Master Project

Polar Codes: Construction and Performance Analysis

Ramtin Pedarsani

ramtin.pedarsani@epfl.ch

Supervisor: Prof. Emre Telatar
Assistant: Hamed Hassani

Information Theory Laboratory (LTHI)
School of Computer and Communication Sciences (IC)
Swiss Federal Institute of Technology (EPFL)

June 2011

Abstract

Polar coding, recently invented by Erdal Arıkan, is an encoding/decoding scheme that provably achieves the capacity of the class of symmetric binary memoryless channels. In this thesis, we address two important problems regarding polar codes: the construction of polar codes and performance of polar codes. First, we consider the problem of efficiently constructing polar codes over symmetric binary discrete memoryless channels and provide some algorithms for channel quantization that can be analyzed for complexity and accuracy. In particular, we show that the algorithm can find almost all the "good" channels with computing complexity which is essentially linear in block-length. Next, we introduce different methods and algorithms to enhance the performance of the successive cancellation polar decoder. We provide numerical evidence as well as some mathematical analysis to show that the performance of polar codes are improved by using these algorithms.

Acknowledgment

First and foremost I offer my sincerest gratitude to my supervisor, Prof. Emre Telatar, who has supported me throughout my thesis with his patience, vast knowledge, and amazing brilliance. I attribute the level of my Masters degree to his wisdom and effort, and without his guidance and persistent help this thesis would not have been possible. I greatly appreciated the kindness and honesty that were part of every interaction we had. One simply could not wish for a better or friendlier supervisor.

I am indebted to Dr. Olivier Lévêque who was my first adviser at EPFL. I began doing research in Information Theory under his supervision in a semester project which led to two publications. I learned many things from Olivier, both on a technical and on a personal level.

My special thanks go to Prof. Erdal Arıkan, from whom I learned many things during his visit to EPFL. I would also like to thank Prof. Alexander Vardy for accepting to be the expert of my defense. I am grateful to Dr. Sheng Yang and Dr. Ido Tal for their help and contributions in our joint papers.

I would like to express my deepest appreciation to Hamed Hassani, who his continuous help during this period played a major role on the fulfillment of this project. Besides being a great assistant to my project, Hamed was one of my closest friends during my studies at EPFL.

Finally, I devote my special thanks to my parents, my brother Pedram, and his wife Noushin. I would like to thank Pedram and Noushin for their constant support and encouragement during my stay in Lausanne. There are no words that can fully express my gratitude to my parents. This thesis is dedicated with love to them.

Contents

List of Figures

List of Tables

1 Introduction

The purpose of communication systems is to transmit data reliably over a noisy channel. The general block diagram of a communication system is shown in Figure 1.1. The source encoder removes the redundant information from the source's data. The channel encoder adds redundancy to the data such that reliable communication can be achieved over a noisy channel. The task of the channel decoder is to reproduce the data sent over the channel from the channel output. In the end, the source decoder reproduces the source's data from the output of channel decoder. Our main concern in this thesis will be the blocks channel encoder, channel, and channel decoder which we describe in more detail in the sequel.

Figure 1.1: Block diagram of communication systems

1.1 Channel Coding Preliminaries

1.1.1 Channel Models

A channel is defined mathematically as a set of possible inputs to the channel $\mathcal{X}$, a set of possible outputs to the channel $\mathcal{Y}$, and a conditional probability distribution on the set of the outputs conditioned on the set of the inputs $W(y|x)$. The simplest class of channels are discrete memoryless channels (DMC).

Definition 1. *A symmetric binary discrete memoryless channel (B-DMC) is a B-DMC $W : \{0,1\} \to \mathcal{Y}$ with the additional property that there exists a permutation over the outputs of the channel $\pi : \mathcal{Y} \to \mathcal{Y}$ such that $\pi = \pi^{-1}$ and $W(y|0) = W(\pi(y)|1)$.*

Symmetric B-DMCs are an important class of channels studied in information theory. Two important examples of B-DMCs are binary symmetric channels (BSC) and binary erasure channels (BEC).

1.1.2 Representation of a Symmetric B-DMC as a Collection of BSC's

Any symmetric B-DMC can be represented as a collection of binary symmetric channels (BSC's). The binary input is given to one of these BSC's at random such that the i-th BSC is chosen with probability p_i. The output of this BSC together with its cross over probability x_i is the output of the channel. Therefore, a symmetric B-DMC W can be completely described by a random variable $\chi \in [0, 1/2]$. The *pdf* of χ will be of the form:

$$P_\chi(x) = \sum_{i=1}^{m} p_i \delta(x - x_i) \tag{1.1}$$

such that $\sum_{i=1}^{m} p_i = 1$ and $0 \leq x_i \leq 1/2$.

1.1.3 Two Important Parameters

We define two important parameters of symmetric B-DMC's: the mutual information and the Bhattacharyya parameter.

Definition 2. *The mutual information of a B-DMC with input alphabet $\mathcal{X} = \{0,1\}$ is defined*

$$I(W) \triangleq \frac{1}{2} \sum_{y \in \mathcal{Y}} \sum_{x \in \mathcal{X}} W(y|x) \log \frac{W(y|x)}{\frac{1}{2}W(y|0) + \frac{1}{2}W(y|1)} \tag{1.2}$$

Note that the capacity of a symmetric B-DMC equals the mutual information between the input and output of the channel with uniform distribution on the inputs. $I(W)$ is a measure of rate in a channel. It is well-known that reliable communication is possible over a symmetric B-DMC at any rates up to $I(W)$.

Definition 3. *The Bhattacharyya parameter of a channel is defined as*

$$Z(W) \triangleq \sum_{y \in \mathcal{Y}} \sqrt{W(y|0)W(y|1)}. \tag{1.3}$$

The Bhattacharyya parameter is a measure of the reliability of a channel since $Z(W)$ is an upper bound on the probability of maximum-likelihood (ML) decision error for uncoded transmission over W.

Furthermore, note that $Z(W)$ and $1 - I(W)$ are expectations of the functions $f(x) = 2\sqrt{x(1-x)}$ and $g(x) = -x\log(x) - (1-x)\log(1-x)$ over the distribution P_χ, respectively.

1.1.4 Channel Degradation

We define the notion of channel degradation which will be used in Chapter 3.

Definition 4. *Suppose that P_1 and P_2 are two channels with probability transitions matrices $p_1(y|x)$ and $p_2(z|x)$ respectively. Channel P_2 is said to be a stochastically degraded form of P_1 if there exists a probability transition matrix $p_3(z|y)$ such that*

$$p_2(z|x) = \sum_{y \in \mathcal{Y}} p_1(y|x) p_3(z|y)$$

Note that a degraded channel has larger Bhattacharrya parameter and smaller mutual information than the original channel.

1.2 Polar Codes

Polar codes, introduced by Arıkan in [1], are linear codes which provably achieve the capacity of symmetric B-DMC's. The idea of polar codes is to create from N independent copies of a B-DMC W, N different channels $W_N^{(i)}$, $1 \leq i \leq N$ through a linear transformation, such that as N grows large these synthesized channels are polarized. I.e., their mutual information are close to either 0 or 1. It is shown that the fraction of indices i for which $I(W_N^{(i)})$ is close to 1 is $I(W)$, and the fraction of indices i for which $I(W_N^{(i)})$ is close to 0 is $1 - I(W)$, asymptotically. The encoding/decoding complexity of the codes is $\mathcal{O}(N \log N)$. We will discuss polar codes in more detail in Chapter 2.

1.3 Thesis Main Contribution and Organization

The two important concerns about using polar codes in practice, as we will see later, are

1. The construction of polar codes
2. The performance of polar codes in short block-length

This thesis will address these two important issues and provide some practical methods and algorithms to address them.

The rest of the thesis is organized as follows:

Chapter 2 goes briefly through the method of channel polarization and introduces polar codes as a linear capacity-achieving code. We describe the low-complexity successive cancellation (SC) polar decoder which is used to achieve the capacity of symmetric channels.

Chapter 3 considers the problem of efficiently constructing polar codes over binary discrete memoryless symmetric channels. Following the approach of Tal and Vardy [3], we present a framework where the algorithms of [3] and new related algorithms can be analyzed for complexity and accuracy.

Chapter 4 introduces different algorithms to improve the performance of polar codes over different types of channels. The results are mainly supported by numerical analysis and partly by analytical evidence.

1.4 Notations

Throughout the thesis, we use the following notations. We use upper case letters X to denote the random variables and lower case letters x for their realizations. The boldface lower case letters $\mathbf{x}$ represent a vector. $\mathbf{X}^i$ represents a vector of random variables $[X_1, X_2, \ldots, X_i]$. Matrices are shown by boldface upper case letters $\mathbf{G}$. $\mathbb{E}_X(\cdot)$ stands for the expectation operator over the random variable X. $[\cdot]^T$ denotes the matrix transposition. $\log(\cdot)$ and $\ln(\cdot)$ stand for the base-2 and natural logarithms, respectively.

Channel Polarization

2

In this chapter, we discuss polar codes, introduced by Arıkan in [1] for channel coding. The chapter is a short overview of Arıkan's paper [1] without too much concern about the proofs and mathematical details.

2.1 Polar Encoder

Polar codes are linear codes, i.e., any linear combination of codewords is another codeword of the code. The polar transform is to apply the transform $\mathbf{G_2^{\otimes n}}$, the n^{th} Kronecker power of $\mathbf{G_2} = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}$ to the block of $N = 2^n$ bits $\mathbf{U}$.

The polar encoder chooses a set of NR rows of the matrix $\mathbf{G_n}$ to form a $NR \times N$ matrix which is used as the generator matrix in the encoding procedure.The way this set is chosen is dependent on the channel W and uses a phenomenon called channel polarization which is described later.

Using the fast transform methods in signal processing, it is easy to show that the complexity of the polar encoder is $\mathcal{O}(N \log N)$. In fact, due to the recursive channel combining the encoding complexity of blocklength N, $\chi_E(N)$ is

$$\chi_E(N) = \frac{N}{2} + 2\chi_E(\frac{N}{2}), \tag{2.1}$$

since we require $\frac{N}{2}$ XOR operations plus encoding two blocks of length $\frac{N}{2}$. For blocklength 2, $\chi_E(2) = 1$. Therefore, one can conclude that $\chi_E(N) = \mathcal{O}(N \log N)$.

2.2 Channel Polarization

Channel polarization is an operation which produces N channels $\{W_N^{(i)} : 1 \leq i \leq N\}$ from N independent copies of a B-DMC W such that the new parallel channels are polarized in the sense that their mutual information is either close to 0 (completely noisy channels) or close to 1 (perfectly noiseless channels). Channel polarization consists of two phases:

1. Channel Combining: In this phase, copies of a B-DMC are combined in a recursive manner in n steps to form a vector channel W_N, where $N = 2^n$. The basic transformation used in channel combining is the following.

