The W_4 channel is decomposed into four synthesized channels: $\mathrm{W}_4^{(1)} : u_1 \to (y_1^4)$, $\mathrm{W}_4^{(2)} : u_2 \to (y_1^4, u_1)$, $\mathrm{W}_4^{(3)} : u_3 \to (y_1^4, u_1^2)$, and $\mathrm{W}_4^{(4)} : u_2 \to (y_1^4, u_1^3)$.

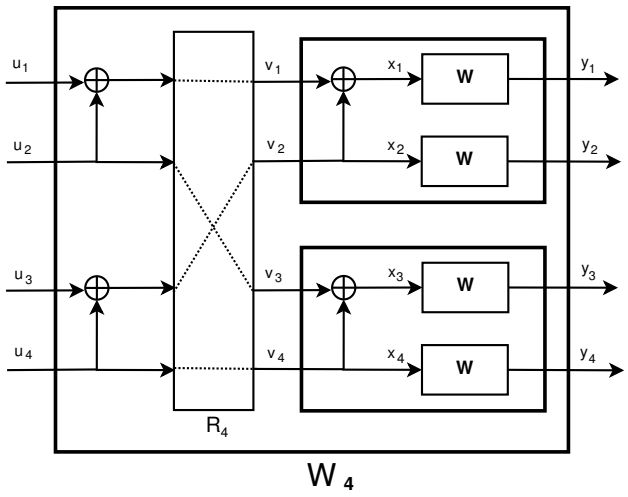


Figure 2.5: Recursive construction of W_4 from two copies of W_2.

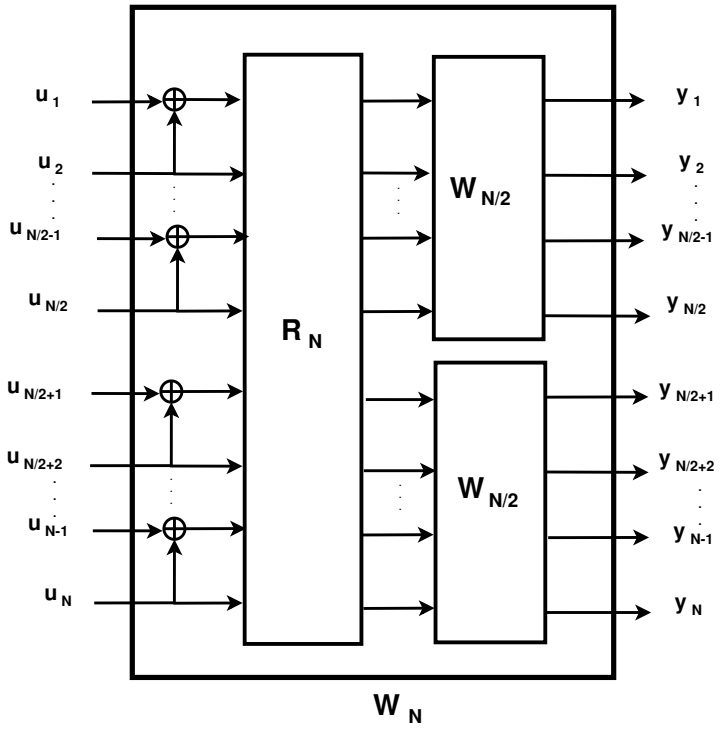


Figure 2.6: Recursive construction of W_N from two copies of $W_{\frac{N}{2}}$.

The generalized structure for N copies is illustrated in Figure (2.6).

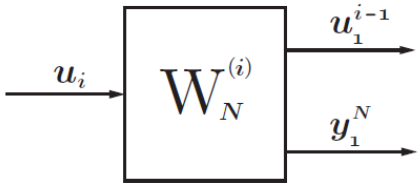


Figure 2.7: Polarized bit-channel.

The transition probability of the bit-channel $\mathrm{W}_N^{(i)}$ is defined as

$$\mathrm{W}_N^{(i)}(\mathrm{y}_1^N, \mathrm{u}_1^{i-1}|u_i) = \sum_{\mathrm{u}_{i+1}^N} \frac{1}{2^{N-1}} \mathrm{W}_N(\mathrm{y}_1^N|\mathrm{u}_1^N). \tag{2.13}$$

The following two recursive formulas manage the efficient computation of the transition probabilities of the synthesized channels.

$$W_{2N}^{(2i-1)}(y_1^{2N}, u_1^{2i-2}|u_{2i-1}) = \sum_{\mathrm{u}_{2i}} \frac{1}{2} W_N^{(i)}(y_1^N, u_{1,e}^{2i-2} \oplus u_{1,o}^{2i-2}|u_{2i-1} \oplus u_{2i}) \\ W_N^{(i)}(y_{N+1}^{2N}, u_{1,e}^{2i-2}|u_{2i}), \tag{2.14}$$

$$W_{2N}^{(2i)}(y_1^{2N}, u_1^{2i-1}|u_{2i}) = \frac{1}{2} W_N^{(i)}(y_1^N, u_{1,e}^{2i-2} \oplus u_{1,o}^{2i-2}|u_{2i-1} \oplus u_{2i}) \\ W_N^{(i)}(y_{N+1}^{2N}, u_{1,e}^{2i-2}|u_{2i}). \tag{2.15}$$

Polar codes exploit the channel polarization phenomenon by which N channels are synthesized out of combination of N independent copies of DMC channels, such that if N goes to infinity, the symmetric capacity terms $I(\mathrm{W}_N^{(i)})$ tend towards 0 or 1. This means that a fraction of the channels $\mathrm{W}_N^{(i)}$ become perfect for data transmission, while the channels in the complementary fraction become completely useless.

The capacity preserving property can be generalized for N bits, which is mathematically expressed as

$$\sum_{i=1}^{N} I(\mathrm{W}_N^{(i)}) = NI(W). \tag{2.16}$$

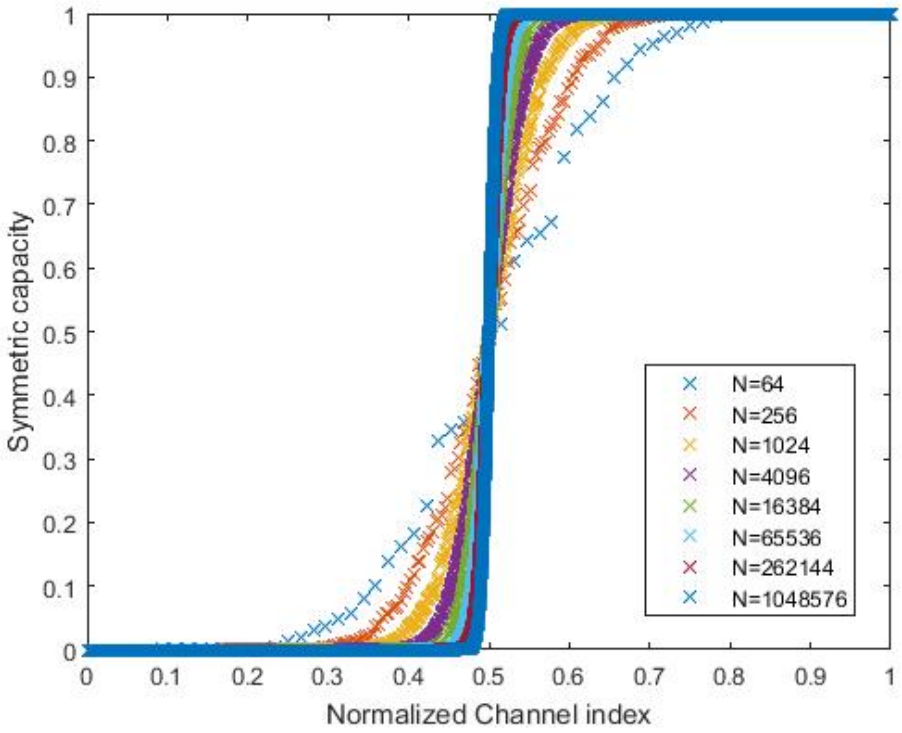


Figure 2.8: Polarization for different block lengths on BEC.

