5-2013

A Non-Asymptotic Approach to the Analysis of Communication Networks: From Error Correcting Codes to Network Properties

Ali Eslami
University of Massachusetts Amherst, eslami.ali@gmail.com

A NON-ASYMPTOTIC APPROACH TO THE ANALYSIS OF COMMUNICATION NETWORKS: FROM ERROR CORRECTING CODES TO NETWORK PROPERTIES

A Dissertation Presented

by

ALI ESLAMI

Submitted to the Graduate School of the
University of Massachusetts Amherst in partial fulfillment
of the requirements for the degree of

DOCTOR OF PHILOSOPHY

May 2013

Electrical and Computer Engineering

A NON-ASYMPTOTIC APPROACH TO THE ANALYSIS OF COMMUNICATION NETWORKS: FROM ERROR CORRECTING CODES TO NETWORK PROPERTIES

A Dissertation Presented

by

ALI ESLAMI

Approved as to style and content by:

Hossein Pishro-Nik, Chair

Erik Learned-Miller, Member

Dennis Goeckel, Member

Patrick Kelly, Member

Robert W. Jackson, Department Chair
Electrical and Computer Engineering

To my dear parents.

ACKNOWLEDGMENTS

Before anything, I would like to express my sincere gratitude to my parents who have always been a source of love and inspiration throughout my life. Without their support and guidance I would have never been at the point where I stand now.

I would also like to thank my advisor Professor Pishro-Nik for his support during my studies. I am also grateful to Professor Goeckel, both a great teacher and researcher, for his valuable comments on this dissertation. Additionally, I would like to thank the other members of my dissertation committee, Professor Kelly and Professor Learned-Miller, for their valuable insights.

This work was supported by the National Science Foundation under grants CCF-0728970, CCF-0830614, and ECCS-0636569.

ABSTRACT

A NON-ASYMPTOTIC APPROACH TO THE ANALYSIS OF COMMUNICATION NETWORKS: FROM ERROR CORRECTING CODES TO NETWORK PROPERTIES

MAY 2013

ALI ESLAMI

B.Sc., SHARIF UNIVERSITY OF TECHNOLOGY

M.Sc., SHARIF UNIVERSITY OF TECHNOLOGY

Ph.D., UNIVERSITY OF MASSACHUSETTS AMHERST

Directed by: Professor Hossein Pishro-Nik

This dissertation has its focus on two different topics: 1. non-asymptotic analysis of polar codes as a new paradigm in error correcting codes with very promising features, and 2. network properties for wireless networks of practical size. In its first part, we investigate properties of polar codes that can be potentially useful in real-world applications. We start with analyzing the performance of finite-length polar codes over the binary erasure channel (BEC), while assuming belief propagation (BP) as the decoding method. We provide a stopping set analysis for the factor graph of polar codes, where we find the size of the minimum stopping set. Our analysis along with bit error rate (BER) simulations demonstrates that finite-length polar codes show superior error floor performance compared to the conventional capacity-approaching coding techniques. Motivated by good error floor performance, we introduce a mod-

ified version of BP decoding while employing a guessing algorithm to improve the BER performance.

Each application may impose its own requirements on the code design. To be able to take full advantage of polar codes in practice, a fundamental question is which practical requirements are best served by polar codes. For example, we will see that polar codes are inherently well-suited for rate-compatible applications and they can provably achieve the capacity of time-varying channels with a simple rate-compatible design. This is in contrast to LDPC codes for which no provably universally capacity-achieving design is known except for the case of the erasure channel. This dissertation investigates different approaches to applications such as UEP, rate-compatible coding, and code design over parallel sub-channels (non-uniform error correction). Furthermore, we consider the idea of combining polar codes with other coding schemes, in order to take advantage of polar codes' best properties while avoiding their shortcomings. Particularly, we propose, and then analyze, a polar code-based concatenated scheme to be used in *Optical Transport Networks* (OTNs) as a potential real-world application.

The second part of the dissertation is devoted to the analysis of *finite* wireless networks as a fundamental problem in the area of wireless networking. We refer to networks as being finite when the number of nodes is less than a few hundred. Today, due to the vast amount of literature on large-scale wireless networks, we have a fair understanding of the asymptotic behavior of such networks. However, in real world we have to face finite networks for which the asymptotic results cease to be valid. Here we study a model of wireless networks, represented by random geometric graphs. In order to address a wide class of the network's properties, we study the threshold phenomena. Being extensively studied in the asymptotic case, the threshold phenomena occurs when a graph theoretic property (such as connectivity) of the network experiences rapid changes over a specific interval of the underlying parameter. Here, we

find an upper bound for the threshold width of finite line networks represented by random geometric graphs. These bounds hold for all monotone properties of such networks. We then turn our attention to an important non-monotone characteristic of line networks which is the Medium Access (MAC) layer capacity, defined as the maximum number of possible concurrent transmissions. Towards this goal, we provide a linear time algorithm which finds a maximal set of concurrent non-interfering transmissions and further derive lower and upper bounds for the cardinality of the set. Using simulations, we show that these bounds serve as reasonable estimates for the actual value of the MAC-layer capacity.

Keywords: Polar Codes, Channel Capacity, Rate-Compatible Codes, Non-Uniform Coding, Unequal Error Protection, Concatenated Codes, Belief Propagation, Random Geometric Graphs, Monotone Properties, Threshold Phenomena, Percolation Theory, Finite Wireless Networks, Connectivity, Coverage, MAC-Layer Capacity.

TABLE OF CONTENTS

Page

LIST OF FIGURES

Figure **Page**

CHAPTER 1

INTRODUCTION

Our goal here is to take a step towards analyzing practical models of communication systems and wireless networks. One of the most important features of real-life communication systems is that they all function in the finite regime, meaning that everything is finite in such systems. For instance, transmission power, bandwidth, packet length, code-length, and number of nodes in the network are all finite. There currently exist a vast amount of literature on the asymptotic analysis of various aspects of communication systems. Asymptotic results are very important for two reasons. First, they give us good estimates for large-scale systems. Second, they show some fundamental trade-offs in the underlying communication system. However, as we will see later in this dissertation, the asymptotic results cease to be valid for practical systems in the finite regime. Thus, it is very crucial from the practical point of view to perform a non-asymptotic analysis of such systems. These analytic results will essentially help us to understand, design, and analyze real-life communication systems, and also to design more suitable communication protocols.