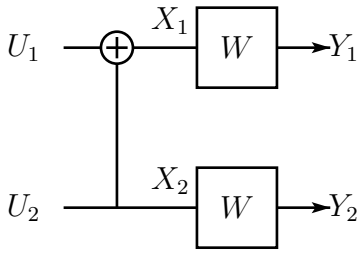

Figure 2.1: Combining two channels

$$W_2(y_1, y_2|u_1, u_2) = W(y_1|u_1 \oplus u_2)W(y_2|u_2) \tag{2.2}$$

As one can see, two seperate channels are combined to create a new vector channel of size 2, which is $W_2 : \{0,1\}^2 \to \mathcal{Y}^2$. Since the linear transform between (U_1, U_2) and (X_1, X_2) is a one-to-one mapping the following equality holds:

$$I(U_1, U_2; Y_1, Y_2) = I(X_1, X_2; Y_1, Y_2) = 2I(W) \tag{2.3}$$

The channel combining for general $N = 2^n$ is done recursively in the following way: The channel $W_N : \mathcal{X}^N \to \mathcal{Y}^N$ is defined as

$$W_N(\mathbf{Y}^N|\mathbf{U}^N) = W_{N/2}(\mathbf{Y}^{N/2}|\mathbf{U}_o^N \oplus \mathbf{U}_e^N)W_{N/2}(\mathbf{Y}_{N/2+1}^N|\mathbf{U}_e^N),$$

where $\mathbf{U}_o^{N-1} = (u_1, u_3, \ldots, u_{N-1})$ and $\mathbf{U}_e^{N-1} = (u_2, u_4, \ldots, u_N)$. The channel combining procedure for $N = 8$ is shown in Figure 2.2. For more details please refer to [1] and [2].

2. Channel Splitting: In the second phase, the vector channel W_N is split back into N channels $W_N^{(i)} : \{0,1\} \to \mathcal{Y}^N \times \{0,1\}^{i-1},\ 1 \leq i \leq N$.

   In the basic case, $N = 2$, using chain rule of mutual information, the left hand side of equation (2.3) can be written as

$$I(U_1, U_2; Y_1, Y_2) = I(U_1; Y_1, Y_2) + I(U_1; Y_1, Y_2, U_1) \tag{2.4}$$

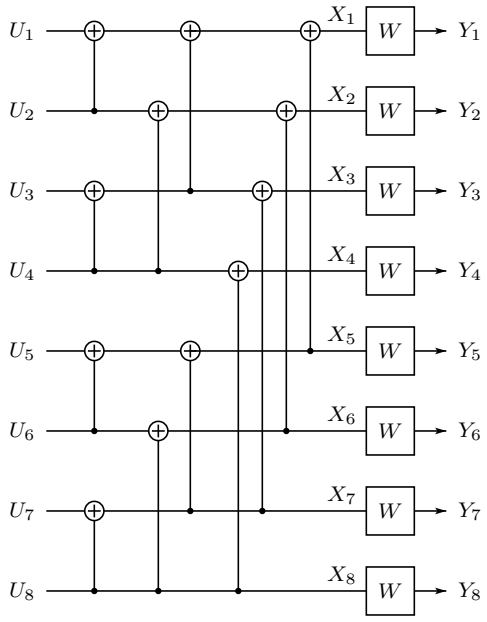

Figure 2.2: Channel combining

We can see that $I(U_1; Y_1, Y_2)$ is the mutual information of the channel between U_1 and Y_1, Y_2. Let this channel be $W^- : \{0,1\} \to \mathcal{Y}^2$. Furthermore, $I(U_2; Y_1, Y_2, U_1)$ is the mutual information of the channel between U_2 and the output given that U_1 is known. Let this channel be W^+. The transition probabilities of these two channels can be written as

$$W^-(y_1, y_2|u_1) = \frac{1}{2} \sum_{u_2 \in \{0,1\}} W(y_1|u_1 \oplus u_2) W(y_2|u_2) \tag{2.5}$$

$$W^+(y_1, y_2, u_1|u_2) = \frac{1}{2} W(y_1|u_1 \oplus u_2) W(y_2|u_2) \tag{2.6}$$

In other words, the combined channel W_2 is split into two channels W^+ and W^-.

The following two properties hold for channels created by operations (2.5) and (2.6). [1]

(i)

$$I(W^+) + I(W^-) = 2I(W) \tag{2.7}$$

(ii)

$$Z(W^-) \leq 2Z(W) - Z(W)^2 \tag{2.8}$$

$$Z(W^+) = Z(W)^2 \tag{2.9}$$

Note that $I(W^+) \geq I(W) \geq I(W^-)$. As one can see, $I(W^+)$ and $I(W^-)$ are pushed to the extremes 0 and 1 with respect to $I(W)$ while their sum is still $2I(W)$.

Similarly, we can split channel vector W_4 into 4 channels W^{--}, W^{-+}, W^{+-}, and W^{++}. In general, from $N = 2^n$ copies of channel W, we can create N channels $W^{s_1,s_2,\ldots,s_n}$, $s_i \in \{-,+\}$, $1 \le i \le n$. Alternatively, we represent these channels by $W_N^{(i)}$, $1 \le i \le N$. It is easy to check that the channels created following the operations (2.5) and (2.6), are those channels that appear in the chain rule expansion of the mutual information.

$$\begin{aligned} NI(W) &= I(\mathbf{X}^N; \mathbf{Y}^N) && (2.10)\\ &= I(\mathbf{U}^N; \mathbf{Y}^N) && (2.11)\\ &= \sum_{i=1}^{N} I(U_i; \mathbf{Y}^N, \mathbf{U}^{i-1}) && (2.12) \end{aligned}$$

To analyze the behavior of these channels, we define a random process. Let $\{B_n : n \ge 1\}$ be a sequence of i.i.d. Bernoulli random variables with parameter $\frac{1}{2}$, and $\{\mathcal{F}_n, n \ge 1\}$ be the σ-field generated by $\mathbf{B}^n$. Define a tree process as the following

$$W_{n+1} = \begin{cases} {W_n}^- & \text{if } B_n = 0, \\ {W_n}^+ & \text{if } B_n = 1, \end{cases}$$

and $W_0 = W$. We are interested to see the behavior of the random processes $I(W_n)$ and $Z(W_n)$. In [1, 2], it is shown that

(i) The sequence $\{I_n, \mathcal{F}_n, n \ge 0\}$ is a bounded martingale.

(ii) The sequence $\{Z_n, \mathcal{F}_n, n \ge 0\}$ is a bounded super-martingale.

Therefore, using the martingale properties it is readily shown that

(i) The sequence $\{I_n\}$ converges almost surely to a random variable I_∞ and

$$I_\infty = \begin{cases} 1 & \text{w.p. } I(W), \\ 0 & \text{w.p. } 1 - I(W). \end{cases}$$

(ii) The sequence $\{Z_n\}$ converges almost surely to a random variable Z_∞ and

$$Z_\infty = \begin{cases} 1 & \text{w.p. } 1 - I(W), \\ 0 & \text{w.p. } I(W). \end{cases}$$

In other words, the N created channels from recursively applying operations (2.5) and (2.6), are polarized to the extremes: they are either close to completely noisy channels (mutual information close to 0), or perfectly clean channels (mutual information close to 1).

2.3 Polar Codes Achieve Symmetric Capacity of the Channel

In section 2.2, we showed how to create polarized channels which are either noiseless (good channels) or completely noisy (bad channels). Therefore, a natural coding scheme is to send information bits on those good channels, i.e., U_i is an information bit if $I(U_i; \mathbf{Y}^N, \mathbf{U}^{i-1})$ is close to 1, and freeze the bit U_i otherwise, and reveal this value to the decoder. Note that $I(U_i; \mathbf{Y}^N, \mathbf{U}^{i-1})$ corresponds to decoding U_i with the knowledge of the output $\mathbf{Y}$ and the previously decoded bits $\mathbf{U}^{i-1}$. This naturally leads to a successive cancellation (SC) decoder which decodes the bits $U_1, U_2, \ldots, U_N$ in order. In fact, the decoder has only an estimate of the the bits $\hat{U}_j,\ 1 \leq j \leq i-1$. In [1, 2], it is shown that the block error probability of the SC decoder decays to zero for rates below $I(W)$; consequently, polar codes achieve the capacity of symmetric B-DMC's using the SC decoder.

It is worth mentioning that the following bounds for the block error probability of polar codes hold:

$$\max_{i \in F^c} \frac{1}{2}\left(1 - \sqrt{1 - Z(W_N^{(i)})^2}\right) \leq P_{SC} \leq \sum_{i \in F^c} Z(W_N^{(i)}), \tag{2.13}$$

where F is the set of indices of frozen bits and P_{SC} is the average block error probability of the code under SC decoder.

Furthermore, it is shown in [7] that the block error probability decays to zero like $\mathcal{O}(2^{-\sqrt{N}})$ asymptotically.

2.4 Polar Decoder

The SC decoder generates an estimate $\hat{\mathbf{u}}^N$ of $\mathbf{u}^N$ by observing the channel output $\mathbf{y}^N$. The decoder takes N decisions for each u_i. If u_i is a frozen bit, the decoder will fix $\hat{u}_i$ to its known value. If u_i is an information bit, the decoder waits to estimate all the previous bits, and then computes the following likelihood ratio:

$$L_N^{(i)}(\mathbf{y}^N, \hat{\mathbf{u}}^{i-1}) = \frac{W_N^{(i)}(\mathbf{y}^N, \hat{\mathbf{u}}^{i-1}|0)}{W_N^{(i)}(\mathbf{y}^N, \hat{\mathbf{u}}^{i-1}|1)} \tag{2.14}$$

The decoder sets $\hat{u}_i = 0$, if $L_N^{(i)} \geq 1$, and $\hat{u}_i = 1$ otherwise. Therefore, the complexity of the decoding algorithm is determined by the complexity of computing the LR's. The recursive relations for computing the LR values are

$$L_N^{(2i-1)}(\mathbf{y}^N, \hat{\mathbf{u}}^{2i-2}) = \frac{1 + L_{N/2}^{(i)}(\mathbf{y}^{N/2}, \hat{\mathbf{u}}_o^{2i-2} \oplus \hat{\mathbf{u}}_e^{2i-2}) L_{N/2}^{(i)}(\mathbf{y}_{N/2+1}^N, \hat{\mathbf{u}}_e^{2i-2})}{L_{N/2}^{(i)}(\mathbf{y}^{N/2}, \hat{\mathbf{u}}_o^{2i-2} \oplus \hat{\mathbf{u}}_e^{2i-2}) + L_{N/2}^{(i)}(\mathbf{y}_{N/2+1}^N, \hat{\mathbf{u}}_e^{2i-2})}, \tag{2.15}$$

$$L_N^{(2i)}(\mathbf{y}^N, \hat{\mathbf{u}}^{2i}) = L_{N/2}^{(i)}(\mathbf{y}^{N/2}, \hat{\mathbf{u}}_o^{2i-2} \oplus \hat{\mathbf{u}}_e^{2i-2})^{1-2\hat{u}_{2i-1}} L_{N/2}^{(i)}(\mathbf{y}_{N/2+1}^N, \hat{\mathbf{u}}_e^{2i-2}). \tag{2.16}$$