The above is true iff it applies on the basic polarization step. The capacity extremization property can also be generalized for N bits.

As the code length N approaches infinity, the polarized channels go to two extremes: the bit-channels with capacity almost 0 or the ones with almost 1. This is the reason why bigger block lengths are preferred when choosing the parameters of polar codes.

In Figure (2.8), we observe the effect of channel polarization. As expected, almost half of the channels are perfect and the other half are useless. The fraction of the channels that become perfect for transmission is equal to the capacity of the channel. Since the upgraded channels have higher capacities or lower error probabilities than the degraded (noisy channels), the channel polarization phenomenon introduces a new philosophy for channel coding, selecting the noiseless channels for information-bits transmission. The selec-

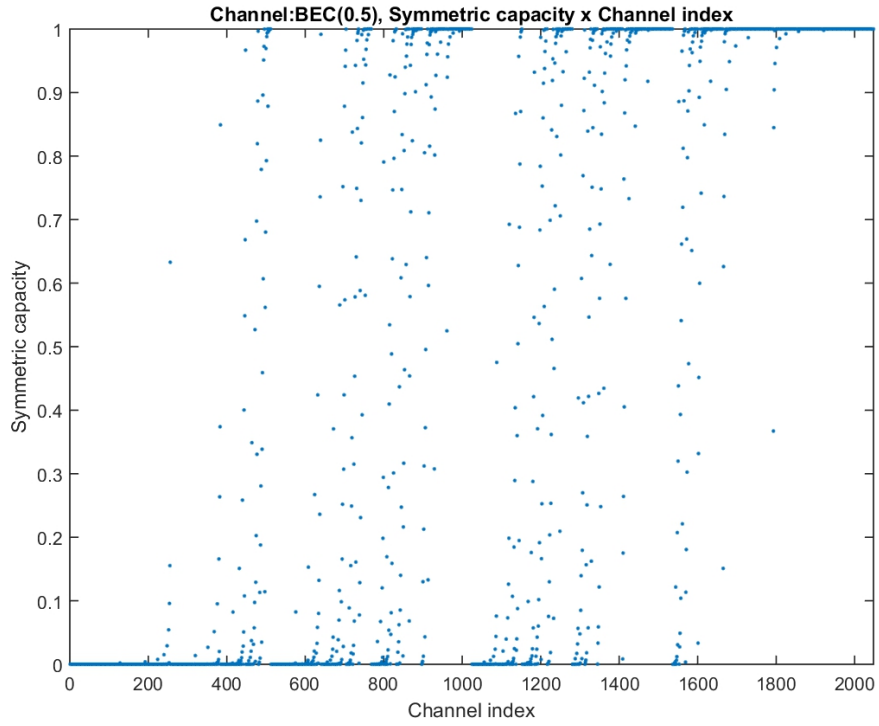


Figure 2.9: $\mathrm{I}(\mathrm{W}_N^{(i)})$ vs. i=1,...,N=2048 for BEC($\frac{1}{2}$).

tion of good channels is explained in section (5).

The polarization effect is illustrated in Figure (2.9) for the case W is a BEC with erasure probability $\epsilon = 0.5$. The figure shows that $\mathrm{I}(\mathrm{W}_N^{(i)})$ tends to be near 1 for small i and 0 near for large i. The numbers $\mathrm{I}(\mathrm{W}_N^{(i)})$ have been computed using the following recursive relations.

$$I(\mathrm{W}_N^{(2i-1)}) = I(\mathrm{W}_{\frac{N}{2}}^{(i)})^2, \tag{2.17}$$

$$I(\mathrm{W}_N^{(2i)}) = 2I(\mathrm{W}_{\frac{N}{2}}^{(i)}) - I(\mathrm{W}_{\frac{N}{2}}^{(i)})^2 \text{with } I(\mathrm{W}_1^{(1)}) = 1 - \epsilon. \tag{2.18}$$

By recursively applying the polarization transformation over the resulting channels, the result is that "the good ones get better and the bad ones get worse." Regarding the Bhattacharyya parameter of the synthesized channels, for any B-DMC W, $N = 2^n, n \geq 0, 1 \leq i \leq N$, the relations (2.10) and (2.11)

can be generalized. Thus,

$$Z(\mathrm{W}_{2N}^{(2i)}) = Z(\mathrm{W}_{N}^{(i)})^2, \tag{2.19}$$

$$Z(\mathrm{W}_{2N}^{(2i-1)}) \leq 2Z(\mathrm{W}_{N}^{(i)}) - Z(\mathrm{W}_{N}^{(i)})^2. \tag{2.20}$$

Chapter 3

Encoding of Polar Codes

The implementation of polar encoder can be accomplished in two ways. The first way is the matrix multiplication, which is of quadratic complexity. On the other hand, butterfly recursive encoding reduces the complexity to $\mathcal{O}(NlogN)$.

3.1 Matrix multiplication

Having shown the basic step of polarization and the general channel trnasformation, we can now define the recursion that constructs the generator matrix of polar codes. We define the Arikan kernel matrix as F=$\begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}$. The n-th Kronecker product of F is denoted $F^{\otimes n}$ and is defined recursively as

$$F^{\otimes n}=\begin{bmatrix} F^{\otimes n-1} & 0 \\ F^{\otimes n-1} & F^{\otimes n-1} \end{bmatrix}, \text{ where } F^{\otimes 1}=\text{F}.$$

For example $F^{\otimes 2}$ can be calculated in the following way

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

Polar codes in the original format are not systematic. In other words, the information bits do not appear as part of the codeword transparently. The generator matrix for polar codes of block length $N = 2^n$ is $G_N = B_N F^{\otimes n}$, where B is the bit-reversal permutation matrix. If $v_1^N = u_1^N B_N$, then $v_{b_1,b_2,\ldots,b_n} = u_{b_n,b_{n-1},\ldots,b_1}$. Polar encoder splits the source word into two parts: u=(u_A, u_{A^c})

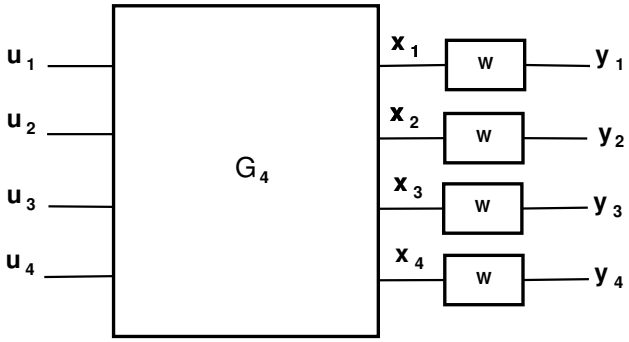


Figure 3.1: Generator matrix for N=4.

so that the first part consists of user data that is free to change in each round of transmission, while the second part consists of digits that are frozen at the beginning of the session and made known to the decoder. The codeword produced by the polar encoder is $\bar{x}=\bar{u}_A G_A \oplus \bar{u}_{A^c} G_{A^c}$ where G_A, G_{A^c} submatrices of G_N consisting of rows with indices in A and A_c, respectively. For example, let $G_4 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix}$ and the $(4, 2, \{2, 4\}, (0, 0))$ polar code. On the 2nd level of recursion $(n = 2, N = 4)$, we use two independent copies of W_2 to create the vector channel W_4 Figure (2.5). The codeword produced by the polar encoder is: $x_1^4=u_1^4 G_4=(u_2, u_4) \begin{bmatrix} 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix} + (0, 0) \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \end{bmatrix}$.