The question that arises here is, can we do small-scale analysis? We recognize some obstacles as follows. First, in large-scale analysis we can use asymptotic estimates that make the analysis much simpler. These estimates are not available in small-scale analysis. Among these estimations is ignoring the constant factors in the analysis. This is possible in large-scale analysis since we only care about the parts of the formula that grow with size of the network. Hence, the constant factors are often ignored. However, these constant factors could become significant when dealing with

networks of moderate sizes with a few hundred nodes. Thus, small-scale analysis is usually more difficult. Second, even if we can perform the small-scale analysis, we usually obtain very complicated formulas that are not very useful practically. In this dissertation, we want to circumvent these problems and provide guidelines for small scale-analysis. We assume the reader is familiar with large-scale (asymptotic) analyses to some extent.

Here, we deal with small scale-analysis for two important topics in communications: first, error correcting codes as an essential building block of any communication systems, and second, network properties for wireless networks. Among all error correcting coding schemes, we investigate finite-length performance of *polar codes* as the first class of provably capacity achieving codes for symmetric binary-input discrete memoryless channels (B-DMCs) with low encoding and decoding complexity. We will then focus on analyzing small and moderate-size wireless networks as we face in real-world applications. Below, we provide a short background on these topics and list our contributions to each.

1.1 Results on Polar Codes

As a new paradigm, polar codes proved to be very powerful in theory. Since their introduction, polar codes proved their capability to solve some problems (sometimes open problems) that could not be handled using other coding schemes. For example, as we see in this dissertation, polar codes can solve the problem of designing universally capacity-achieving rate-compatible codes. These facts suggest that coding schemes based on polarization techniques can have great potentials to be used in practice. However, in most cases, polar codes cannot still compete with the existing more mature coding schemes from a practical point of view. An important example is the finite-length error correction performance where polar codes show significant disadvantages compared to conventional schemes such as turbo codes and LDPC

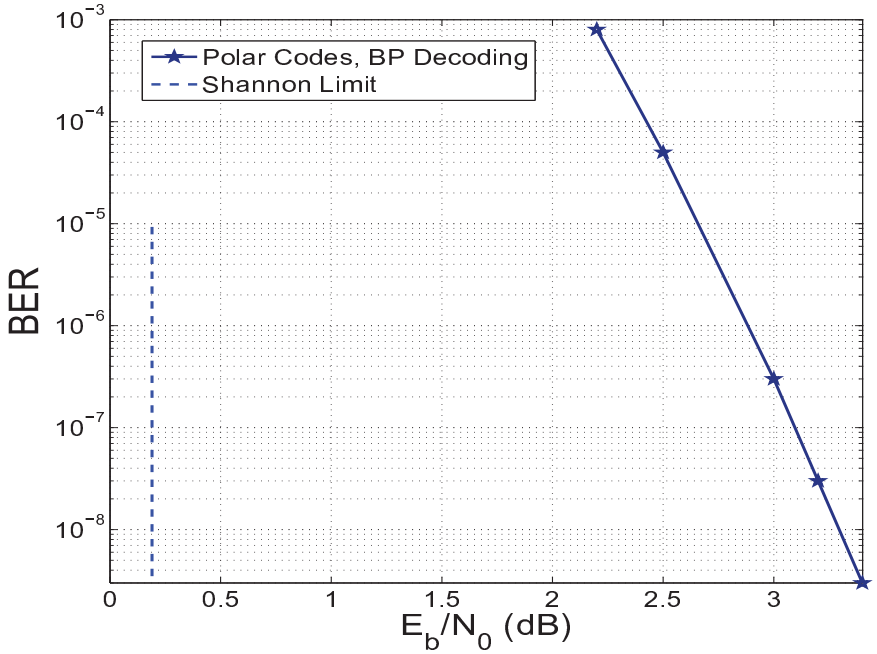


Figure 1.1. BER performance of polar codes under Belief Propagation (BP) decoding over the binary Gaussian channel, compared to the Shannon limit for error correcting codes. The code-length and code-rate are 2^{13} and 1/2, respectively. As you can see, polar codes stay far from the capacity when used in finite block lengths.

codes. In fact, the research in this area has often been theoretical in the sense that asymptotic characteristics of these codes have been investigated. Here, we are mainly concerned with closing the gap between the powerful theoretical advantages of polar codes and their performance in real-world applications.

Polar codes and the core concept of channel polarization in fact open new horizons in coding theory. However, there still exist some drawbacks when it comes to practicality. Probably, the most important issue is their performance in finite regime. While polar codes are asymptotically capacity achieving, their Bit Error Rate (BER) performance in the finite-length stays far from the capacity. Fig. 1.1 shows the BER performance of a polar code of length 2^{13} and rate 0.5. As it can be seen, the performance is far from the capacity. For polar codes to be used in practical scenarios, this distance needs to be reduced.

1.1.1 Finite-Length Analysis of Polar Codes

To improve finite-length performance, it is first critical to obtain a comprehensive understanding of the finite-length behavior of polar codes. In this dissertation, we have studied the performance of polar codes over the binary erasure channel by considering stopping sets in the factor graph of polar codes. We prove several theorems regarding the structure and size of the stopping sets in polar codes' graph. Particularly, the size of the minimum stopping set (stopping distance) is obtained for polar codes.

1.1.2 Improved Decoding for Polar Codes

In order to address the currently poor error rate performance of polar codes, we propose improved decoding schemes based on the finite analysis. One example of such schemes uses the simple idea of guessing the value of some of the undecoded bits. This scheme continues the decoding process until it faces a contradiction. Then it flips the value of the guessed bits. If there is no other contradiction, then the guessed values are correct. As it will be shown, this scheme can result in a significant improvement in BER while keeping the decoding complexity almost the same.

1.1.3 Rate-Compatible Polar Codes

Another important question is to identify practical scenarios for which polar codes can provide significant advantages over other coding schemes. In this dissertation, we discuss that polar codes can be designed as universally capacity-achieving rate-compatible codes for symmetric B-DMCs. As we will see, this design employs a rate-compatible encoder with minimal complexity and is based on the inherent characteristics of polar codes. This is very interesting because even Raptor codes and punctured LDPC codes are only proved to be universally capacity-achieving over the erasure channel. As another approach to rate-compatible codes, we study different puncturing schemes for polar codes while we compare the performance-versus-

complexity trade-off for these schemes to our aforementioned universally capacity-achieving scheme.