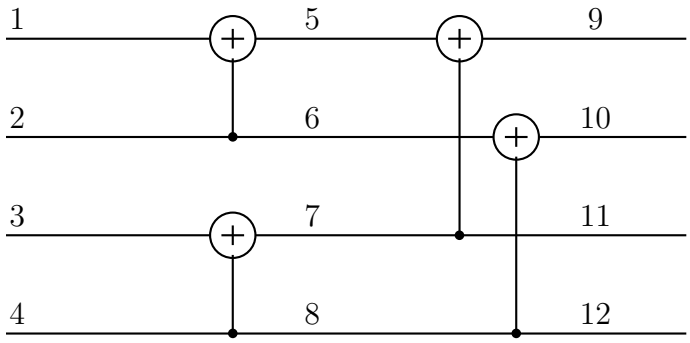

Figure 2.3: Polar decoder for $N = 4$

Therefore, the LR's can be computed with $\mathcal{O}(N)$ computations from the likelihood ratios of the previous level. Let $\chi_D(N)$ be the complexity of the decoder with blocklength N. Thus,

$$\chi_D(N) = \mathcal{O}(N) + 2\chi_D(N/2) \tag{2.17}$$

This implies the $\mathcal{O}(N \log N)$ complexity of the SC polar decoder.

To clarify, we describe the decoding procedure in more details for $N = 4$. As one can see in Figure 2.3, the decoder should compute $N(\log N + 1) = 12$ LR's. At first, the decoder activates node 1 to decide the value of $\hat{u}_1$. Node 1 needs the LR's of nodes 5 and 6 so it activates them. Nodes 5 and 6 themselves activate nodes 9,11,10, and 12 respectively. The LR values at these nodes are known since they are at the channel level. Therefore, LR's at nodes 5 and 6 are computed using formula 2.15 and from them, LR at node 1 is computed. Now the decoder decides on $\hat{u}_1$ on the basis of the LR if it is an information bit. For computing LR at node 2, this node activates nodes 5 and 6 whose LR's are already computed. If $\hat{u}_1 = 0$, the LR's at nodes 5 and 6 are combined using formula 2.16, and if $\hat{u}_1 = 1$, the LR at node 6 will be combined with the inverse of LR at node 5 using formula 2.16. Estimating $\hat{u}_1$ and $\hat{u}_2$ update the estimated bits at nodes 5 and 6. To estimate u_3, node 3 activates nodes 7 and 8. The LR's at these nodes are computed using LR values of nodes 9 to 12 alongside the status of nodes 5 and 6, as previously described. In fact, node 5 determines whether the LR at node 11 should be combined with the LR at node 9 or the inverse of the LR at node 9. Finally, from nodes 7 and 8 the LR's at node 3 and then 4 (on the basis of $\hat{u}_3$) are computed and the decoding procedure is completed.

2.5 Simulation Results

In this part, we see the performance of a polar code using SC decoder over a BSC channel with cross-over probability 0.11 (capacity 0.5). For determining the frozen indices we have used a Monte-Carlo method [1]. As one can see, the performance of polar codes is not very impressive in short block-lengths [2, 10, 11]. In chapter 4, we try to improve the performance of polar codes under SC decoder by various methods.

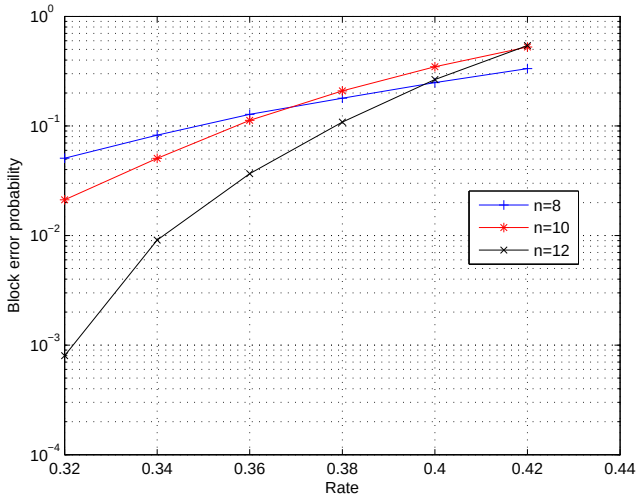

Figure 2.4: Performance of SC decoder in terms of block error probability, when transmission takes place over the BSC with capacity 0.5.

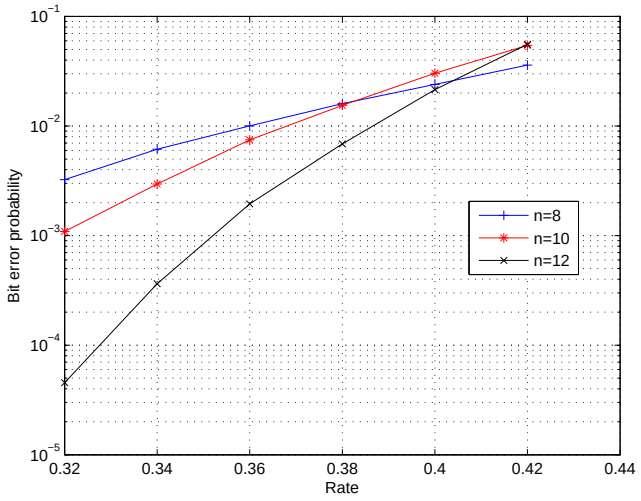

Figure 2.5: Performance of SC decoder in terms of bit error probability, when transmission takes place over the BSC with capacity 0.5.

3 Polar Codes Construction

In this chapter, we consider the problem of efficiently constructing polar codes over symmetric binary discrete memory-less channels. As mentioned in Section 1.3, the construction of polar codes is one of the important concerns in using polar codes in practice.

3.1 Problem Formulation

Designing a polar code is equivalent to finding the set of good indices among the N polarized channels described in Section 2.2. The output alphabet of the channel $W_N^{(i)}$ is $\mathcal{Y}^N \times \{0,1\}^i$, and consequently, the cardinality of the output alphabet of the channels after n levels of polarization grows exponentially in block-length. So computing the exact transition probabilities of these channels seems to be intractable and hence we need some efficient methods to "approximate" these channels.

In [1], it is suggested to use a Monte-Carlo method for estimating the Bhattacharyya parameters. In this thesis, we follow the approach of Tal and Vardy [3] to use *quantization* methods for the purpose. Quantization is to approximate a given channel with another channel that has less fewer output symbols [3, 4, 5], [6, Appendix B]. More precisely, given a number k, the task is to come up with efficient methods to replace channels that have more that k outputs with "close" channels that have at most k outputs. In the following, we state some comments to clarify the problem. The term "close" depends on the definition of the quantization error which can be different depending on the context. In our problem, we define the quantization error as the difference between the true set of good indices and the approximate set of good indices. Unfortunately, this type of error seems to be analytically intractable. For the theoretical analysis of the problem, we consider other types of error to analyze.

The quantization problem is to find procedures to replace each channel P at each level

of the binary tree with another symmetric channel $\tilde{P}$ with the number of outputs limited to k such that firstly, the set of good indices obtained with this procedure is a subset of the true good indices obtained from the channel polarization, i.e., channel $\tilde{P}$ is *polar degraded* with respect to P, and secondly the ratio of these good indices is maximized. As a result, the indices identified as good by the quantization procedure is never bad, i.e., misidentification is always conservative.

To clarify the procedure, we start from channel W, and quantize it to $\tilde{W}$. Then performing the channel splitting operations (2.5) and (2.6), we create channels $\tilde{W}^+$ and $\tilde{W}^-$. Then, we quantize these two channels and proceed to create the N quantized channels in the end. A formal definition of a quantization procedure would be the following. Let Q_k be a quantization procedure that assigns to each channel P a binary symmetric channel $\tilde{P}$ such that the output alphabet of $\tilde{P}$ is limited to a constant k. We call Q_k admissible if for any i and n

$$I(\tilde{W}_N^{(i)}) \leq I(W_N^{(i)}). \tag{3.1}$$

Alternatively, we call Q_k admissible if for any i and n

$$Z(\tilde{W}_N^{(i)}) \geq Z(W_N^{(i)}). \tag{3.2}$$

Note that (3.1) and (3.2) are essentially equivalent as N grows large. Given an admissible procedure Q_k and a symmetric B-DMC W, let $\rho(Q_k, W)$ be[1]

$$\rho(Q_k, W) = \lim_{n\to\infty} \frac{|\{i : I(\tilde{W}_N^{(i)}) > \frac{1}{2}\}|}{N} \tag{3.3}$$

So the quantization problem is that given a number $k \in \mathbb{N}$ and a channel W, how can we find admissible procedures Q_k such that $\rho(Q_k, W)$ is maximized and is close to the capacity of W. An important question which arises here is that whether we can achieve capacity of the channel as k tends to infinity. Obviously, if we first let k tend to infinity and then N, the limit is indeed capacity, but we are stating another problem here which is what will happen if we first let N go to infinity and then k. We answer this question in Section 3.4.

3.2 Algorithms for Quantization

In this section, we propose different algorithms for the channel quantization.

Recall the representation of symmetric B-DMC with a collection of BSC's in Section 1.1, Equation (1.1). In the quantization problem we want to replace the mass distribution P_χ with another mass distribution $P_{\tilde{\chi}}$ such that the number of output symbols of $\tilde{\chi}$ is at most k, and the channel $\tilde{W}$ is polar degraded with respect to W.

There are two known operations which imply polar degradation:

(i) Stochastically degrading the channel.

[1]Instead of $\frac{1}{2}$ in (3.3) we can use any number in $(0, 1)$.

(ii) Replacing the channel with a BEC channel with the same Bhattacharyya parameter.

Furthermore, note that the *stochastic dominance* of random variable $\tilde{\chi}$ with respect to χ implies $\tilde{W}$ is stochastically degraded with respect to W. (But the reverse is not true.)

In the following, we state two different algorithms based on different methods of polar degradation of the channel. The first algorithm is a naive algorithm called the mass transportation algorithm based on the stochastic dominance of the random variable $\tilde{\chi}$, and the second one which outperforms the first is called greedy mass merging algorithm. For both of the algorithms the quantized channel is stochastically degraded with respect to the original one.