For systematic polar codes, we focus on a generator matrix without the permutation matrix, namely $G_N = F^{\otimes n}$. In a non-systematic codeword x, information bits do not appear while in a systematic source word is mapped to codeword, such that the message bits are explicitly visible. It was shown by Arikan that the set of K indices in a codeword x, where the message bits appear explicitly can be chosen equal to the set of information bit indices I. There are multiple advantages to systematic encoding such as easier information extraction and better bit-error rate (BER) performance.

Matrix multiplication of G with an information vector is easy for small block

lengths but not convenient for bigger block lengths. The process has a complexity of $\mathcal{O}(N^2)$.

3.2 Butterfly circuit model

The first step is to construct an intermediate message $\begin{bmatrix} u_1 & u_2 & . & . & . & u_N \end{bmatrix}$ of length N from a length-K source message. The general form of the recursion is shown in Figure (2.6), where two independent copies of $W_{\frac{N}{2}}$ are combined to produce channel W_N.

The operator R_N is a permutation, known as the reverse shuffle operation. Reverse shuffling distincts the odd and even indexed elements and passes them to separate channels. Odd indexed signals become input to the first copy of $W_{\frac{N}{2}}$ and even indexed to the second. This permutation can also be interprended as re-ordering the binary expression of i-1 where i represents the input index. By re-ordering we mean that it cyclically rotates the bit-indexes by one place to the left. If $v_1^N = u_1^N R_N$, then $v_{b_1,b_2,\ldots,b_n} = u_{b_2,b_n,\ldots,b_1}$. For $n = 3$, the reverse shuffle operation, $v_1^8 = u_1^8 R_8$ transforms the vector $u_1^8 = (u_{000}, u_{001}, u_{010}, u_{011}, u_{100}, u_{101}, u_{110}, u_{111})$ to $v_1^8 = (u_{000}, u_{010}, u_{100}, u_{110}, u_{001}, u_{011}, u_{101}, u_{111})$ which means that we go from $u_1^8 = (u_1, u_2, u_3, u_4, u_5, u_6, u_7, u_8)$ to $v_1^8 = (u_1, u_3, u_5, u_7, u_2, u_4, u_6, u_8)$. Note that R_N and B_N are not equal. B_8 happens to have the same effect on u_1^8 as R_8 does, but this does not happen for bigger N.

In terms of complexity, let us assume that the computational complexity of a mod-2 addition process is unitary (1 unit) and the complexity of the reverse shuffle operation R_N is N units of time, the complexity is derived with the help of Master Theorem is

$$\begin{aligned} &\mathcal{T}(N) = \frac{N}{2} + \mathcal{O}(N) + 2\mathcal{T}(\frac{N}{2}) \Rightarrow \\ &\mathcal{T}(N) = \mathcal{O}(N \log_2 N) \end{aligned} \tag{3.1}$$

Chapter 4

Decoding of Polar Codes

4.1 Successive cancellation

Arikan proposed successive cancellation (SC) decoding in order to achieve capacity with low complexity. The information bits are decoded sequentially in the increasing order of their indices. The decoder's task is to generate an estimate $\hat{u}_1^N$ of u_1^N, where u_1^N denotes the input data sequence, given knowledge of A, u_{A_c} and y_1^N. If u_i is a frozen bit, the decoder will set $\hat{u}_i$ to zero. If u_i is an information bit, the decoder decides on ith bit ($1 \leq i \leq N$) computing the following likelihood ratio (LR) after estimating all the previous bits u_1^{i-1}. The SC decoder follows a depth first approach, meaning that it examines only one path from the root node to the leaf nodes assuming a tree structure.

$$L_N^{(i)}(y_1^N, \hat{u}_1^{i-1}) = \frac{W_N^{(i)}(y_1^N, \hat{u}_1^{i-1}|0)}{W_N^{(i)}(y_1^N, \hat{u}_1^{i-1}|1)} \tag{4.1}$$

and generates its decision as

$$\hat{u}_i = \begin{cases} 0, & \text{if } L_N^{(i)}(y_1^N, \hat{u}_1^{i-1}) \geq 1 \\ 1, & \text{otherwise} \end{cases} \tag{4.2}$$

In general, the notation for LRs is $L_i^{(j)}$ where i indicates the stage and j the level, with $1 \leq i \leq N$ and $1 \leq j \leq n$.

4.1.1 Naive implementation

The complexity of SC decoding is determined by the complexity of the computing LRs. The recursive relations for computing the LRs are

$$L_N^{(2i-1)}(y_1^N, \hat{u}_1^{2i-2}) = \frac{1 + L_{\frac{N}{2}}^{(i)}(y_1^{\frac{N}{2}}, \hat{u}_{1,e}^{2i-2} \oplus \hat{u}_{1,o}^{2i-2}) L_{\frac{N}{2}}^{(i)}(y_{\frac{N}{2}+1}^N, \hat{u}_{1,e}^{2i-2})}{L_{\frac{N}{2}}^{(i)}(y_1^{\frac{N}{2}}, \hat{u}_{1,e}^{2i-2} \oplus \hat{u}_{1,o}^{2i-2}) + L_{\frac{N}{2}}^{(i)}(y_{\frac{N}{2}+1}^N, \hat{u}_{1,e}^{2i-2})} \tag{4.3}$$

$$L_N^{(2i)}(y_1^N, \hat{u}_1^{2i-1}) = L_{\frac{N}{2}}^{(i)}(y_1^{\frac{N}{2}}, \hat{u}_{1,e}^{2i-2} \oplus \hat{u}_{1,o}^{2i-2})^{1-2\hat{u}_{2i-1}} L_{\frac{N}{2}}^{(i)}(y_{\frac{N}{2}+1}^N, \hat{u}_{1,e}^{2i-2}) \tag{4.4}$$

The relations (4.3), (4.4) can be respectively rewritten in simplifed way as:

$$L = \frac{1 + L_1 L_2}{L_1 + L_2} \text{ and} \tag{4.5}$$

$$L = L_1 L_2 \text{ in case } \hat{u}_{2i-1} = 0 \text{ or} \tag{4.6}$$

$$L = \frac{L_1}{L_2} \text{ otherwise .} \tag{4.7}$$

A decision table is demonstrated for border cases like $\frac{0}{0}$, $\frac{\infty}{\infty}$ in (4.5).

L_1 \ L_2	0	1	∞
0	∞		0
1			1
∞	0	1	∞

On the Nth level, each calculation of an LR at this length N requires the calculation of two LRs at length $\frac{N}{2}$. Each calculation of an LR at length $\frac{N}{2}$ requires the calculation of two LRs at length $\frac{N}{4}$ and so on, until we reach length 1, at which point the recursion is terminated and the LRs have the form $L_1^{(1)}(y_i) = \frac{W(y_i|0)}{W(y_i|1)}$.