1.1.4 Non-Uniform Polar Codes and Unequal Error Protection

Another important application which is closely related to the rate-compatible schemes is coding for "non-uniform channels". Non-uniform coding schemes use one encoder-decoder pair to transmit data over a set of parallel channels. Indeed, some applications, such as volume holographic memory (VHM) systems, can be modeled using a set of parallel channels with different channel parameters [3,4]. Here, we investigate the potentials that may be found in polar codes for this application. Benefiting from the results of finite analysis of polar codes, one can design codes specialized for non-uniform channels. Our finite analysis reveals that different code-bits undergo different degree of protection dictated by the code's factor graph. As we will see, this fact can be used to design "non-uniform polar codes". We will present the results for a simple design of polar codes based on this idea, showing the improvement achieved over the case of using separate codes for different sub-channels. Another related problem is unequal error protection, where the code is designed such that a specific subset of the information bits (usually a small fraction) is better protected (have less BER). In finite-length polar codes, different information bits face different channels by the design. As opposed to the asymptotic case, these channels maintain a wide range of error rates, from close to zero to close to one. This brings up an interesting question: is it possible to benefit from this property to design unequal error protection codes based on polar codes.

1.1.5 Concatenated Polar Codes

Polar codes show many advantages over the conventional coding techniques, while they suffer from some disadvantages. For instance, as we will see, polar codes maintain poor finite-length error correction performance while showing impressive error

floor performance. A promising approach to exploit these advantages while compensating for the shortcomings of polar codes, is to use them in a hybrid fashion, i.e. in combination with other coding schemes. By carefully designing such a combination, the two codes act as complements leading to a significant improvement over the case of using each code stand alone. Therefore, a critical step in improving the practical aspects of polar codes is a detailed study of their performance in combination with other known coding schemes. In this research, we consider different combinations of this type while we analyze the effectiveness of each of them and the role that each coding scheme plays in such a combination.

An example of a hybrid scheme is concatenation. In fact, in many real-world applications, a "concatenation" of two or more coding schemes is used [5]. As a first step in this research direction we propose a polar-LDPC concatenated scheme. While finite-length LDPC codes are very successful in achieving low BERs, they usually suffer from the error floor problem. On the other hand, as it is shown by our results in finite analysis, polar codes show a good error floor performance. Hence, we propose a concatenated polar-LDPC scheme. We observe that this scheme shows a significant improvement over the original polar coding scheme, without showing any sign of error floor, as opposed to the capacity approaching LDPC codes. The proposed scheme also maintains the same decoding complexity as of polar codes. This hybrid scheme is in fact designed for "Optical Transport Networks" as a real-world application, and beats the BER performance of the ITU standard for these networks, called OTU-4, by a large distance. This itself means that great potential can be expected from polar codes in this area.

1.2 Results on Wireless Networks

In the past, many analytic results on the connectivity, coverage, and capacity of wireless ad-hoc and sensor networks have been obtained. In almost all of the results,

it is assumed that the number of nodes n in the network tends to infinity (large-scale networks). In other words, these results are asymptotic. Asymptotic results are very important as discussed earlier; however, in many practical wireless networks the number of nodes may be limited to a few hundred (small-scale/finite networks). Thus, analyzing finite networks is extremely important from a practical point of view.

To clarify, let us consider, for example, capacity analysis of wireless networks which has been studied extensively (e.g., in [6–12]). Today we have good understanding of scaling laws in capacity of wireless networks. However, suppose we need to design a wireless sensor network consisting of a hundred sensor nodes. Some fundamental questions are as follows. What is the transport capacity? What is the information theoretic capacity? What is the maximum number of concurrent transmissions (also called "MAC-Layer Capacity") possible? How do network parameters such as communication radius of nodes, number of nodes, and so on, affect these capacities? Unfortunately, the available asymptotic results fail to give answers to these questions. Similar questions are remained unanswered for other properties of the network such as connectivity, coverage, etc.

The main goal of our effort is to initiate the small-scale analysis of wireless sensor and ad hoc networks. Such analyses can be very useful in analyzing and evaluating communication and security protocols for practical sensor and ad hoc networks and is completely overlooked in the literature. There is an important need to develop mathematically rigorous results to guarantee the performance measures and also to understand the properties of finite networks. To achieve this goal we will try several approaches. A combination of these approaches will help in providing guidelines for rigorous analysis of finite networks. Here, we study two possible approaches.

1.2.1 Threshold Intervals Approach

Threshold phenomenon is the most important concept in asymptotic analysis of random graphs. It basically refers to situations in which the probability for an event to occur changes rapidly as some underlying parameter varies. The threshold phenomenon has been studied extensively in different areas such as probability, statistics, percolation theory and statistical physics.

A general framework is provided in the literature by studying pivotality and influence of variables for Boolean functions (see for example [13–22]). In this general framework, the transition from zero to one of the probability of certain events as a function of an underlying parameter ρ is studied. An important measure called the *threshold interval* is usually studied. This is the interval $[\rho_1, \rho_2]$ in which the probability of an event A changes from ϵ to $1-\epsilon$. The length of threshold interval, shown as $\tau(A, \epsilon)$, indicates the sharpness of the threshold. The value $\tau(A, \epsilon)$ has been studied asymptotically so far. That is, it is usually shown that we have a sharp threshold and thus $\tau(A, \epsilon)$ is small. Our goal is to study $\tau(A, \epsilon)$ for finite Boolean functions and extend the current available literature for probabilities regarding finite networks. This will immediately give us rigorous regions for the probabilities we are studying in finite networks. This approach also helps us to understand the transition from finite domain to the asymptotic domain. In other words, we can see how exactly the threshold interval shrinks as the number of nodes increases. Moreover, for a given network property, we will be able to determine how large the network should be so that the asymptotic formulas provide acceptable estimations.

1.2.2 MAC-Layer Capacity

Asymptotic MAC-layer capacity of ad hoc wireless networks has been studied in [6]. The MAC-layer capacity is defined as the maximum possible number of concurrent transmissions at the media access layer. In [6], it is shown that for a wide class of

MAC protocols including IEEE 802.11, the MAC-layer capacity can be modeled as a maximum D2-matching (D2EMIS) problem in the underlying wireless network. The main result of [6] is that for a network with n nodes and communication radius r, the MAC-layer capacity is optimized at $r = \Theta(\frac{1}{\sqrt{n}})$ and is given by $\Theta(n)$. Although this is an important and valuable result, it is not as precise when we consider finite networks. For example, suppose we have a network consisting of 100 sensors and we want to choose the communication radius such that the MAC-layer capacity is optimized. The asymptotic result does not tell us what the optimum MAC-layer capacity and its corresponding communication radius are. In this dissertation, we analyze the average MAC-layer capacity for finite line networks modeled by random geometric graphs. Simple closed-form expressions will be provided for the lower and upper bounds on the capacity. While giving a good estimate of the exact value, such expressions can be easily used in finding the optimum communication radius in practical network models.

1.3 Publications

Below is a selected list of our publications based on this dissertation.