3.2.1 Greedy Mass Transportation Algorithm

In the most general form of this algorithm we basically look at the problem as a *mass transport* problem. In fact, we have non-negative masses p_i at locations $x_i, i = 1, \cdots, m, x_1 < \cdots < x_m$. What is required is to move the masses, by only moves to the right, to concentrate them on $k < m$ locations, and try to minimize $\sum_i p_i d_i$ where $d_i = x_{i+1} - x_i$ is the amount i^{th} mass has moved. Later, we will show that this method is not optimal but useful in theoretical analysis of algorithms that follow.

Algorithm 1 Mass Transportation Algorithm

1: Start from the list $(p_1, x_1), \cdots, (p_m, x_m)$.
2: Repeat $m - k$ times
3: Find $j = \text{argmin}\{p_i d_i : i \neq m\}$
4: Add p_j to p_{j+1} (i.e. move p_j to x_{j+1})
5: Delete (p_j, x_j) from the list.

Note that Algorithm 1 is based on the stochastic dominance of random variable $\tilde{\chi}$ with respect to χ. Furthermore, in general, we can let $d_i = f(x_{i+1}) - f(x_i)$, for an arbitrary bounded increasing function f.

3.2.2 Mass Merging Algorithm

The second algorithm merges the masses. Two masses p_1 and p_2 at positions x_1 and x_2 would be merged into one mass $p_1 + p_2$ at position $\bar{x}_1 = \frac{p_1}{p_1+p_2} x_1 + \frac{p_2}{p_1+p_2} x_2$. This algorithm is based on the stochastic degradation of the channel, but the random variable χ is not stochastically dominated by $\tilde{\chi}$. The greedy algorithm for the merging of the masses is shown in Algorithm 2.

Note that in practice, the function f can be any increasing concave function, for example, the entropy function or the Bhattacharyya function. In fact, since the algorithm is greedy and suboptimal, it is hard to investigate explicitly how changing the function f will affect the total error of the algorithm in the end (i.e., how far $\tilde{W}$ is from W). In Section 3.5, we will see the results of applying Algorithm 2 for 3 different functions: the Bhattacharrya function, the entropy function, and the function $f(x) = x(1 - x)$.

Algorithm 2 Merging Masses Algorithm

1: Start from the list $(p_1, x_1), \cdots, (p_m, x_m)$.
2: Repeat $m - k$ times
3: Find $j = \text{argmin}\{p_i(f(\bar{x}_i) - f(x_i)) - p_{i+1}(f(x_{i+1}) - f(\bar{x}_i)) : \ i \neq m\}$ $\quad \bar{x}_i = \frac{p_i}{p_i+p_{i+1}} x_i + \frac{p_{i+1}}{p_i+p_{i+1}} x_{i+1}$
4: Replace the two masses (p_j, x_j) and (p_{j+1}, x_{j+1}) with a single mass $(p_j + p_{j+1}, \bar{x}_j)$.

3.3 Bounds on the Approximation Loss

In this section, we provide some bounds on the maximum approximation loss we have in the algorithms. We define the "approximation loss" to be the difference between the expectation of the function f under the true distribution P_χ and the approximated distribution $P_{\tilde{\chi}}$. Note that the kind of error that is analyzed in this section is different from what was defined in Section 3.1. The connection of the approximation loss with the quantization error is made clear in Theorem 1. For convenience, we will simply stick to the word "error" instead of "approximation loss" from now on.

Lemma 1. *The maximum error made by Algorithms 1 and 2 is upper bounded by* $\mathcal{O}(\frac{1}{k})$.

Proof. First, we derive an upper bound on the error of Algorithms 1 and 2 in each iteration, and therefore a bound on the error of the whole process. Let us consider Algorithm 1. The problem can be reduced to the following optimization problem:

$$e = \max_{p_i, x_i} \min_i (p_i d_i) \tag{3.4}$$

such that

$$\sum_i p_i = 1, \ \sum_i d_i \leq 1, \tag{3.5}$$

where $d_i = f(x_{i+1}) - f(x_i)$. We prove the lemma by Cauchy-Schwarz inequality.

$$\min_i p_i d_i = \left(\sqrt{\min_i p_i d_i}\right)^2 = \left(\min_i \sqrt{p_i d_i}\right)^2 \tag{3.6}$$

Now by applying Cauchy-Schwarz we have

$$\sum_{i=1}^{m} \sqrt{p_i d_i} \leq \left(\sum_{i=1}^{m} p_i\right)^{1/2} \left(\sum_{i=1}^{m} d_i\right)^{1/2} \leq 1 \tag{3.7}$$

Since the sum of m terms $\sqrt{p_i d_i}$ is less than 1, the minimum of the terms will be certainly less than $\frac{1}{m}$. Therefore,

$$e = \left(\min \sqrt{p_i d_i}\right)^2 \leq \frac{1}{m^2}. \tag{3.8}$$

For Algorithm 2, achieving the same bound as Algorithm 1 is trivial. Denote $e^{(1)}$ the error made in Algorithm 1 and $e^{(2)}$ the error made in Algorithm 2. Then,

$$\begin{aligned} e_i^{(2)} &= p_i \left(f(\bar{x}_i) - f(x_i)\right) - p_{i+1} \left(f(x_{i+1}) - f(\bar{x}_i)\right) && (3.9)\\ &\leq p_i \left(f(\bar{x}_i) - f(x_i)\right) && (3.10)\\ &\leq p_i \left(f(x_{i+1}) - f(x_i)\right) = e_i^{(1)}. && (3.11) \end{aligned}$$

Consequently, the error generated by running the whole algorithm can be upper bounded by $\sum_{i=k+1}^{n} \frac{1}{i^2}$ which is $\mathcal{O}(\frac{1}{k})$.

□

What is stated in Lemma 1 is a loose upper bound for the error of Algorithm 2. To achieve better bounds, we upper bound the error made in each iteration of the Algorithm 2 as the following:

$$\begin{aligned} e_i &= p_i \left(f(\bar{x}_i) - f(x_i)\right) - p_{i+1} \left(f(x_{i+1}) - f(\bar{x}_i)\right) && (3.12)\\ &\leq p_i \frac{p_{i+1}}{p_i + p_{i+1}} \Delta x_i f'(x_i) - p_{i+1} \frac{p_i}{p_i + p_{i+1}} \Delta x_i f'(x_{i+1}) && (3.13)\\ &= \frac{p_i p_{i+1}}{p_i + p_{i+1}} \Delta x_i \left(f'(x_i) - f'(x_{i+1})\right) && (3.14)\\ &\leq \frac{p_i + p_{i+1}}{4} \Delta x_i^2 |f''(c_i)|, && (3.15) \end{aligned}$$

where $\Delta x_i = x_{i+1} - x_i$ and (3.13) is due to concavity of function f. Furthermore, (3.15) is by mean value theorem, where $x_i \leq c_i \leq x_{i+1}$.

If $|f''(x)|$ is bounded for $x \in (0, 1)$, for example for $f(x) = x(1-x)$, we can prove that $\min_i e_i \sim \mathcal{O}(\frac{1}{m^3})$ by methods similar to those used in Lemma 1. Therefore the error of the whole algorithm would be $\mathcal{O}(\frac{1}{k^2})$. Unfortunately, this is not the case for either of entropy function or Bhattacharyya function. However, we can still achieve a better upper bound for the error of Algorithm 2.

Lemma 2. *The maximum error made by Algorithm 2 for the entropy function $h(x)$ can be upper bounded by $\mathcal{O}(\frac{\log(k)}{k^{1.5}})$.*

Proof. Let us first find an upper bound for the second derivative of the entropy function $h(x) = -x\log(x) - (1-x)\log(1-x)$.

$$|h''(x)| = \frac{1}{x(1-x)\ln(2)}. \tag{3.16}$$

For $0 \leq x \leq \frac{1}{2}$ we have

$$|h''(x)| \leq \frac{2}{x\ln(2)}. \tag{3.17}$$

Now we are ready to prove the lemma. Using (3.17) the minimum error can further be upper bounded by

$$\min_i e_i \leq \min_i (p_i + p_{i+1}) \Delta x_i^2 \frac{1}{x_i \ln(4)}. \tag{3.18}$$

Now suppose that we have l mass points with $x_i \leq \frac{1}{\sqrt{m}}$ and $m-l$ mass points with $x_i \geq \frac{1}{\sqrt{m}}$. For the l first mass points we use the upper bound obtained in Algorithm 1. Hence, for $1 \leq i \leq l$ we have

$$\min_i e_i \leq \min_i p_i \Delta h(x_i) \tag{3.19}$$

$$\sim \mathcal{O}\left(\frac{\log(m)}{l^2\sqrt{m}}\right), \tag{3.20}$$

where (3.19) is due to (3.11) and (3.20) can be derived again by applying Cauchy-Schwarz inequality. Note that this time

$$\sum_{i=1}^{l} \Delta h(x_i) \leq h(\frac{1}{\sqrt{m}}) \sim \mathcal{O}\left(\frac{\log(m)}{\sqrt{m}}\right). \tag{3.21}$$

For the $m-l$ mass points one can write

$$\min_i e_i \leq \min_i (p_i + p_{i+1}) \Delta x_i^2 \frac{1}{x_i \ln(4)} \tag{3.22}$$

$$\leq \min_i (p_i + p_{i+1}) \Delta x_i^2 \frac{\sqrt{m}}{\ln(4)} \tag{3.23}$$

$$\sim \mathcal{O}\left(\frac{\sqrt{m}}{(m-l)^3}\right), \tag{3.24}$$

where (3.24) is due to Holder inequality as follows:

Let $q_i = p_i + p_{i+1}$. Therefore, $\sum_i (p_i + p_{i+1}) \leq 2$ and $\sum_i \Delta x_i \leq 1/2$.

$$\min_i q_i \Delta x_i^2 = \left(\left(\min_i q_i \Delta x_i^2\right)^{1/3}\right)^3 = \left(\min_i \left(q_i \Delta x_i^2\right)^{1/3}\right)^3 \tag{3.25}$$

Now by applying Holder inequality we have

$$\sum_i \left(q_i \Delta x_i^2\right)^{1/3} \leq \left(\sum_i q_i\right)^{1/3} \left(\sum_i \Delta x_i\right)^{2/3} \leq 1 \tag{3.26}$$

Therefore,

$$\min_i e_i \leq \sqrt{m} \left(\min_i (q_i \Delta x_i^2)^{1/3}\right)^3 \sim \mathcal{O}\left(\frac{\sqrt{m}}{(m-l)^3}\right). \tag{3.27}$$

Overall, the error made in the first step of the algorithm would be

$$\min_i e_i \sim \min\left\{\mathcal{O}\left(\frac{\log(m)}{l^2\sqrt{m}}\right), \mathcal{O}\left(\frac{\sqrt{m}}{(m-l)^3}\right)\right\} \tag{3.28}$$

$$\sim \mathcal{O}\left(\frac{\log(m)}{m^{2.5}}\right). \tag{3.29}$$

Therefore, the error generated by running the whole algorithm can be upper bounded by $\sum_{i=k+1}^{m} \frac{\log(i)}{i^{2.5}} \sim \frac{\log(k)}{k^{1.5}}$. □

We can see that the error is improved by a factor of $\frac{\log k}{\sqrt{k}}$ in comparison with Algorithm 1.