From the above equations, it can be seen that each calculation of LR $\{L_N^{(i)}(y_1^N, \hat{u}_1^{i-1}) : (1 \leq i \leq N)\}$ is reduced to the calculation of two LRs at

Algorithm 1 A high-level description of the SC decoder

Input: *The received vector* y
Output: *a decoded codeword* $\hat{c}$

for $\phi = 0, 1, ..., N-1$ **do**
  calculate $L_N^{(\phi)}(y_1^N, \hat{u}_1^{\phi-1})$
  if u_ϕ *is frozen* **then**
    set $\hat{u}_\phi$ *to the frozen value of* u_ϕ
  else
    if $L_N^{(\phi)}(y_1^N, \hat{u}_1^{\phi-1}) \geq 1$ **then**
      set $\hat{u}_\phi \leftarrow 0$
    else
      set $\hat{u}_\phi \leftarrow 1$
    end if
  end if
end for
return *the estimated source word* $\hat{u}_A$

length $\frac{N}{2}$, thus overall $\mathcal{O}(N)$ complexity. Hence the calculation of all N LRs requires $\mathcal{O}(N^2)$ time complexity.

4.1.2 Smart implementation

A refined successive cancellation decoder was proposed to decrease the redundant calculations in SC decoding without affecting the error performance. To see where the computational saving will come from, we notice that each LR value in the pair $(L_N^{(2i-1)}(y_1^N, \hat{u}_1^{2i-2}), L_N^{(2i)}(y_1^N, \hat{u}_1^{2i-1}))$ is assembled from the same pair of LRs $(L_{\frac{N}{2}}^{(i)}(y_1^{\frac{N}{2}}, \hat{u}_{1,e}^{2i-2} \oplus \hat{u}_{1,o}^{2i-2}), L_{\frac{N}{2}}^{(i)}(y_{\frac{N}{2}+1}^N, \hat{u}_{1,e}^{2i-2}))$. Let us split the N LRs at length $\frac{N}{2}$ into two classes, namely,

$$\{L_{\frac{N}{2}}^{(i)}(y_1^{\frac{N}{2}}, \hat{u}_{1,e}^{2i-2} \oplus \hat{u}_{1,o}^{2i-2}) : 1 \leq i \leq \frac{N}{2}\} \tag{4.8}$$

$$\{L_{\frac{N}{2}}^{(i)}(y_{\frac{N}{2}+1}^N, \hat{u}_{1,e}^{2i-2}) : 1 \leq i \leq \frac{N}{2}\} \tag{4.9}$$

Each class generates a set of $\frac{N}{2}$ LR calculation requests at length $\frac{N}{4}$, for a total of N requests.

If we store the already calculated LRs, the calculation of all N LRs at length N requires the calculation of N LRs at length $\frac{N}{2}$ and not 2N calculations as in the previous algorithm. The total number of distinct LRs needed to be calculated and stored with this algorithm is exactly $N(1+logN)$. Thus, both the computational and memory complexity of the SC decoder is $\mathcal{O}(N \log N)$.

Below, we describe the decoding procedure for $N = 4$. The procedure is shown in Figure (4.1), where $N(1 + logN) = 12$ nodes are illustrated. Each node corresponds to an LR request. To begin with, the decoder activates node 1 to decide the value of $\hat{u_1}$. Node 1 needs the LLRs at nodes 5 and 7, $L_1^{(1)}, L_3^{(1)}$, so it activates the two branches on its right. Then, nodes 5 and 7 need the LR values of nodes 9, 10, 11, 12. Therefore, LRs at node 5 and 7 are computed using formula (4.3) and from them, $L_1^{(2)}$ is calculated using the same formula which makes the decoder capable to decide the value of $\hat{u_1}$. For computing , $L_1^{(2)}$ at node 2, nodes 5 and 7 are activated and the formula (4.4) is used based on the previous values already computed. To estimate $\hat{u}_3$, we activate nodes 6 and 8.

4.1.3 Performance on the BEC and BSC

Let us consider how successive cancellation decoder behaves in practice for BEC and BSC.

The following two figures are presented to demonstrate the performance of polar codes over BEC. In Figure (4.2), we consider transmissions of R=$\frac{1}{2}$ over the BEC. The block length N is set to 32, 64, 128, 256, 512, and 1024. We plot the bit error rate as a function of the erasure probability ϵ. In Figure (4.3), we consider BEC($\frac{1}{2}$), the same variety of block lengths as before and plot the BER as a function of rate R. We observe that the BER performance improves as the block length grows larger. This is justified by the fact that greater level of channel polarization is accomplished when the block length increases to infinity.

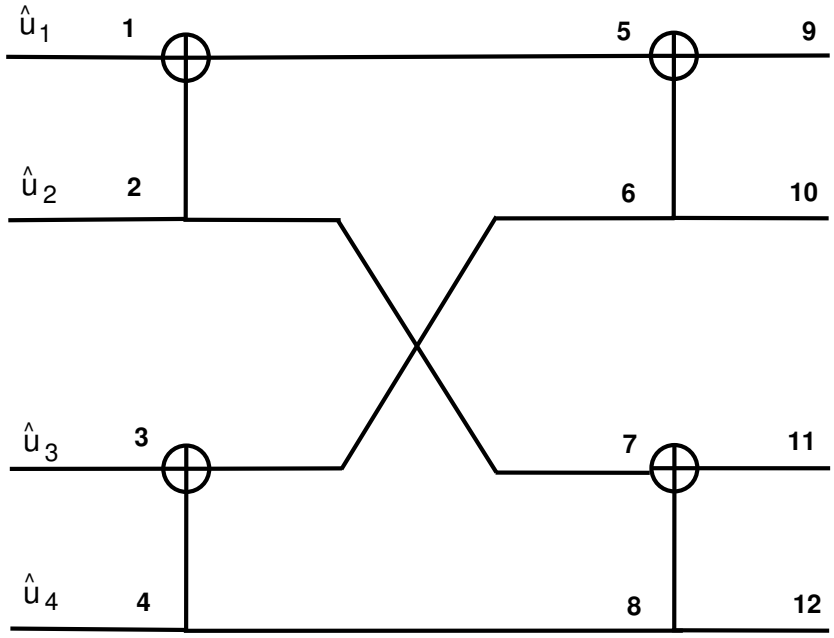


Figure 4.1: Successive cancellation decoder for $N = 4$.

4.1.4 Inferiority of SC decoder

The SC decoding method leads to error propagation which affects the bit error rate of polar codes. Once an unfrozen bit is set, there is "no going back". A bit that was set at step i can not be changed at step $j > i$. In SC decoding, incorrect bit decisions can be caused by channel noise or by error propagation due to previous erroneous bit decisions. The first incorrect decision is always caused by the channel noise since there are no previous errors.

The performance of SC decoding at finite blocklengths is inferior to other modern codes, such as LDPC codes. The reason is that for finite block lengths there exists a broad "grey zone" of channels whose capacities are far from being polarized.