Journal Papers:

1. A. Eslami and H. Pishro-Nik, "On Finite-Length Performance of Polar Codes: Stopping Sets, Error Floor, and Concatenated Design," *IEEE Transactions on Communications*, Available on ieeexplore.com, Aug. 2012.

2. A. Eslami, M. Nekoui, H. Pishro-Nik, and F. Fekri, "Results on Finite Wireless Sensor Networks: Connectivity and Coverage" *ACM Transactions on Sensor Networks*, Vol. 10, No. 1, Feb. 2014 (to appear).

3. A. Eslami, M. Nekoui, and H. Pishro-Nik, "Results on Finite Wireless Networks on A Line," *IEEE Transactions on Communications*, Vol. 58, No. 8, Aug. 2010.

Conference Papers:

1. A. Eslami and H. Pishri-Nik, "A Practical Approach to Polar Codes," in *Proc. IEEE International Symposium on Information Theory (ISIT)*, Aug 2011.

2. A. Eslami and H. Pishro-Nik, "On Bit Error Rate Performance of Polar Codes in Finite Regime," in *proc. 48th Annual Allerton Conference on Communication, Control, and Computing*, Sept. 2010.

3. A. Eslami, M. Nekoui, and H. Pishro-Nik, "Results on Finite Wireless Networks on A Line," in *Proc. IEEE Information Theory Workshop (ITW)*, Jan. 2010.

CHAPTER 2

CHANNEL CODING: A REVIEW

2.1 Introduction

Reliable data transmission is a central topic of information theory with applications everywhere; consider mobile communication, MP3 players, the Internet, CDs, or any other modern day digital technology. To make communication reliable in the presence of noise, the common procedure is to add redundancy to the data before transmission. The intended receiver only has access to a noisy version of the data. However, if the redundancy is added in a clever way, then it is possible to reconstruct the original data at the receiver. Adding redundancy is called *coding*. Coding is a central part of any communication system; e.g., consider wired phones, mobile phones, or the Internet. Coding is also used for storage on CDs and DVDs to prevent data loss due to scratches or errors during the reading process.

2.2 Channel Model

Shannon, in his seminal work [23], formalized the above problem of communication and determined its fundamental limits. He provided a mathematical framework to study the problems systematically which led to the advances in the past 50 years. The generality of his approach allows us to study even modern day scenarios, like mobile communication or ad hoc networks. The basic model addressed by Shannon consists of a source, which generates the information, a sink which receives the information, and a channel, which models the physical transfer of information.

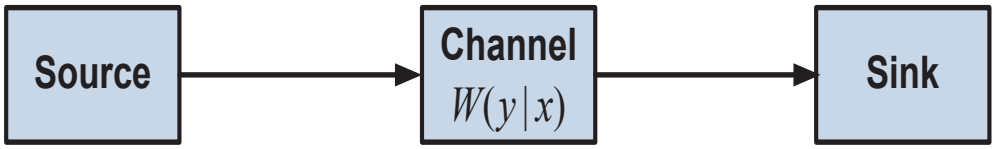


Figure 2.1. The basic communication scenario with information source and sink, and a communication channel.

The channel is modeled by a conditional probability distribution (see Fig. 2.1). Let $\mathcal{X}$ and $\mathcal{Y}$ denote the input and output alphabet of the channel. The channel $W : \mathcal{X} \to \mathcal{Y}$ is a conditional probability distribution $W(y|x)$. When x is transmitted through the channel, the output at the receiver is the realization of a random variable $Y \in \mathcal{Y}$ distributed as $W(y|x)$. Shannon showed that in spite of this randomness, by intelligently adding redundancy, the data can be reproduced exactly at the receiver with high probability. He computed the capacity of a channel, $C(W)$, which quantifies the maximum rate at which reliable transmission of information is possible. In other words, for any $R < C(W)$, there exists a scheme which transmits R bits per channel use with vanishing error probability in the block length. For channel coding to approach its fundamental limits, the blocklength has to be large. This in turn has implications on the complexity as a large block length often means more processing power upon decoding, as well as more memory space to store the code blocks and any required look-up tables. Therefore, for practical applications, we require schemes that operate with low space and computational complexity.

2.3 Existing Low-Complexity Schemes

Since Shannon's seminal work [23] the main goal has been to construct low-complexity coding schemes that achieve the fundamental limits. The complexity issues arise in two different contexts.

The first issue is the amount of memory required to store the code. This refers to the memory required for storing the mapping from the input of the encoder to its

output. A code of rate R and blocklength N consists of 2^{NR} codewords of length N. A naive representation of such a code requires $O(N2^{NR})$ bits of memory, which is not practical. A significant progress in this respect was the result of Elias [24] and [25] (Section 6.2) which shows that linear codes are sufficient to achieve the capacity of an important class of channels, known as symmetric channels. Linear codes are subspaces of vector spaces. Hence, a linear code can be specified in terms of a basis of this subspace. This in turn can be done by describing RN vectors of length N. Therefore, the resulting memory requirement is $O(N^2)$ bits. This is an exponential improvement over the general case.

The second issue is the computational complexity of the encoding and decoding operations. In the following we give a brief history of some of the important developments. For a detailed history of channel coding we refer the interested reader to the excellent article by Costello and Forney [26].

2.3.1 Channel Coding

The initial research in coding was based on an algebraic approach. More precisely, the focus was on developing linear binary codes with large minimum distance (the smallest distance between any two distinct codewords) and good algebraic properties. The first algebraic codes were developed by Hamming and are named after him. *Hamming codes* are single error correcting codes and they are optimal in the sense of sphere packings. Other important algebraic codes are *Golay codes, BCH codes, Reed-Muller codes, and Reed-Solomon codes* [5]. For all these codes efficient algebraic decoding algorithms are known. The complexity for these algorithms normally ranges between $O(Nlog^2N)$ to $O(N^2)$. These codes are prominently used today in CDs, DVDs and modems. BCH and Reed-Solomon codes are instances of *Cyclic Codes* that are widely used in communication systems for error correction. Cyclic codes are very attractive

mainly because their encoding, syndrome computation, and decoding algorithms can be implemented easily by employing shift registers with feedback connections.

As mentioned in the previous section, to achieve optimal performance one has to consider large blocklengths. The improvement in computational resources made it feasible to consider larger and larger codes in practice. But the previously discussed algebraic codes either have vanishing rate or vanishing relative distance (ratio of minimum distance and blocklength) for increasing blocklengths.