Now we use the result of Lemma 1 to provide bounds on the total error made in estimating the mutual information of a channel after n levels of operations (2.5) and (2.6).

Theorem 1. *Assume W is a symmetric B-DMC and using Algorithm 1 or 2 we quantize the channel W to a channel $\tilde{W}$. Taking $k = n^2$ is sufficient to give an approximation error that decays to zero.*

Proof. First notice that for any two symmetric B-DMCs W and V, doing the polarization operations (2.5) and (2.6), the following is true:

$$(I(W^-) - I(V^-)) + (I(W^+) - I(V^+)) = 2(I(W) - I(V)) \tag{3.30}$$

Let S_n be the sum of the errors in approximating the mutual information of the N channels after n level of polarizations. Replacing V with $\tilde{W}$ in (3.30) and using the result of Lemma 1, we get the following recursive relation for the sequence s_n.

$$\begin{aligned} s_0 &\leq \frac{1}{k} \\ s_n &\leq 2s_{n-1} + \frac{2^n}{k} \end{aligned} \tag{3.31}$$

Solving (3.31) explicitly, we have $s_n \leq \frac{(n+1)2^n}{k}$. Therefore, we conclude that after n levels of polarization the sum of the errors in approximating the mutual information of the 2^n channels is upper-bounded by $\mathcal{O}(\frac{n2^n}{k})$. In particular, taking $k = n^2$, one can say that the "average" approximation error of the 2^n channels at level n is upper-bounded by $\mathcal{O}(\frac{1}{n})$. Therefore, at least a fraction $1 - \frac{1}{\sqrt{n}}$ of the channels are distorted by at most $\frac{1}{\sqrt{n}}$ i.e., except for a negligible fraction of the channels the error in approximating the mutual information decays to zero.

□

The theorem above shows that the algorithm finds all the "good" channels except a negligible fraction of $1 - \frac{1}{\sqrt{n}}$. The computational complexity of construction is $\mathcal{O}(k^2 N)$ which for $k = n^2$ yields to an almost linear complexity in blocklength except a polylogarithmic factor of $(\log N)^4$.

3.4 Exchange of Limits

In this section, we show that there are admissible schemes such that as $k \to \infty$, the limit in (3.3) approaches $I(W)$ for any BMS channel W. In contrast to the previous section, k is a constant that grows, unlike $k = n^2$. We use definition stated in equation (3.2) for the admissibility of the quantization procedure.

Theorem 2. *Given a BMS channel W and for large enough k, there exist admissible quantization schemes Q_k such that $\rho(Q_k, W)$ is arbitrarily close to $I(W)$.*

Proof. Consider the following algorithm: The algorithm starts with a quantized version of W and it does the normal channel splitting transformation followed by quantization according to Algorithms 1 or 2, but once a sub-channel is sufficiently good, in the sense that its Bhattacharyya parameter is less than an appropriately chosen parameter δ, the algorithm replaces the sub-channel with a binary erasure channel which is degraded (polar degradation) with respect to it (As the operations (2.5) and (2.6) over an erasure channel also yields and erasure channel, no further quantization is need for the children of this sub-channel).

Since the ratio of the total good indices of $\mathrm{BEC}(Z(P))$ is $1 - Z(P)$, then the total error that we make by replacing P with $\mathrm{BEC}(Z(P))$ is at most $Z(P)$ which in the above algorithm is less that the parameter δ.

Now, for a fixed level n, according to Theorem 1 if we make k large enough, the ratio of the quantized sub-channels that their Bhattacharyya value is less that δ approaches to its original value (with no quantization), and for these sub-channels as explained above the total error made with the algorithm is δ. Now from the polarization theorem and by sending δ to zero we deduce that as $k \to \infty$ the number of good indices approaches the capacity of the original channel. □

3.5 Simulation Results

In order to evaluate the performance of our quantization algorithm, we compare the performance of the degraded quantized channel with the performance of an upgraded quantized channel. Similarly to Section 3.2 Algorithm 2, we introduce an algorithm which this time splits the masses between its two neighbors. To clarify, consider three neighbor masses in positions (x_{i-1}, x_i, x_{i+1}) with probabilities (p_{i-1}, p_i, p_{i+1}). Let $t = \frac{x_i - x_{i-1}}{x_{i+1} - x_{i-1}}$. Then, we split the middle mass at x_i to the other two masses such that the final probabilities will be $(p_{i-1} + (1-t)p_i, p_{i+1} + tp_i)$ at positions (x_{i-1}, x_{i+1}). The greedy algorithm is shown in Algorithm 3.

An upper bound on the error of this algorithm can be provided similarly to Section 3.3

Algorithm 3 Splitting Masses Algorithm

1: Start from the list $(p_1, x_1), \cdots, (p_n, x_n)$.
2: Repeat $n - k$ times
3: Find $j = \text{argmin}\{p_i \left(f(x_i) - tf(x_{i_1}) - (1-t)f(x_{i-1})\right) : i \neq 1, n\}$
4: Add $(1-t)p_j$ to p_{j-1} and tp_j to p_{j+1}.
5: Delete (p_j, x_j) from the list.

with a little bit of modification. Consider the error made in each step of the algorithm:

$$\begin{aligned} e_i &= p_i \left(f(x_i) - tf(x_{i+1}) - (1-t)f(x_{i-1})\right) && (3.32)\\ &= -tp_i \left(f(x_{i+1} - f(x_i)\right) + (1-t)p_i \left(f(x_i) - f(x_{i-1})\right) && (3.33)\\ &\leq -tp_i(1-t)\Delta x_i f'(x_{i+1}) + (1-t)p_i t \Delta x_i f'(x_{i-1}) && (3.34)\\ &= p_i t(1-t)\Delta x_i^2 |f''(c_i)|, && (3.35) \end{aligned}$$

where $x_{i-1} \leq c_i \leq x_{i+1}$ and $\Delta x_i \triangleq x_{i+1} - x_{i-1}$. The difference with Section 3.3 is that now $\sum_i \Delta x_i \leq 1$ (not 1/2). On the other hand, for $0 \leq t \leq 1$ we have $t(1-t) \leq \frac{1}{4}$. Therefore, exactly the same results of Section 3.3 can be applied here, and the total error of the algorithm can be upper bounded by $\mathcal{O}\left(\frac{\log(k)}{k\sqrt{k}}\right)$ for the entropy function, and $\mathcal{O}(\frac{1}{k^2})$ for functions that $|f''(x)|$ is bounded.

In the simulations, we measure the maximum achievable rate while keeping the probability of error less than 10^{-3} by finding maximum possible number of channels with the smallest Bhattacharyya parameters such that the sum of their Bhattacharyya parameters is upper bounded by 10^{-3}. The channel is a binary symmetric channel with capacity 0.5. First, we compare 3 different functions $f_1(x) = h(x)$ (entropy function), $f_2(x) = 2\sqrt{x(1-x)}$ (Bhattacharrya function), and $f_3(x) = x(1-x)$ in Algorithms 2 and 3 for degrading and upgrading the channels and compare their performances. We obtain the following results:

$f(x)$	$f_1(x)$	$f_2(x)$	$f_3(x)$
degrade	0.3208	0.3210	0.3022
upgrade	0.3220	0.3218	0.3245

Table 3.1: Achievable rate with error probability at most 10^{-3} for 3 different functions while block-length $N = 2^{12}$ and maximum number of outputs $k = 16$

We can see that the Bhattacharyya function has the best performance in the greedy algorithm, and $f_3(x) = x(1-x)$ the worse. One surprising point is that we prove an upper bound of $\mathcal{O}(\frac{1}{k^2})$ for $f_3(x)$ and $\mathcal{O}(\frac{\log k}{k^{1.5}})$ for the entropy function, but the entropy function has better performance. In the following simulations, we use the Bhattacharrya function for our simulations.

k	2	4	8	16	32	64
degrade	0.2895	0.3667	0.3774	0.3795	0.3799	0.3800
upgrade	0.4590	0.3943	0.3836	0.3808	0.3802	0.3801

Table 3.2: Achievable rate with error probability at most 10^{-3} vs. maximum number of outputs k for block-length $N = 2^{15}$

It is worth restating that the algorithm runs in complexity $\mathcal{O}(k^2N)$. Table 3.2 shows the achievable rates for Algorithms 2 and 3 when the block-length is fixed to $N = 2^{15}$ and k changes in the range of 2 to 64.

It can be seen from Table 3.2 that the difference of achievable rates within the upgraded and degraded version of the scheme is as small as 10^{-4} for $k = 64$. We expect that for a fixed k, as the block-length increases the difference will also increase (see Table 3.3).

n	5	8	11	14	17	20
degrade	0.1250	0.2109	0.2969	0.3620	0.4085	0.4403
upgrade	0.1250	0.2109	0.2974	0.3633	0.4102	0.4423

Table 3.3: Achievable rate with error probability at most 10^{-3} vs. block-length $N = 2^n$ for $k = 16$

However, in our scheme this difference will remain small even as N grows arbitrarily large as predicted by Theorem 2. (see Table 3.4).

n	21	22	23	24	25
degrade	0.4484	0.4555	0.4616	0.4669	0.4715
upgrade	0.4504	0.4575	0.4636	0.4689	0.4735

Table 3.4: Achievable rate with error probability at most 10^{-3} vs. block-length $N = 2^n$ for $k = 16$

We see that the difference between the rate achievable in the degraded channel and upgraded channel gets constant 2×10^{-3} even after 25 levels of polarizations for $k = 16$.

4 Performance of Polar Codes

In this chapter we analyze the performance of polar codes. As mentioned in previous chapters, the performance of polar codes in short length, for example the block-length mainly used in practical situations such as wireless communication, is not very impressive. [2, 10, 11] Therefore, the enhancement of polar codes' performance is an important issue. This chapter is divided into two sections: polar codes over BEC channel and general channel. We use the block error probability as the measure of our performance analysis.