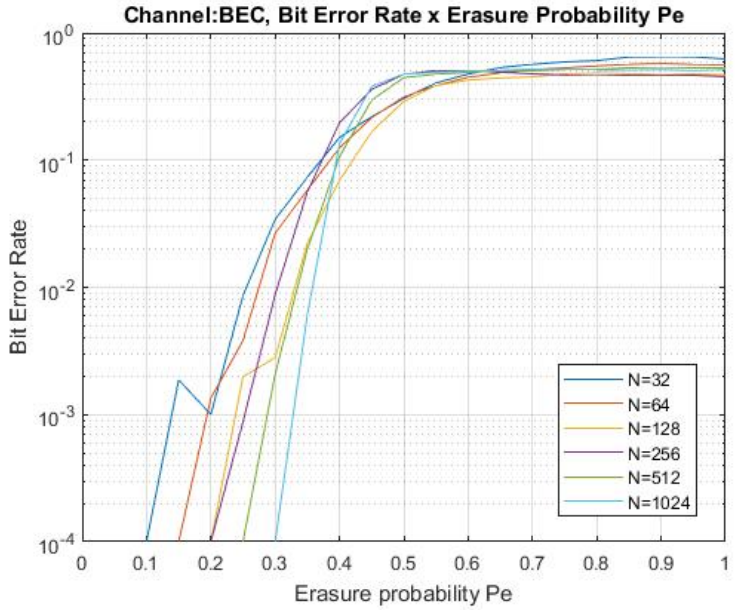


Figure 4.2: BER vs erasure probability for polar code of R=$\frac{1}{2}$ and $N = 32$, 64, 128, 256, 512, and 1024 on the BEC.

4.2 List decoding of polar codes

To improve the finite blocklength performance, SC list (SCL) decoder was introduced recently. The decoding algorithm uses SC as the underlying decoder, but improves its performance by exploring multiple paths on a decision tree simultaneously, with each path resulting in one candidate codeword. The computational and memory complexities of SC list decoding are though much higher than simple SC decoder.

At each decoding step $i \in A$, instead of fixing a decision on the unfrozen bit u_i, two decoding paths corresponding to either possible value of u_i are created and decoding is continued in two parallel decoding threads. This means that SCL uses a breadth-first approach by searching more than one paths in each level of the tree. Each split doubles the number of paths to be examined. In order to avoid the exponential growth of the number of decoding paths, a pruning procedure is used to discard all but the L best paths. The pruning criterion will be to keep the most likely paths. Finally, the

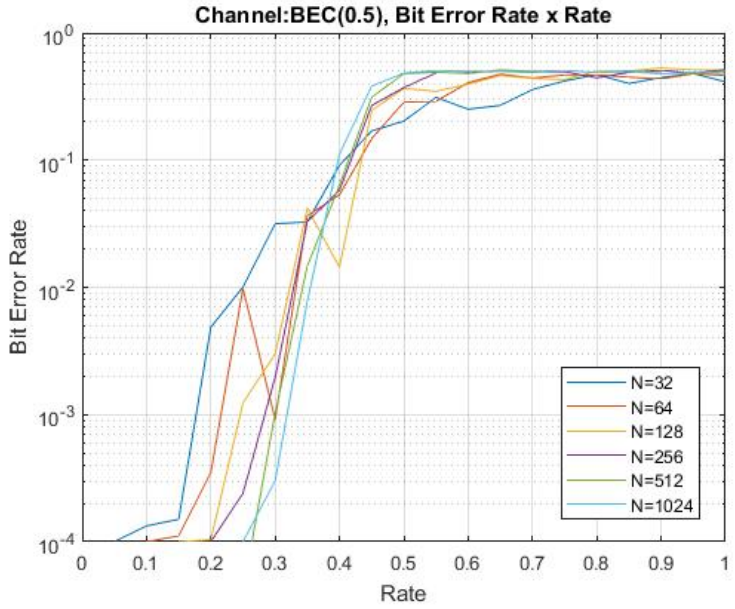


Figure 4.3: BER vs Rate for polar code of ϵ=$\frac{1}{2}$ and N = 32, 64, 128, 256, 512, and 1024 on the BEC.

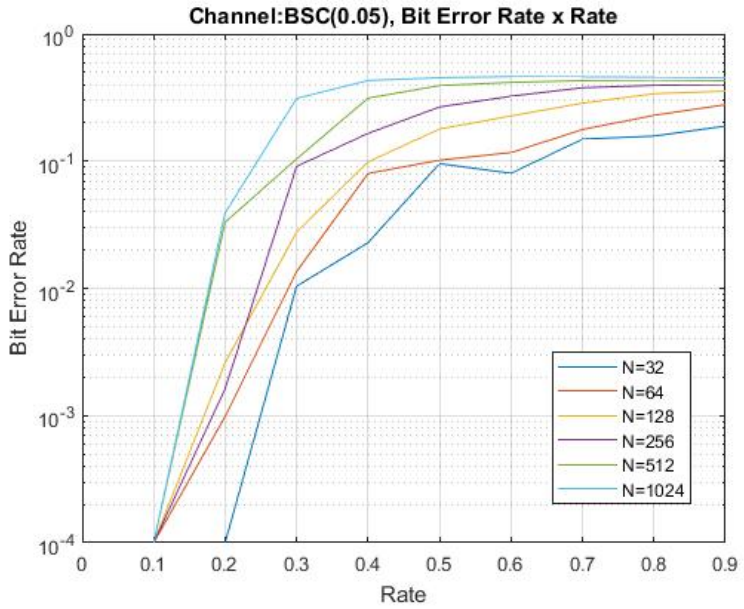
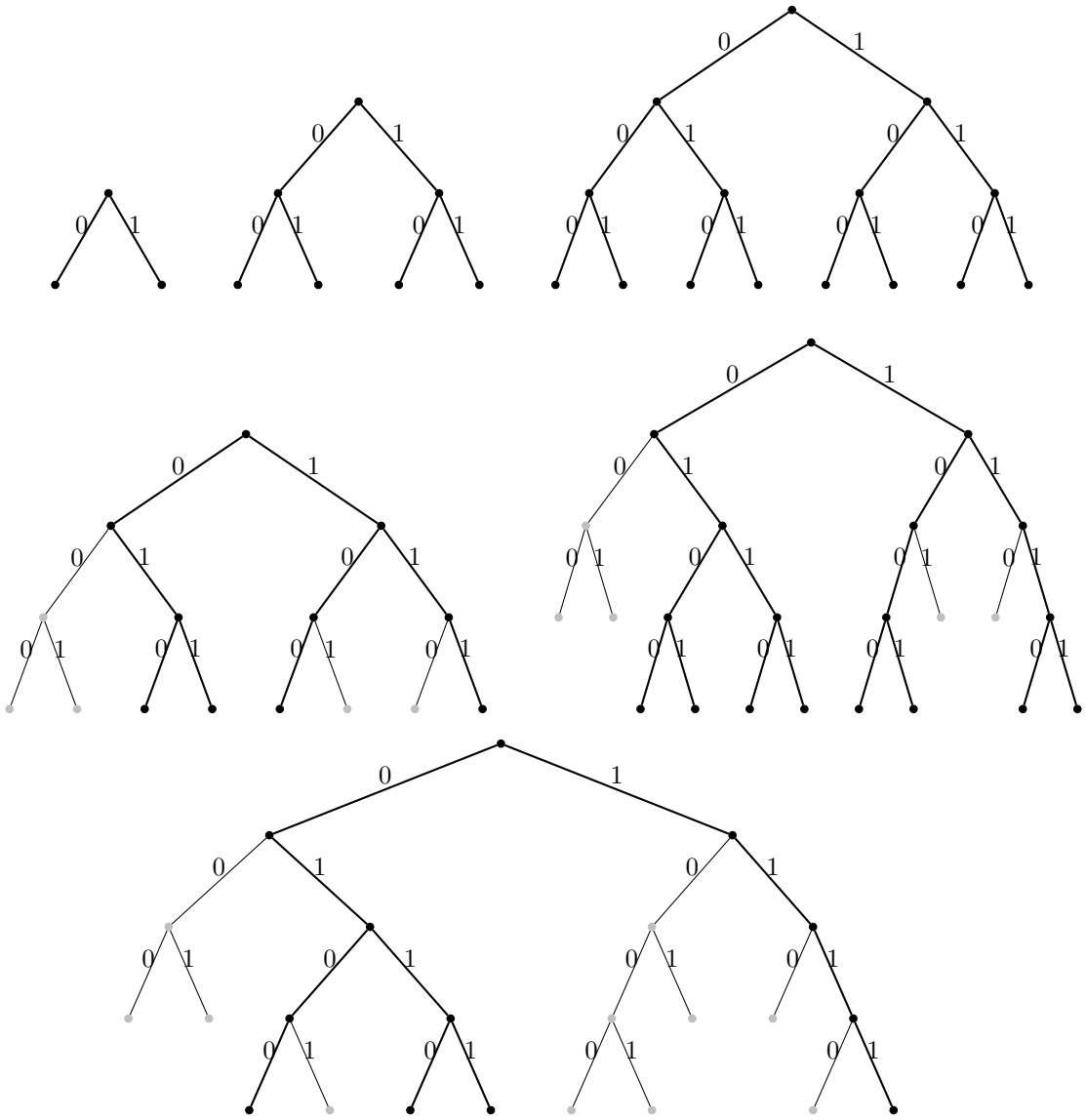