Product codes, introduced by Elias [27], were the first constructions which asymptotically (in the blocklength) achieved both non-vanishing relative distance and rate. The idea is to construct large codes by combining two or more codes of smaller length. Consider two codes C_1 and C_2 of length N_1 and N_2. Each codeword of the product code can be viewed as an $N_1 \times N_2$ matrix such that each column is a codeword of C_1 and each row is a codeword of C_2. Code concatenation, introduced by Forney [28], is another construction based on combining codes. The idea is to first encode the data using C_1 and then encode the resulting output with C_2. Forney showed that for any rate below the capacity, by an appropriate choice of the two component codes, the error probability can be made to decay almost exponentially with a decoding algorithm that has polynomial complexity.

The next big step in improving the decoding performance came from considering probabilistic decoding. In typical scenarios the capacity is significantly reduced by making hard-decisions at the decoder. E.g., for Gaussian channels, close to half the power is lost due to this first step. The idea of probabilistic decoding is to make use of the channel outputs directly in the decoding algorithm and to avoid this loss.

The first class of codes well suited for probabilistic decoding were *convolutional codes*, introduced by Elias in [24]. The code structure enabled the development of efficient decoding algorithms. In particular the Viterbi algorithm [29], which minimizes the block error probability, and the BCJR algorithm [30], which minimizes the

bit error probability, both operate with complexity which is linear in the blocklength. For any rate strictly less than the capacity of the channel one can show that there exist convolutional codes whose probability of error vanishes exponentially in the "constraint length". However, the complexity of the decoding algorithm also scales exponentially with the constraint length. Therefore, for practical purposes Fanos sequential decoding algorithm [31] was considered. For rates less than the "computational cutoff rate", the complexity of this algorithm is linear in the blocklength and independent of the constraint length. The cutoff rate is a rate that can be computed easily and that is strictly smaller than the capacity. For rates above the cutoff rate, the decoding complexity is unbounded. This led to the belief that, using practical algorithms, rates above the cutoff rate could not be achieved.

Another class of codes which were introduced during the 60s were *low-density parity-check (LDPC) codes* [32]. As the name suggests, the parity-check matrices of these codes have very few non-zero entries. In fact, these matrices have a constant number of non-zero entries per row and column. Gallager showed that these codes have a non-zero relative distance. He also proposed a low-complexity iterative decoding algorithm. Unfortunately, due to the lack of computational resources at that time, the power of these codes and the decoding algorithms was not realized.

The invention of *turbo codes* by Berrou, Glavieux and Thitimajshima [33] was a breakthrough in the practice of coding. Turbo codes achieved rates close to the capacity, and far above the cutoff rate, using a linear complexity decoding algorithm. The code is constructed by concatenating two convolutional codes but with a random bit interleaver in between. The decoding algorithm operates in iterations. In each iteration the BCJR algorithm is performed on each of the component codes and the reliabilities are exchanged. Since the complexity of the BCJR algorithm is linear in the blocklength, the resulting decoding algorithm is also of linear complexity. The original turbo code of [33] matched the error probability of the best existing schemes

which were operating with twice the power. The interleaver and the iterative property of the decoding algorithm are the two crucial components which are recurrent in the capacity achieving schemes constructed later.

Wiberg, Loeliger and Kotter [34] unified turbo codes and LDPC codes under the framework of codes on graphs. The success of turbo codes and the subsequent rediscovery of LDPC codes revived the interest in LDPC codes and message passing algorithms. Some of the first and important contributions to the analysis of message-passing algorithms was done in a series of papers by Luby, Mitzenmacher, Shokrollahi, and Spielman [35]. In [35], the authors analyzed a suboptimal decoder known as peeling decoder for the binary erasure channel (BEC). They constructed codes for the BEC which achieve capacity using the peeling decoder.

In [36], Richardson and Urbanke developed density evolution which generalizes the analysis of the BEC to any symmetric channel and a class of algorithms known as message-passing algorithms. This class includes the important belief propagation algorithm. Combining density evolution for belief propagation with optimization techniques, codes that approach capacity of Gaussian channel to within 0.0045dB [37] were constructed. However, unlike the BEC, no capacity achieving codes are known for general channels.

Until now, many turbo and LDPC codes have been proposed which empirically achieve rates close to capacity for various channels. However, none of these codes are proven to achieve capacity for channels other than the BEC. In this thesis, we discuss polar codes. Polar codes, recently introduced by Arıkan [1], are a family of codes that provably achieve the capacity of symmetric channels with "low encoding and decoding complexity". This settles the long standing open problem of achieving capacity with low complexity.

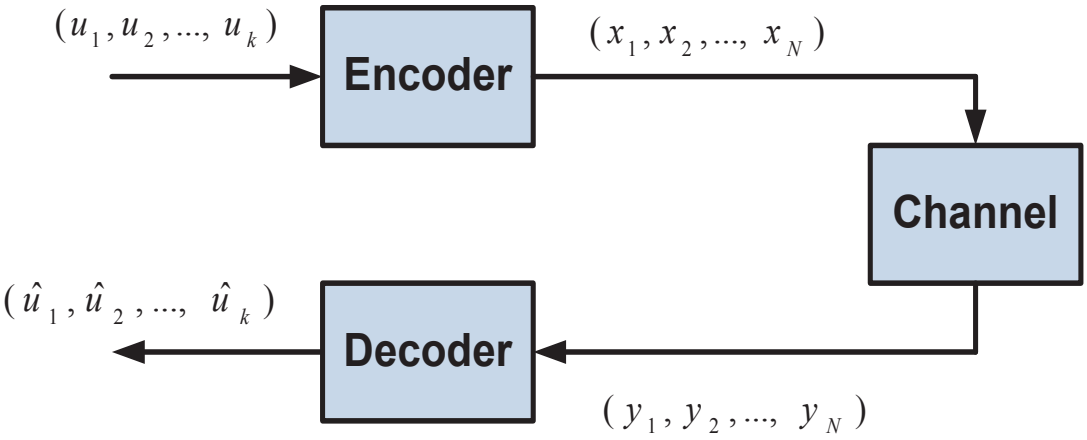


Figure 2.2. Block codes for noisy channels. k information bits will be mapped to a block of N code-bits by the encoder.

2.3.2 Linear Block Codes

Block codes are one of the most important classes of error-correcting codes. In block coding, before the transmission, information bits are divided to several blocks. Each block has the same length k. For example in Fig. 2.2, the block is the vector $(u_1, u_2, ..., u_k)$ where u_is are the *information bits*. Then the information block is mapped to a longer block $(x_1, x_2, ..., x_N)$, which is called the *codeword*. In the following, let $\bar{y} = (y_1, ..., y_N)$ denote the channel output bits. We use upper case letters U, X, Y to denote random variables and lower case letters u, x, y to denote their realizations. We also use u_i^j to denote $(u_i, ..., u_j)$. The length of the codeword, N, is called the *code-length* (or *blocklength*). Note that $N > k$, thus there is some redundancy in the codeword. In fact, this redundancy is used in the decoding process to detect and correct errors.