4.1 BEC Channel

In this section, we first investigate the performance of polar codes under successive cancellation decoder in a BEC channel. Later on it becomes more clear why we have restricted ourselves to the BEC channel first. We use the BEC channel with erasure probability 0.5 in our simulations. Figure 4.1 shows the performance of the SC decoder compared to the MAP decoder in the BEC channel.

As one can see there exists a considerable gap between the error probability of the SC decoder and the MAP decoder. Unfortunately, the complexity of a MAP decoder in general (except for the BEC) is exponential in block-length. Therefore, an important question is that whether one can modify the SC decoder such that this gap decreases while we still have a low-complexity decoder. We try to address this question in the following. First, we introduce the notion of a genie-aided decoder.

4.1.1 Genie-aided Decoder

In this part, we define what a genie-aided decoder is. The following notions are mostly developed by Erdal Arıkan.

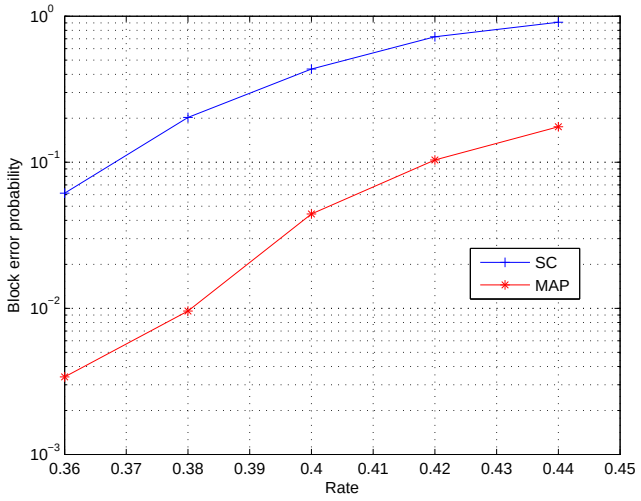

Figure 4.1: Performance of SC decoder and MAP decoder in terms of block error probability, when transmission takes place over a BEC channel with capacity 0.5 and block-length $N = 2^{10}$.

For the moment, consider that we are transmitting data over a general channel. Suppose that you have access to a genie who knows what the correct information bits are. While decoding using the successive cancellation decoder, you might make an error in estimating one of the information bits. Due to the successive nature of the SC decoder, this will cause lots of more errors in decoding the next-coming bits. Now consider requesting help from our genie. One approach known as the *active* genie-aided decoder is that the genie will come to help us as soon as we make an error. Therefore, we correct this error and continue the decoding procedure. We define a k-active genie-aided decoder, a decoder where the genie comes to our help at most k times. We call a decoder to be a *passive* genie-aided decoder if the genie does not come to our help as soon as we make an error. Instead, we are allowed to ask the genie whether this estimated information bit is right or not. We call a decoder to be a k-passive genie-aided decoder if we are allowed to ask the genie about the correctness of at most k bits. So the question that naturally appears here is that how we can find out that it is a good chance that we have made an error in the estimated bit. Obviously, the active and passive genie-aided decoders are the same in the case of BEC channel since no errors will occur in decoding the information bits and we will have only erasures. Active genie-aided decoder cannot be implemented in reality, but a passive genie-aided decoder completely coincides with a list-decoder. Simply instead of asking a genie we proceed our decoding with both 0 and 1 for the estimated bit and put the resulting decoded information-word in our list. Therefore, a k-passive genie-aided decoder is nothing but a list decoder with list size of 2^k.

4.1.2 Polar List-Decoder over BEC Channel

Now we investigate the performance of a k genie-aided SC decoder in a BEC channel with capacity 0.5 and block-length $N = 2^{10}$. In Figure 4.2, the rate is fixed to 0.35. We can see that with a list size of 32, the error will be reduced to order of 10^{-7}. Furthermore, it is interesting to notice that the error probability has approximately an exponential behavior in k. In other words, with one extra help from the genie, the error probability is reduced by an order of magnitude. In Figure 4.3, the rate is fixed to 0.4. As expected, we have

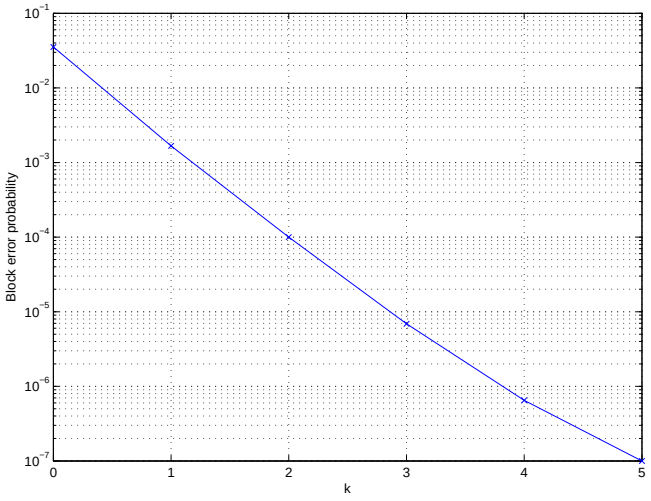

Figure 4.2: Performance of 2^k list-decoder in terms of block error probability, when transmission takes place over a BEC channel with capacity 0.5, block-length $N = 2^{10}$, and $R = 0.35$.

less improvement in error probability with each extra help from the genie but still the exponential behavior of the curve can be seen.

In Figure 4.4, the performance of a simple SC decoder is compared with list-decoders of size 4 and 16 in different rates. Again, it can be seen that with only 2 helps from the genie, the error probability is reduced significantly.

4.2 General Channel

In this section, we consider improving the performance of a polar SC decoder over a general channel, for example the BSC channel with cross-over probability ϵ. The problem with a general channel is that it is not easy to guess in which information bits we have made an error in our estimation (since we do not have only erasures), so it is difficult to implement a passive genie-aided decoder (list-decoder).

Motivated by Forney's framework [8], we try to use the concepts of erasure and feedback to enhance the performance of polar codes. Therefore, an idea to improve the performance

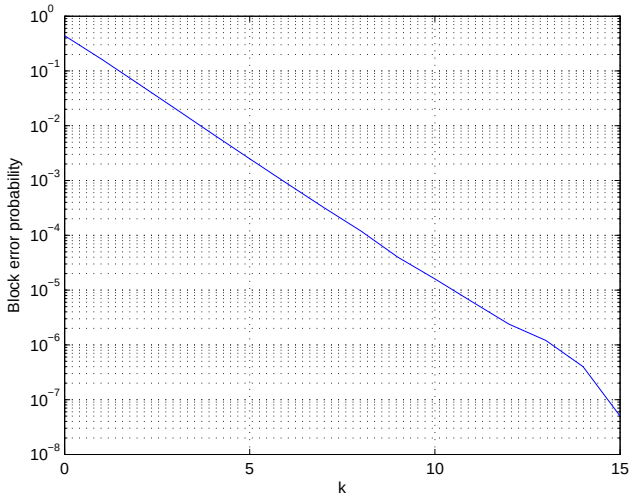

Figure 4.3: Performance of 2^k list-decoder in terms of block error probability, when transmission takes place over a BEC channel with capacity 0.5, block-length $N = 2^{10}$, and $R = 0.4$.

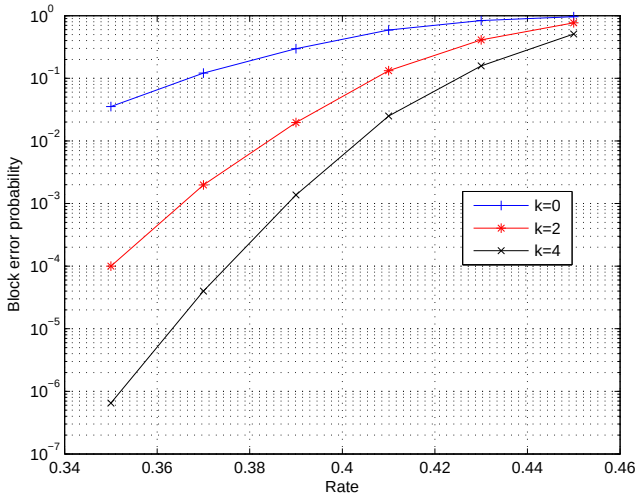

Figure 4.4: Performance of 2^k list-decoder in terms of block error probability, when transmission takes place over a BEC channel with capacity 0.5 and block-length $N = 2^{10}$.

of a polar SC decoder is to determine whether we have estimated a whole block with error or not instead of being worried about each information bit. If we can detect our block errors with high probability, then we can erase the block and retransmit using an ARQ system to achieve better performance. In the following, we will discuss two methods for detecting our errors, and then combine them and use them in a HARQ system mentioned

later.

4.2.1 Typicality Check

Considering that we have estimated our information bits correctly, we expect the sequence of the outputs with respect to the conditional probability distribution on our codeword to be a typical sequence. More precisely, we expect $\frac{1}{N}\log\frac{1}{p(\mathbf{y}|\mathbf{x})}$ to be close to $H(Y|X)$ (which is $h(\epsilon)$ in the BSC channel). Therefore, one method to detect whether we have made an error in our estimation or not is to re-encode the estimated information bits $\hat{\mathbf{u}}$ in the decoder and compute the estimated codeword $\hat{\mathbf{x}}$. Then we can declare an erasure if

$$\frac{1}{N}\log\frac{1}{p(\mathbf{y}|\hat{\mathbf{x}})} - H(Y|X) \geq t, \tag{4.1}$$

where t is a threshold value which needs to be adjusted properly. Obviously, we have a trade-off between "false" erasure probability and error probability. By false erasure probability we mean that we had estimated the codeword right but declared an erasure. We can provide a theoretical bound on this quantity.

Lemma 3. *The false erasure probability decays exponentially in block-length.*

Proof. We prove the lemma for a BSC channel with cross-over probability ϵ. Considering that the estimated codeword is true, the false erasure probability is

$$P_{false} = Pr\Big(\frac{1}{N}\log\frac{1}{p(\mathbf{y}|\mathbf{x})} - h(\epsilon) \geq t\Big), \tag{4.2}$$

where $h(\epsilon) = -\epsilon\log\epsilon - (1-\epsilon)\log(1-\epsilon)$. Let us define a new random variable

$$V_i \triangleq \frac{1}{\log\frac{1-\epsilon}{\epsilon}}\Big(\log\frac{1}{p(y_i|x_i)} - \log\frac{1}{1-\epsilon}\Big). \tag{4.3}$$

Clearly, V_i is a Bernouli random variable with parameter ϵ ($\mathbb{E}(V_i) = \epsilon$). Rewriting (4.2) in terms of V_i, we have

$$P_{false} = Pr\Big(\frac{1}{N}\sum_{i=1}^{N} V_i - \mathbb{E}(V_i) \geq \frac{t}{\log\frac{1-\epsilon}{\epsilon}}\Big) \tag{4.4}$$

$$\leq exp\Big(-ND(\epsilon + \frac{t}{\log\frac{1-\epsilon}{\epsilon}}||\epsilon)\Big), \tag{4.5}$$

where $D(x||y) = x\log\frac{x}{y} + (1-x)\log\frac{1-x}{1-y}$ is the Kullback-Leibler divergence between Bernoulli distributed random variables with parameters x and y respectively, and (4.5) is the Chernoff bound for a sequence of i.i.d. Bernoulli random variables.