Figure 4.4: BER vs Rate for polar code of ρ=0.05 and N = 32, 64, 128, 256, 512, and 1024 on the BSC.

decoder will end up with a list of L candidates for u_A out of which the most likely one is declared as the final estimate $\hat{u}_A$. The estimated codeword is declared by the binary labels u_1^N that correspond to the edges $\{e_1, e_2, ..., e_N\}$ of this chosen path. If u_i is a frozen bit, there will be no splitting. Instead, $\hat{u}_i$ is set to the fixed frozen value.

In a naive implementation, at each split the data structures used by the parent path are duplicated, with one copy given to the first fork and the other to the second. Since the number of splits is $\Omega(LN)$, and since the size of the data structures used by each path is $\Omega(N)$, the copying operation alone would take time $\Omega(LN^2)$, which is practical only for codes with small blocklength. To avoid wasting a lot of time on copy operations at each stage, we could perform only a small number of copy operations with the following tricks. At each split, flag the corresponding variables as belonging to both paths and give each path a unique variable (make a copy) only before that variable will be written to. If a path is killed, deflag its corresponding variables. In the end, the result is a running time of $\mathcal{O}(LNlog_2N)$ with $\mathcal{O}(LN)$ memory requirements.

To understand the concept of SCL decoding, let us examine the case of polar codes of block length $N = 4, K = 4$ (all bits are unfrozen), and $L = 4$. The decoding algorithm starts and the first bit can be either 0 or 1 ($\hat{u}_0 = 0$ or $\hat{u}_0 = 1$). In the next step the decision over the second bit is split again into 0 or 1. The possible words are $\hat{u}_0^1 = 00, \hat{u}_0^1 = 01, \hat{u}_0^1 = 10, \hat{u}_0^1 = 11$ but the number of paths is not greater than $L = 4$ so we don't need to prune, yet. In the following step we examine all possible values of $\hat{u}_0^2$ but now the paths are 8 so we must keep track of only the $L = 4$ most likely paths. Only the words $\hat{u}_0^2 = 010, \hat{u}_0^2 = 011, \hat{u}_0^2 = 100, \hat{u}_0^2 = 111$ are kept. The proccess continues for the fourth bit at which point the possible words are 8 which is too much so the algorithm prunes keeping only the best $L = 4$ paths. The decoding algorithm terminates and we obtain the codewords $\hat{u}_0^3 = 0100, \hat{u}_0^3 = 0110, \hat{u}_0^3 = 0111, \hat{u}_0^3 = 1111$.



Regarding the implementation of SCL, the set of candidate paths that correspond to the level i ($\mathbb{L}^{(i)}$) are kept in a list structure and updated at every level. The first step is the path initialization. The initial list contains a null path ($\mathbb{L}^{0)} = \emptyset$). The next step involves expanding all the existing paths with one bit, once with a 0 once with a 1, doubling the total number of the new candidate paths. Each corresponding path metric is updated in accordance with equations (2.14), (2.15). In the following step, half of the candidate paths, the ones with the lowest probability metric are deleted, so that ex-

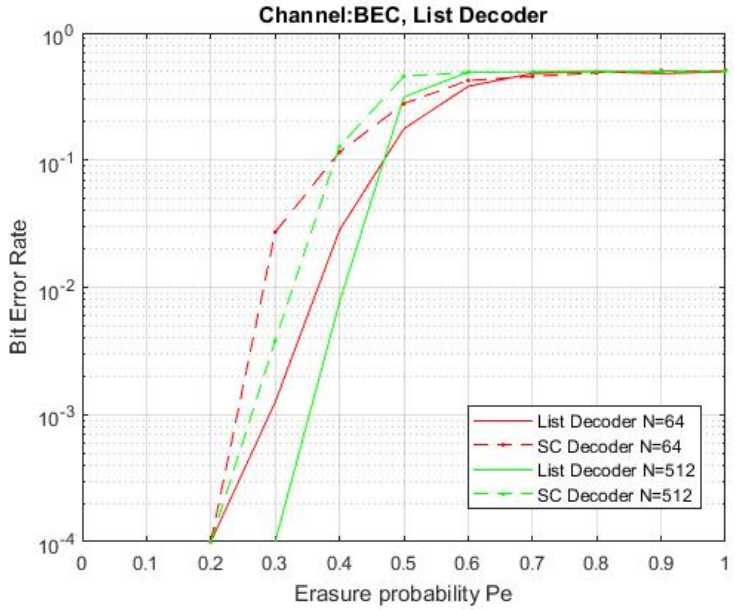


Figure 4.5: SCL vs SC for small ($N = 64$) and moderate ($N = 512$) size of block length

actly L paths are preserved. If the candidate paths are no more than L, then no path is deleted. The last 2 steps are reapeted until length N is reached, when the path with the largest probability metric is selected.

In Figure (4.5), the improvement in the bit error rate performance by using SCL decoder instead of SC decoder is depicted in case of small and moderate block lengths. The SCL decoder has a parameter L, called the list size. If $L = 1$ the SCL decoder is equal to the simple SC decoder while for $L = 2^N$, we have the case of the ML decoder. Generally speaking, larger values of mean lower error rates but longer running times. This is illustrated in Figure (4.6).