Most practical block codes are linear, that is the mapping from the information block to the codeword is a linear mapping. This means that there exists a binary matrix G, the generator matrix, that defines the encoding process. Specifically if $\bar{u} = (u_1, u_2, ..., u_k)$ and $\bar{x} = (x_1, x_2, ..., x_N)$ are the information block and the codeword, respectively, then $\bar{x} = \bar{u}G$.

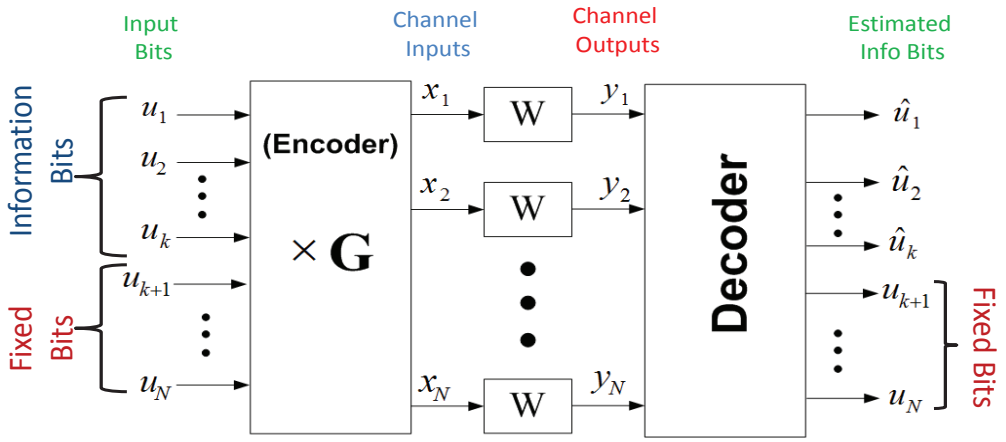


Figure 2.3. A linear block code of length N over a B-DMC. The figure depicts the full-matrix representation with information and fixed (frozen) bits.

Equivalently, a linear block code can be defined by a parity-check matrix, H. The parity-check matrix H is a $(N-k) \times N$ matrix satisfying $GH^T = 0$. A binary vector $\bar{x} = (x_1, x_2, ..., x_N)$ is a valid codeword if and only if $\bar{x}H^T = 0$.

2.3.2.1 Full-Matrix Representation

Fig. 2.3 shows an alternative yet simple way of representing a linear block code which proves helpful in understanding polar codes. Here, we have a block of length N of *input bits* from which k bits form our set of information bits, and the rest of $N-k$ bits are fixed to some known values. Note that the values of the fixed bits are also known to the decoder, so they can be chosen arbitrarily. Now, these N input bits are multiplied by a $N \times N$ generator matrix $G_{N\times N}$. Note that to make this representation equivalent to the scheme above, the $G_{N\times N}$ is in fact obtained by adding $N-k$ rows to the $G_{k\times N}$ above. These extra $N-k$ bits correspond to the $N-k$ fixed bits, and hence, can be chosen arbitrarily. Therefore, at the encoder side, the two representation are equivalent. At the receiver side, decoder sets all the fixed bits to their values before it starts decoding a block. As a result, there remain the same k bits as before to be estimated by the decoder.

2.4 Polar Codes

In this section we discuss the construction of polar codes for channel coding. It is based entirely on the work of Arıkan [1]. This section lays the foundation and sets the notation for the rest of our analysis of polar codes. For the sake of brevity we skip all the proofs.

2.4.1 Channel Polarization

The construction of polar codes is based on the following observation: Let

$$F = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}$$

be the kernel used for construction of polar codes. First, apply the transform $F^{\otimes n}$ (where $\otimes n$ denotes the nth Kronecker power) to a block of $N = 2^n$ bits U_1^N. For instance, for $N = 4$ and $N = 8$ we will have

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

and

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

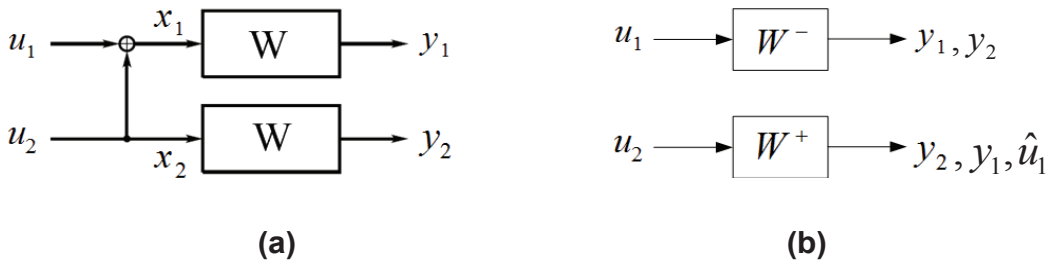


Figure 2.4. Figure on the left shows a rate 1 polar code of length 2. Using a SC decoder, the equivalent channels seen by input bits can be obtained as depicted on the right.

Then, transmit the output $X_1^N = U_1^N F^{\otimes n}$ through independent copies of a symmetric *binary discrete memoryless channel (B-DMC)* W. Note that we are using the full-matrix representation of block codes here. Now, apply the chain rule to the mutual information between the input U_1^N and the output Y_1^N. This gives

$$I(U_1^N; Y_1^N) = \sum_{i=1}^{N} I(U_i; Y_1^N | U_1^{i-1}) = \sum_{i=1}^{N} I(U_i; Y_1^N, U_1^{i-1}), \tag{2.1}$$

where the last equality follows from the fact that $U_i s$ are independent. The central observation of polar codes is that as n grows large, except for a negligible fraction, the terms in the summation either approach 0 (bad) or 1 (good). Moreover, the fraction of those terms tending to 1 approaches the symmetric channel capacity $C(W)$. This phenomenon is referred to as *channel polarization*.