□

In the following, we show that the error probability of the typicality-check algorithm is highly correlated to the distance distribution of polar codes. Suppose that we have estimated some of the information bits incorrectly. Without loss of generality we suppose that codeword $\mathbf{0}$ is sent. Then,

$$\hat{\mathbf{x}} = \hat{\mathbf{u}}\mathbf{G} = (\mathbf{u} + \mathbf{e})\mathbf{G} = \mathbf{e}\mathbf{G}, \tag{4.6}$$

where $\mathbf{e}$ is the error vector. Considering the rows of generator matrix $\mathbf{G}$, where $\mathbf{G} = [\mathbf{g_1}, \mathbf{g_2}, \ldots, \mathbf{g_N}]^T$, we have

$$\hat{\mathbf{x}} = \sum_{i \in \mathcal{I}_e} \mathbf{g_i}, \tag{4.7}$$

where $\mathcal{I}_e$ is the set of information bits indices that have been estimated with error. Therefore, the number of positions where $p(y_i|\hat{x}_i)$ differs from its true value $p(y_i|x_i)$ is nothing but the hamming weight of the vector $\sum_{i \in \mathcal{I}_e} \mathbf{g_i}$. Now the relationship between the error probability of the algorithm and the distance distribution of polar codes is getting clear. If the minimum distance of polar codes was $\mathcal{O}(N)$, we could have immediately proved that the algorithm detects all the errors and declares erasures as N grows large. Unfortunately, this is not the case as the hamming distance of polar codes is $\mathcal{O}(\sqrt{N})$. On the other hand, in the SC polar decoder usually many errors occur after the first information bit is estimated wrongly. So the number of indices in $\mathcal{I}_e$ is large and we know that there are few codewords with distance of close to d_{min} from the distance distribution of polar codes so this implies that the algorithm does not let any errors escape with high probability as N grows large, but it appears to be difficult to provide a rigorous mathematical proof.

Tables 4.1 and 4.2 show the performance of the typicality-check algorithm for 2 different values of t. In Table 4.1, t is chosen such that no false erasure is declared in 10^5 trials of the simulation. The false erausre probability is

$$P_{false} = P_{erase} + P_e - P_{SC} \tag{4.8}$$

One can roughly say that 70-percent of the errors of the SC decoder are detected and declared as erasures without declaring any false erasures. In Table 4.2, the threshold t is chosen less conservatively and a few false erasure declarations are allowed. We can see that around 95-percent of the errors are detected, while missing 2-percent of the previously correctly-detected blocks. (declaring them falsely as erasures) Also note that the parameter t, may also be tuned for different rates. A good approach will be Neyman-Pearson type of argument, i.e., choose t to minimize error probability P_e, subject to the constraint that $P_{false} \leq \tau$, and τ is set by the application.

4.2.2 Random Parity Constraints

In this section, we propose another method to detect the block errors made in decoding the output. The method is to add p random parity check bits to the information bits. Therefore, the actual rate is

$$r = \frac{k - p}{N}. \tag{4.9}$$

R	0.35	0.37	0.39	0.41	0.43
P_e	0.0168	0.0336	0.0603	0.1043	0.1436
P_{erase}	0.0612	0.1121	0.2171	0.3232	0.4644
P_{SC}	0.0780	0.1457	0.2774	0.4275	0.6080

Table 4.1: Performance of typicality-check algorithm with threshold value $t = 0.1$, when transmission takes place over a BSC channel with capacity 0.5 and block-length $N = 2^{10}$

R	0.35	0.37	0.39	0.41	0.43
P_e	0.0018	0.0048	0.0093	0.0273	0.0463
P_{erase}	0.1098	0.1645	0.2780	0.4031	0.5619
P_{SC}	0.0779	0.1456	0.2774	0.4275	0.6080

Table 4.2: Performance of typicality-check algorithm with threshold value $t = 0.05$, when transmission takes place over a BSC channel with capacity 0.5 and block-length $N = 2^{10}$

As one can see, we have sacrificed the rate by $\frac{p}{N}$. The encoding process is the following. First, $k - p$ information bits will be encoded to k bits by the parity generator matrix $\mathbf{G^{(p)}}_{(k-p)\times k}$, and then the k bits is sent to the polar encoder to form the N bits codeword. The polar decoder estimates the k bits $\hat{\mathbf{u}}$, then checks the parity-check constraint:

$$\mathbf{H}_{p\times k}\hat{\mathbf{u}} = \mathbf{0}, \tag{4.10}$$

where $\mathbf{H}$ is the parity-check matrix.

Lemma 4. *Using a random parity-check matrix, the error probability of the SC polar decoder is reduced to $\frac{P_{SC}}{2^p}$ in expectation, where P_{SC} is the block error probability of a SC polar decoder and p is the number of parity-check bits.*

Proof. Without loss of generality, we can assume that codeword $\mathbf{0}$ is sent. Define $\alpha_{\mathbf{i}}$ and $1 \leq i \leq 2^k - 1$, the sequence of all non-zero binary vectors of length k. We can write the

expectation of the block error probability as the following:

$$P_e = \mathbb{E}_{\mathbf{H}}\left(Pr\left(\mathbf{H}\hat{\mathbf{u}} = \mathbf{0}, \hat{\mathbf{u}} \neq \mathbf{0}\right)\right) \tag{4.11}$$

$$= \mathbb{E}_{\mathbf{H}}\left(\sum_{i=1}^{2^k-1} Pr\left(\hat{\mathbf{u}} = \alpha_{\mathbf{i}}\right) Pr\left(\mathbf{H}\alpha_{\mathbf{i}} = 0\right)\right) \tag{4.12}$$

$$= \sum_{i=1}^{2^k-1} Pr\left(\hat{\mathbf{u}} = \alpha_{\mathbf{i}}\right) \mathbb{E}_{\mathbf{H}}\left(Pr\left(\mathbf{H}\alpha_{\mathbf{i}} = 0\right)\right) \tag{4.13}$$

$$= \sum_{i=1}^{2^k-1} Pr\left(\hat{\mathbf{u}} = \alpha_{\mathbf{i}}\right) \prod_{j=1}^{p} \mathbb{E}_{\mathbf{H}}\left(Pr\left(\mathbf{h_j}.\alpha_{\mathbf{i}} = 0\right)\right) \tag{4.14}$$

$$= \frac{1}{2^p} \sum_{i=1}^{2^k-1} Pr\left(\hat{\mathbf{u}} = \alpha_{\mathbf{i}}\right) \tag{4.15}$$

$$= \frac{P_{SC}}{2^p}, \tag{4.16}$$

where (4.15) is due to the fact that the entries of random parity-check matrix $\mathbf{H}$ are i.i.d. Bernoulli random variables with parameter $\frac{1}{2}$.

□

Theorem 3. *Using a random parity-check algorithm, we can achieve an error exponent* $\frac{N}{\log N}$ *and erasure exponent* $\sqrt{N}$ *without any rate loss.*

Proof. Using Lemma 4, let p be $\mathcal{O}(\frac{N}{\log N})$. Then, the rate would be

$$r = \frac{k - \frac{N}{\log N}}{N} = \mathcal{O}(\frac{k}{N}) \tag{4.17}$$

So we do not have a rate loss asymptotically. We know that P_{SC} is asymptotically $\mathcal{O}(2^{-\sqrt{N}})$[7]. Therefore, by Lemma 4.2.2 the error probability is asymptotically

$$P_e = \mathcal{O}(\frac{2^{-\sqrt{N}}}{2^{\frac{N}{\log N}}}) = \mathcal{O}(2^{-\frac{N}{\log N}}) \tag{4.18}$$

The erasure probability is

$$P_{erase} = P_{SC} - P_e = \mathcal{O}(2^{-\sqrt{N}}) \tag{4.19}$$

□

In Table 4.3, we see the simulation results of the parity-check algorithm. The polar code rate is 0.4, but as mentioned previously we have some rate loss due to the use of parity-check bits. The error probability of the SC decoder is the error probability while sending with the "actual" rate. The validity of Lemma 4 is shown numerically in Table 4.3.

p	3	4	5	6	7	8
R	0.3965	0.3955	0.3945	0.3936	0.3926	0.3916
P_e	0.0448	0.0225	0.0104	0.0059	0.0.0033	0.0017
P_{erase}	0.3027	0.3249	0.3371	0.3416	0.3442	0.3458
P_{SC}	0.3298	0.3207	0.3090	0.3024	0.2976	0.2806

Table 4.3: Performance of parity-check algorithm with p bits of parity , when transmission takes place over a BSC channel with capacity 0.5 and block-length $N = 2^{10}$ and the polar code rate is 0.4. ($P_{SC} = 0.3475$)

4.2.3 Combined Algorithm

In this section, we combine the algorithms mentions in sections 4.2.1 and 4.2.2 to provide a powerful tool for detecting the block errors in a general channel while using polar decoder and encoder. In fact, we declare erasures if either the parity-check or the typicality-check fails. The interesting problem here is that we have two parameters t and p to tune. Similarly to Lemma 4, it is easy to prove that for the combined algorithm the error probability will be reduced by a factor of 2^p with respect to the error probability of the typicality-check algorithm. ($P_e = \frac{P_{e,typical}}{2^p}$) One interesting question in this regard is that whether combining the algorithms helps at all. In fact, both of the algorithms are somehow detecting most of the block errors of the SC decoder, in the expense of missing some of the correct blocks, or alternatively some rate loss. Therefore, one algorithm may totally outperform the other one. The intuitive answer to this question is that combining the two algorithms does help the performance since each of them are detecting errors through different underlying logic. The parity-check algorithm detects all of the errors except a fraction of $\frac{1}{2^p}$ on average, and its success has nothing to do with the "quality" of the received data. On the other hand, the typicality-check algorithm detects what we call "strong" noises (which are the non-typical ones indeed) and miss the errors due to "mild" noises. Therefore, combining the two algorithms has the advantage that even in the case of having a mild noise we detect all the errors other than a small fraction of $\frac{1}{2^p}$. Another interesting engineering problem here is how to tune the parameters p and t to design a polar code with certain error probability and erasure probability requirements. In other words, how should we share the weight of detecting errors between these two algorithms to have the optimum error and erasure probabilities. In the following, we state an example of polar code design.