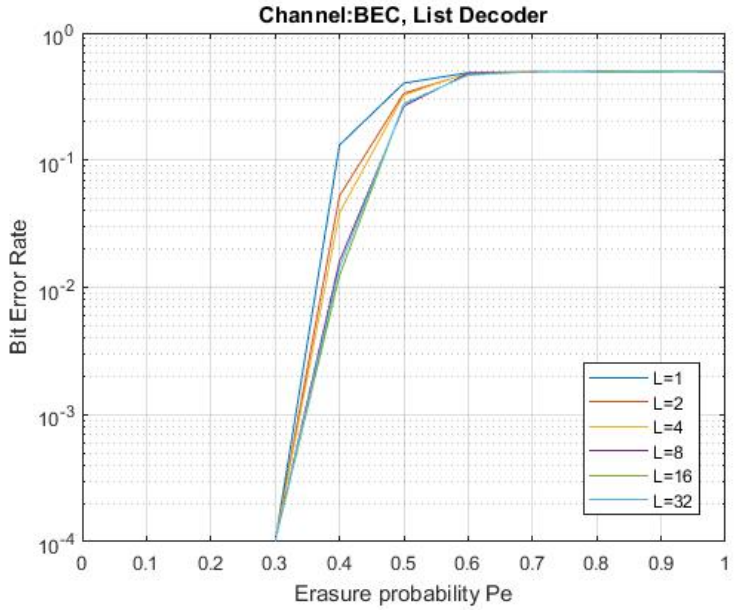


Figure 4.6: SCL for different values of list size, L.

4.2.1 List decoder with CRC check

Polar codes concatenated with a high rate cyclic redundancy check (CRC) code is an effective approach that can further enhance the performance of the polar codes, making them a strong competitor to Turbo and LDPC codes by applying a SCL decoder with sufficiently large list size. Using a CRC as the primary criterion for selecting the final decoder output, increased the error-correction performance significantly. The idea of CRC check is the following: Let there be $k + r$ unfrozen bits. Of these, we use the first k bits to encode information and the last r unfrozen bits to encode the CRC value of the first k bits. In the end, we pick the most probable codeword on the list with correct CRC.

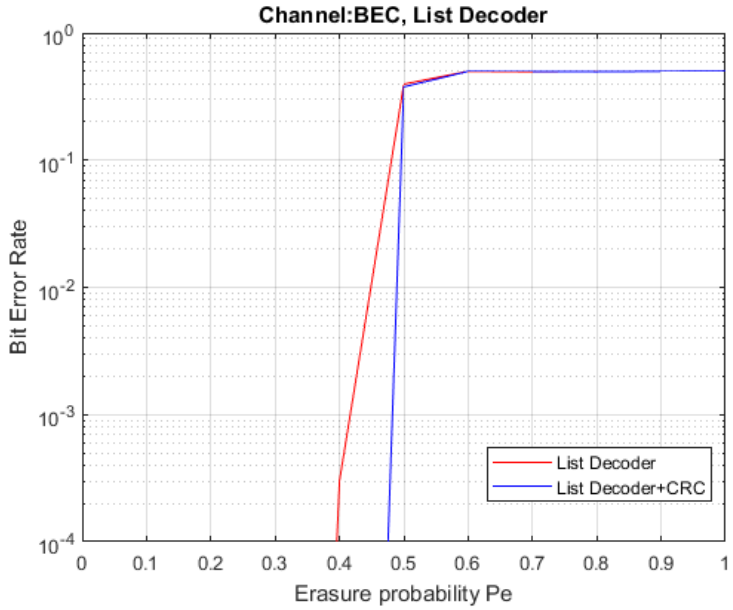


Figure 4.7: SCL vs SCL with CRC with $N = 2048$, $L = 32$, $CRC = 16$.

Chapter 5

Construction of Polar Codes

The consruction of polar codes is one of the important concerns in using polar codes in practice. The input to a polar code construction algorithm is a triple (W, N, K) where W is the B−DMC on which the code will be used, N is the code block length, and K is the dimensionality of the code. The output of the algorithm is an information set $\mathcal{A} \subset \{1, 2..., N\}$ of size K, such that $\sum_{i \in \mathcal{A}} I(\mathrm{W}_N^{(i)})$ is as large as possible. In other words, designing polar codes is equivalent to finding the set of good indices among the N polarized channels that guarantee the maximum reliability. Good channels will have $I(W)$ close to 1 and $Z(W)$ close to 0. In the way those two parameters are defined, one would expect that $I(W)$ goes to 1 iff $Z(W)$ goes to 0 and vice versa.

5.1 Construction over BEC

In BEC, the reliability of the bit channels is defined by the recursive relations (2.17) and (2.18). The capacities are then sorted and the indices with the highest reliabilities are chosen as elements of the set $\mathcal{A}$.

For BEC the parameters $\mathrm{I}(\mathrm{W}_N^{(i)})$ can all be calculated in time $O(N)$ thanks to the recursive formulas.

5.2 Construction over BSC

The recursive relations (2.17) and (2.18) are valid only for BECs. No efficient algorithm is known for calculation of for a general B−DMC W.

5.2.1 Bhattacharyya parameter based channel construction

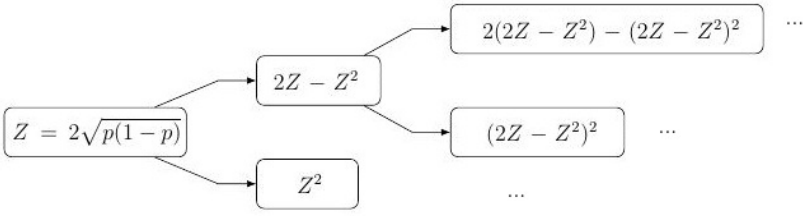


Figure 5.1: Bhattacharyya parameter based channel construction.

In the initial paper, Arikan utilizes the Bhattacharyya parameter $Z(W) = \sum_{y \in \mathcal{Y}} \sqrt{W(y|0)W(y|1)}$ as another measurement of channel quality. Instead of computing the Bhattacharyya parameters directly, a proccess that whould require high computational cost, upper bounds on the Bhattacharyya parameters of bit-channels are calculated in order to estimate the bit channels according to the relations (2.19) and (2.20). The process begins with the value of Bhattacharyya parameter of BSC ($Z_{\mathrm{BSC}(\rho)} = 2\sqrt{p(1-p)}$). The Bhattacharyya parameter of the original channel W is used to generate two values : $Z = 2(Z_{\mathrm{BSC}(\rho)}) - (Z_{\mathrm{BSC}(\rho)})^2$ and $Z = (Z_{\mathrm{BSC}(\rho)})^2$. This step is repeated recursively until N numbers are generated. In the end, the K leaves with the smallest Bhattacharyya parameters are selected and their index is returned as output. This proccess is illustrated in Figure (5.1).
Due to its simplicity, this construction has been widely used, and produced good polar codes.

5.2.2 Proposed algorithm

In this algorithm, we calculate the BER of bit-channels rather than their Bhattacharyya parameters. We consider consecutive transmissions of $K-$ length zero-vectors and we calculate the BER of each synthesized channel after the repetitions. The algorithm utilizes the fact that the vast majority of the reliable channels tend to be concentrated in the last bit-channels and

only a small fraction of the first bit-channels will qualify. In particular, three BER thresholds are set, one for small values of ρ, one for medium and one for large values. The threshold for small values of ρ is lower and for larger values, higher.

On the first pass, the first K channels are tested. Zero-vectors are sent over the first K channels and the BER of each synthesized channel is calculated, after the repetitions. It is expected that almost none of the first channels will reach the threshold. If s channels qualify, the channel index and BER are stored on matrix candidates. A variable named checked keeps track of the index of channels that have been examined so far.