Since the idea of channel polarization is the core concept as well as the beauty of polar codes, let us elaborate on this idea a little more. In its simplest case, consider a rate one block code of length 2 with $F = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}$ as its generator matrix. Fig. 2.4(a) shows such a setting. We have

$$x_1 = u_1 + u_2,$$

$$x_2 = u_2.$$

x_1 and x_2 will be sent over the channel W and we will receive y_1 and y_2 at the receiver. Decoder needs to estimate u_1 and u_2 based on y_1 and y_2. Now let us assume a simple successive cancellation (SC) decoder as follows:

$$\hat{u}_1 = \begin{cases} 0 & \text{if } \frac{\Pr(u_1=0|\bar{y})}{\Pr(u_1=1|\bar{y})} > 1; \\ 1 & \text{otherwise.} \end{cases}$$

$$\hat{u}_2 = \begin{cases} 0 & \text{if } \frac{\Pr(u_2=0|\bar{y},\hat{u}_1)}{\Pr(u_2=1|\bar{y},\hat{u}_1)} > 1; \\ 1 & \text{otherwise.} \end{cases}$$

Note that based on this decoder, we can realize the channels W^- and W^+, "seen" by u_1 and u_2, respectively (See Fig. 2.4(b)). To be precise, given $W(y|x)$, we can find $W^-(y_1y_2|u_1)$ and $W^+(y_1y_2u_1|u_2)$ as follows:

$$W^-(y_1y_2|u_1) = \sum_{u_2} \frac{1}{2} W(y_1|u_1+u_2)W(y_2|u_2),$$
$$W^+(y_1y_2u_1|u_2) = \frac{1}{2} W(y_1|u_1+u_2)W(y_2|u_2).$$

Assuming independent, uniform U_1 and U_2, W^- and W^+ have the following symmetric capacities [1]:

$$C(W^-) = I(U_1; Y_1, Y_2),$$
$$C(W^+) = I(U_2; U_1, Y_1, Y_2).$$

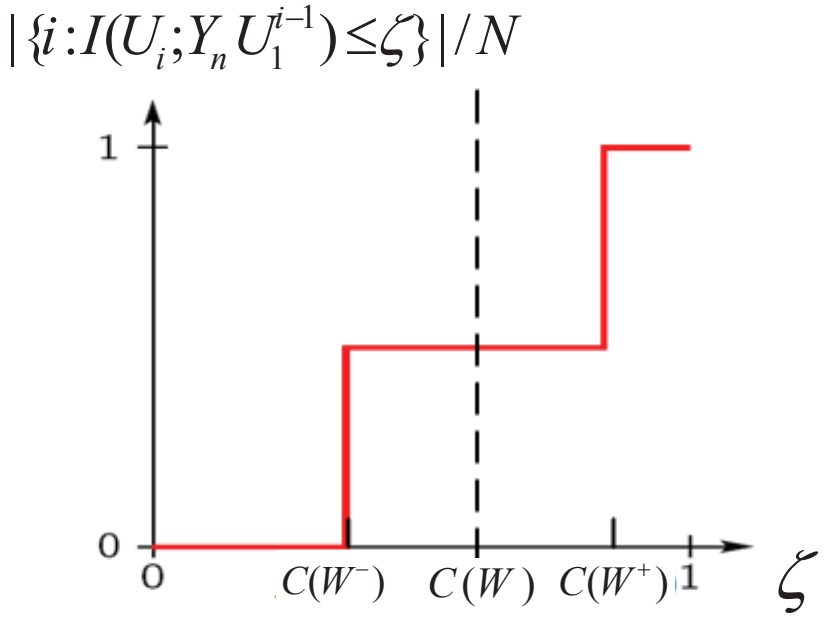


Figure 2.5. Relation between the capacities of W^-, W^+, and W. $C(W^-)$ and $C(W^+)$ diverge from $C(W)$ although the summation of the two will still be $2C(W)$.

Now the interesting point here is the following:

$$C(W^-) + C(W^+) = I(U_1 U_2; Y_1 Y_2) = 2C(W),$$
$$C(W^-) \leq C(W) \leq C(W^+).$$

Fig. 2.5 shows the distribution of capacities for W^- and W^+. To be precise, for any $\zeta \in \mathbb{R}, 0 \leq \zeta \leq 1$, it shows the fraction of the terms in eq. (2.1) that are less than ζ. You can see how $C(W^-)$ and $C(W^+)$ compare to $C(W)$.

Now consider a block code of rate 1 and length 4 with $F^{\otimes 2}$ as its generator matrix. Fig. 2.6 shows how such a code can be constructed by connecting two code of length 2. This is because of the recursive structure of polar codes' generator matrix obtained by the Kronecker power of F. By applying the same setting as above, i.e. using a SC decoder similar to the one above, we can realize the four channels seen by u_1, u_2, u_3,

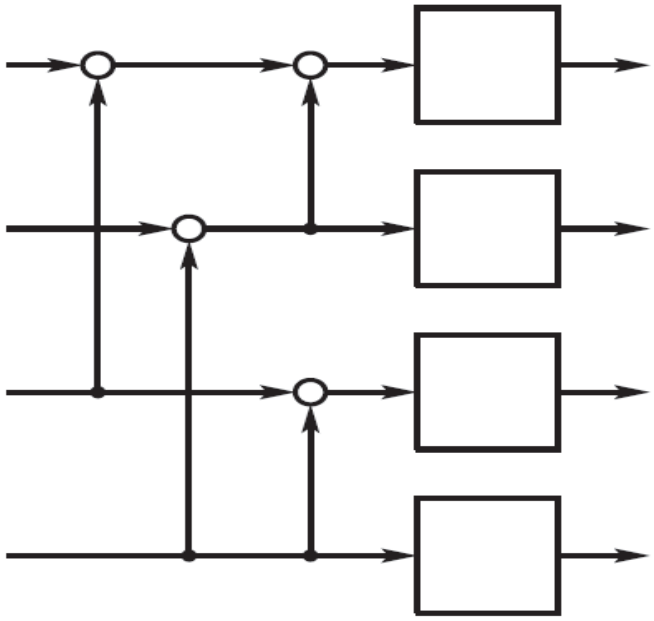


Figure 2.6. Rate 1 polar code of length 4 can be simply synthesized by connecting two length-2 polar codes. This holds because of the recursive nature of the generator matrix.

and u_4. If we find the capacities of these four channels, their distribution will be as is shown in Fig. 2.7.

The same procedure can be applied for the codes of length 2^4, 2^8, 2^{20} and longer. Fig. 2.8 depicts how the capacities of the channels seen by the input bits u_i form around the capacity of the underlying B-DMC, W. As it is clear from the figure, the channels seen by input bits start diverging from the original B-DMC, W, as we increase the code-length. The main result of [1] by Arikan is that, as n grows large, these channels start polarizing to either a noiseless channel or a pure-noise channel, where the fraction of channels becoming noiseless is close to the symmetric capacity $C(W)$. Below, we explain how polar codes use this fact to achieve the channel capacity.