Example 1. *Using the typicality-check and random-parity-check algorithms, design a polar code of block-length $N = 2^{10}$ with maximum possible rate over a BSC channel of capacity 0.5 such that the following conditions are satisfied:*

(i) $P_e \leq 10^{-4}$

(ii) $P_{erase} \leq 0.05$

As one can see, the problem is nothing but an optimization over 3 parameters of R, t, and p. What we want to maximize is $R-\frac{p}{N}$. Obviously, the solution of the problem is only through numerical analysis. One suboptimal method to solve the problem in simpler way is the following. A logical approach is to set t to the minimum possible value such that we do not have any false erasures caused by the typicality-check algorithm. (The parity-check algorithm does not declare any incorrect erasures.) The problem is that optimizing the threshold value t is itself dependent on the rate so it not obvious that for which rate we should locally optimize t. From the results of Table 4.1 and 4.2, one can see that the achievable rate is certainly less than 0.35. Let us fix an optimum t for the rate $R = 0.3$. The obtained value for t is $t = 0.11$. Fixing t, it remains to find a suitable p to satisfy the problem's conditions. Again the dependency to R exists. After some ad-hoc and simultaneous tuning of these two parameters, we set $p = 5$, and achieve actual rate of 0.324 with $P_e = 5 \times 10^{-5}$ and $P_{erase} = 0.04$. One can now optimize t for the new obtained rate and re-do the procedure.

4.2.4 A Hybrid ARQ Algorithm

In this section, we state a practical algorithm for retransmitting the data when erasure has occurred and we provide some simulation results for the throughput of the H-ARQ algorithm based on polar codes. We see that using the H-ARQ algorithm we achieve great improvement in the performance of polar codes which is very important especially in short block-lengths for practical purposes.

The main idea of the algorithm is to retransmit the same block (this method is known as chase combining), and add the log-likelihood ratios (LLRs) obtained in the output of channels to the previous LLRs and run the polar decoder with the new LLR values. A more detailed description of the algorithm for transmitting one block is the following: For the first attempt, send the corresponding codeword as usual. Run the polar decoder while saving the channel LLRs in a buffer. If no erasure is declared, the transmission is finished. If an erasure is detected, retransmit the same codeword and add the new channel LLRs with the previous LLRs. Then, run the polar decoder with these new LLRs and proceed as before. To make implementing the algorithm feasible, one should set a maximum number of attempts allowed for retransmission. We set this number to 4 in our simulations.

It is well-known that the important factors in measuring the performance of an ARQ system is its throughput and error probability. (Suppose that we have let enough re-transmissions such that the erasure probability is negligible.) Recall that throughput is the average rate of a successful frame delivery. First, let us find an upper bound on the achievable throughput in our framework. Suppose that a genie has proposed us an algorithm which detects all of the block errors in the SC decoder without declaring any false erasure and any rate loss. Furthermore, suppose that we retransmit our data so wisely that we always succeed to decode our message in our second attempt without any errors. The normalized expectation of our transmission time will be

$$\mathbb{E}(T) = 1 + P_{SC}(R). \tag{4.20}$$

Therefore, the throughput η is upper bounded by

$$\eta \leq \frac{R}{1 + P_{SC}(R)}. \tag{4.21}$$

Figure 4.5 shows the function in (4.21) versus R for polar codes of block-length 2^8 over a BSC channel with capacity 0.5. It can be seen that the maximum throughput roughly equals 0.322 which is obtained around $R = 0.37$.

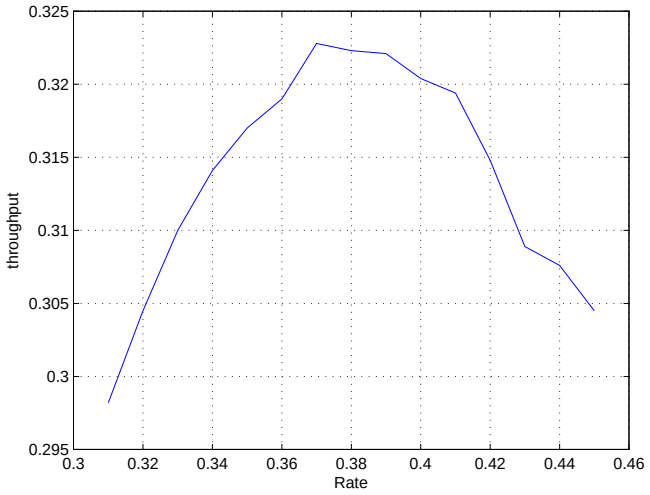

Figure 4.5: Upper-bound on the throughput achievable in our framework with polar code of block-length $N = 2^8$.

Motivated by this result, we test our H-ARQ algorithm for block-length 2^8 at $R = 0.37$. After tuning parameters p and t, we can achieve throughput $\eta = 0.291$ with error probability of $P_e = 10^{-4}$ and negligible erasure probability. One can see that our result is not far from the upper bound shown in Figure 4.5. Furthermore, it is interesting to mention that using a simple SC decoder the same error probability and rate (vs throughput) can roughly be achieved with a block-length of 2^{11}.

5 Conclusion

5.1 Summary of Contributions

Polar coding is a recently discovered coding technique which is capable of achieving the symmetric capacity of binary discrete memoryless channels. Polar codes have low complexity encoding and decoding ($\mathcal{O}(N \log N)$), where N is the blocklength of the code. Furthermore, for any rate below capacity the block error probability of polar codes decay to zero like $\mathcal{O}(2^{-\sqrt{N}})$.

Since polar coding is a new technique, there are still many open questions about polar codes. In this thesis, we address two problems which are important for implementing polar codes in practice:

1. Construction of Polar Codes: Since the output alphabet size of synthesized channels through channel polarization grows exponentially in blocklength, computing the exact transition probabilities of these channels seem to be intractable. Therefore, one needs to come up with an efficient algorithm to approximate these channels such that the approximated set of good indices are close to the true good indices. In this thesis, we followed the approach of Tal and Vardy [3] to quantize the channels, and provided some algorithms which can be analyzed for complexity and accuracy. We showed that using an algorithm which has almost linear complexity in blocklength, one can find all the good channels except a negligible fraction.

2. Performance of Polar Codes: Polar codes in their current format, fall short in terms of performance in comparison with other powerful codes like LDPC. However, polar codes have many advantages, mainly its low-complexity encoding and decoding procedures, as well as its analytical tractability. An interesting question is to come up with modified decoding algorithms such that the performance is improved while

keeping the low complexity decoder. In this thesis, we provided different algorithms for different channels to enhance the performance of polar decoder. For the BEC, using the nature of the channel that no errors occur in estimating the information bits (but only erasures), we introduced the notion of genie-aided decoder, and provided numerical results for the performance of list-decoders over BEC with different list sizes. For general channel, motivated by Forney's seminal work on erasure and feedback schemes, we introduced two algorithms, the typicality-check algorithm and the parity-check constraints algorithm to detect our block errors and declare erasures. Then, we combined these two algorithms and came up with a Hybrid ARQ scheme which significantly improves the performance of polar codes considering the throughput of the system.

5.2 Future Work

In the following, we motivate some important problems as extensions of our work.

- In Chapter 3, we introduced several greedy and low-complexity algorithms for quantizing the channels. The algorithms are indeed suboptimal. An important question is that how far these algorithms are from the optimal algorithm. Can we come up with a low-complexity algorithm which is close in terms of performance to the optimal one? Furthermore, it is interesting to investigate that how changing the function $f(x)$ in Algorithms 1 and 2, affect the performance of the algorithms. Can we come up with an optimum function in the greedy algorithms?

- In Chapter 4, we provided different methods and algorithms for enhancing the performance of polar decoders. The results in this chapter were mainly supported by numerical evidence. Therefore, there are many open problems in the analysis of the performance of these algorithms. Some important questions in this regard are the following:

  (i) What is the error exponent of a polar list-decoder over the BEC channel? Can we relate it to the exponents derived in the seminal works of Gallager and Forney? [8, 9]

  (ii) How can we find out if we have made an error in estimating one of the information bits in a general channel?

  (iii) Can we analytically prove that the typicality check algorithm detects all of the block errors asymptotically? How can we relate the performance of this algorithm to the weight distribution of polar codes?

  (iv) Considering an HARQ system, what is the optimal strategy for sending the packet in our second attempt? Is it optimal to resend the whole block?

Bibliography

[1] E. Arıkan, "Channel Polarization: A Method for Constructing Capacity-Achieving Codes for Symmetric Binary-Input Memoryless Channels," *IEEE Trans. Inf. Theory*, vol. 55, no. 7, pp. 3051–3073, Jul. 2009.

[2] S. B. Korada, "Polar Codes for Channel and Source Coding," Ph.D. dissertation, EPFL, Lausanne, Switzerland, Jul. 2009.

[3] I. Tal and A. Vardy, "How to Construct Polar Codes," *talk given in Information Theory Workshop*, Dublin, Aug. 2010. (also available on arxiv)

[4] S. H. Hassani, S. B. Korada, and R. Urbanke, "The Compound Capacity of Polar Codes," *Proceedings of Allerton Conference on Communication, Control and Computing*, Allerton, Sep. 2009.

[5] R. Mori and T. Tanaka, "Performance and Construction of Polar Codes on Symmetric Binary-Input Memoryless Channels," *Proceedings of ISIT*, Seoul, South Korea, Jul. 2009, pp. 1496–1500.

[6] T. Richardson and R. Urbanke, "Modern Coding Theory," Cambridge University Press, 2008.

[7] E. Arıkan, E. Telatar, "On the Rate of Channel Polarization," *Proceedings of IEEE International Symposium on Information Theory*, Seoul, Jul. 2009.

[8] G. D. Forney, Jr., "Exponential Error Bounds for Erasure, List and Decision Feedback Schemes," *IEEE Trans. Inf. Theory*, vol. IT-14, no. 2, pp. 206–220, Mar. 1968.

[9] R. G. Gallager, "Information Theory and Reliable Communication," New York: Wiley, 1968.

[10] N. Hussami, S. B. Korada, and R. Urbanke, "Performance of Polar Codes for Channel and Source Coding," *Proceedings of IEEE International Symposium on Information Theory*, Seoul, Jul. 2009.

[11] E. Arıkan, "A Performance Comparison of Polar Codes and Reed-Muller Codes," *IEEE Communications Letter*, vol. 12, no. 6, 2008.

[12] T. M. Cover and , J. A. Thomas, "Elements of Information Theory," New York: Wiley, 1991.

[13] R. Pedarsani, S. H. Hassani, I. Tal, and E. Telatar "On the Construction of Polar Codes," *Proceedings of IEEE International Symposium on Information Theory*, Saint Petersburg, Jul. 2011.