On the next passes, the algorithm sends zero-bits over the channels currently present on candidates and assuming the size of candidates is s, the zero-bits are also sent over the next $K - s$ channels. After transmitted repeatedly, the examined channels, will be evaluated according to their BERs. To this point, $checked + s$ first channels will have been evaluated.

The process is repeated until all bit channels are evaluated ($checked = N$).

The capacity of matrix candidates is K. If the matrix is filled up and not all bit-channels are checked, the matrix will be sorted according to BERs. The tuple with the smallest value of BER will be removed and kept on a vector called “candidate on air”. Its place will be taken by the next channel that is to be evaluated. If the bit-channel has better BER performance, then it will replace "candidate on air" permanently. If it does not, the “candidate on air” will be restored on candidates.

There is also the possibility that after examining all the bit channels, less than K channels will be found. To mend the situation, at each evaluation $\frac{K}{2}$ channels that have not qualified on candidates are stored and updated on matrix reservoir according to BERs. If the end is reached and the matrix candidates is not filled up, then candidates will be completed with channels

with the smallest BERs from matrix reservoir.

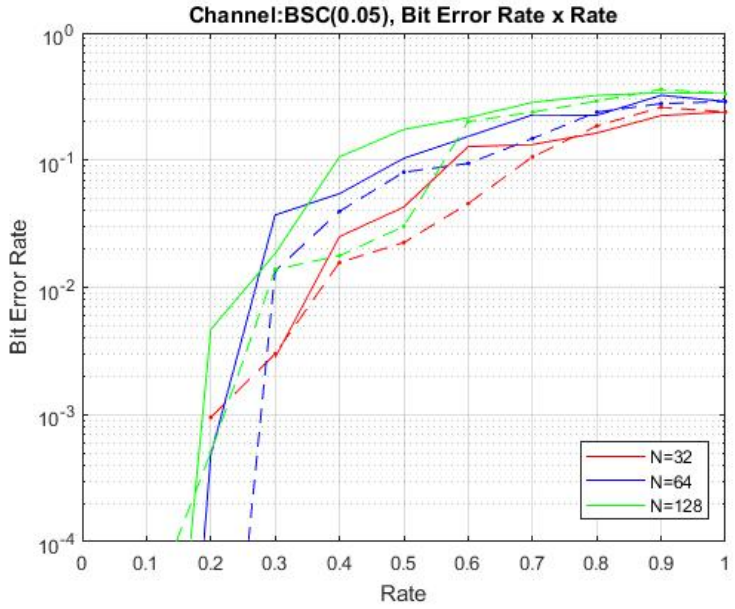


Figure 5.2: Bhattacharyya parameter based construction method vs the proposed code construction algorithm for $N = 32, N = 64, N = 128$.

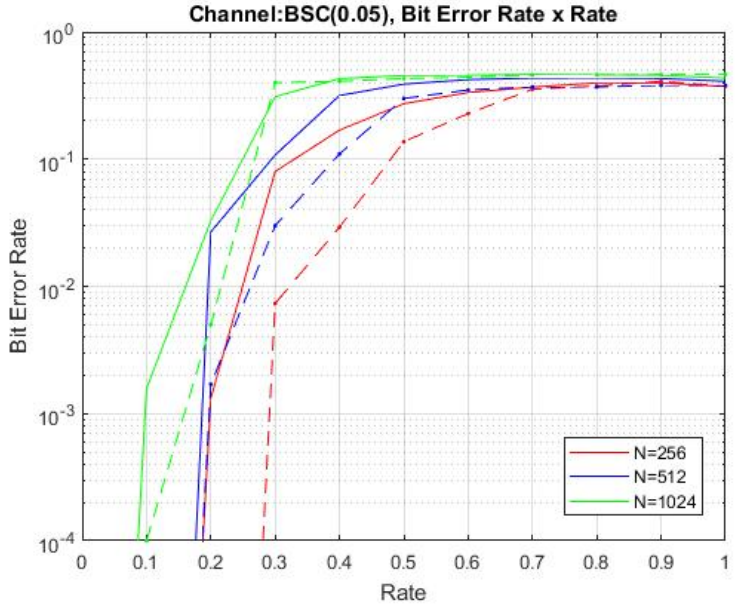


Figure 5.3: Bhattacharyya parameter based construction method vs the proposed code construction algorithm for $N = 256, N = 512, N = 1024$.

5.3 Other Construction methods

Mori and Tanaka attempted an efficient construction of polar codes in the general case utilizing density evolutions, a method that requires large memory and high complexity increasing with the code length.

A straightforward construction of polar codes is unmanageable because the resulting bit-channels have an output alphabet that grows exponentially with the code length. The Tal and Vardy construction method approximates a bit-channel with an intractable large alphabet by a better and a worse channel with smaller alphabet size. The polar code is constructed according to the worse channel and is compared with the better channel to measure the distance from the optimal channel.

Chapter 6

Brief review, recent results and current work

6.1 Brief review

In this thesis the fundamental concepts of polar coding are showcased, including channel polarization, encoding scheme and decoding algorithms such as SC and SCL. The crucial concern to improve the finite length performance of polar codes under SC decoding has been addressed, while at the same time keeping computation and memory complexity as low as possible. Last but not least, results for polar codes with two different construction methods are presented.

6.2 Recent results and current work

Recent progress has been made by Gross and Sarkis (MacGill University), who managed to further improve the performance using systematic polar codes and SCL with CRC.

Polar codes have a large variety of applications, including the construction of a coding scheme that achieves the secrecy capacity for a wide range of wire-tap channels, which was initiated by Mahdavifar and Vardy. Lossy source compression of a binary symmetric source using polar codes has been studied by Korada and Urbanke. In addition, Gross and Sarkis have investigated the suitability of polar codes to data storage applications focusing on error-correction performance and throughput.

Bibliography

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

[2] Erdal Arikan, Systematic Polar Coding. IEEE Communications Letters, 15(8): 860 - 862, August 2011.

[3] Ido Tal and Alexander Vardy, List decoding of polar codes. Information Theory Proceedings (ISIT), 15(8): 860 - 862, 31 July-5 Aug. 2011.

[4] Ido Tal and Alexander Vardy, How to construct polar codes. IEEE Transactions on Information Theory, 59(10): 6562 - 6582, 2013.

[5] Sarkis and Gross, Polar codes for data storage applications. Proc. IEEE Int. Conf. Comput., Netw. Commun. (ICNC) , 840 - 844, Jan.2013.

[6] Sarkis, Giard, Vardy, Thibeault, and Gross, Fast polar decoders: Algorithm and implementation. IEEE Journal on Selected Areas in Communications, 32(5): 946 - 957, May 2014.

[7] Alexios Balatsoukas - Stimming, Mani Bastani Parizi, and Andreas Burg, LLR-Based Successive Cancellation List Decoding of Polar Codes. IEEE Transactions on Signal Processing, 63(19): 3903 - 3907, Oct.1, 2015.

[8] Zhang, Liu, Pan, Gong, Yang. A Practical Construction Method for Polar Codes. IEEE Communications Letters, 18(11): 1871 - 1874, Nov. 2014.

[9] Vangala, Viterbo, Hong. A comparative study of polar code constructions for the AWGN channel, arXiv:1501.02473, 2015.

[10] Mori and Tanaka, Performance of polar codes with the construction using density evolution. IEEE Comm. Letters, 13(7): 519 - 521, July 2009.