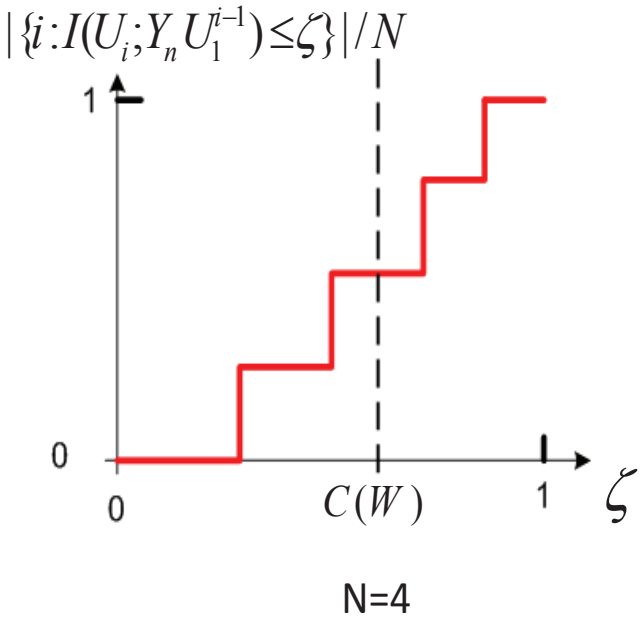


Figure 2.7. Relation between the capacities of channels seen by the input bits in a polar code of length 4. Capacities of the channels seen by the input bits form around $C(W)$ while the sum of these capacities adds up to $4C(W)$.

2.4.2 Construction of Polar Codes

As it was explained above, we can associate the i-th row of $F^{\otimes n}$ with $I(U_i; Y_1^N, U_1^{i-1})$. More precisely, this term can be interpreted as the channel "seen" by the bit U_i assuming that we have already decoded all the previous bits. With this interpretation we choose the generator matrix of the polar code of rate R in the following way; choose those NR rows of $F^{\otimes n}$ with the largest $I(U_i; Y_1^N, U_1^{i-1})$. Arıkan [1] proved that such codes achieve rates close to $C(W)$, with vanishing block error probability, using a low-complexity Successive Cancellation decoder (explained below). Polar codes use the noiseless channels for transmitting information while fixing the symbols transmitted through the noisy ones to a value known both to the sender and the receiver. Accordingly, part of the block that carries information includes "information bits" while the rest of the block includes "frozen bits". Since we only deal with symmetric channels here, we assume without loss of generality that the fixed positions are set

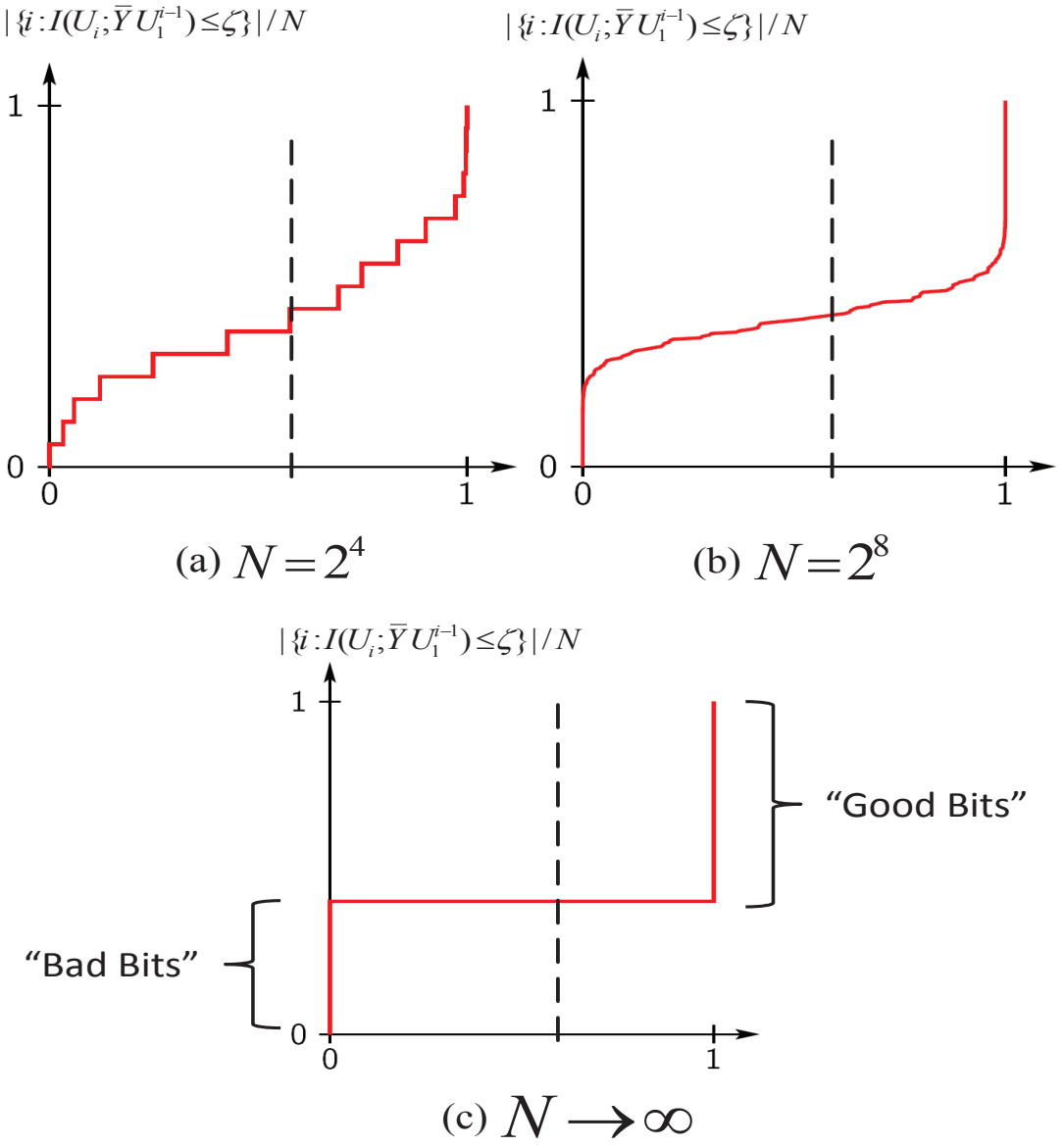


Figure 2.8. The capacities of channels seen by the input bits polarizing around $C(W)$ as the code-length grows large. Arıkan [1] proved that a ratio of $C(W)$ of all the channels will become noiseless while the rest will turn to pure noise channels. To achieve the capacity, we only need to carry information over the noiseless channels.

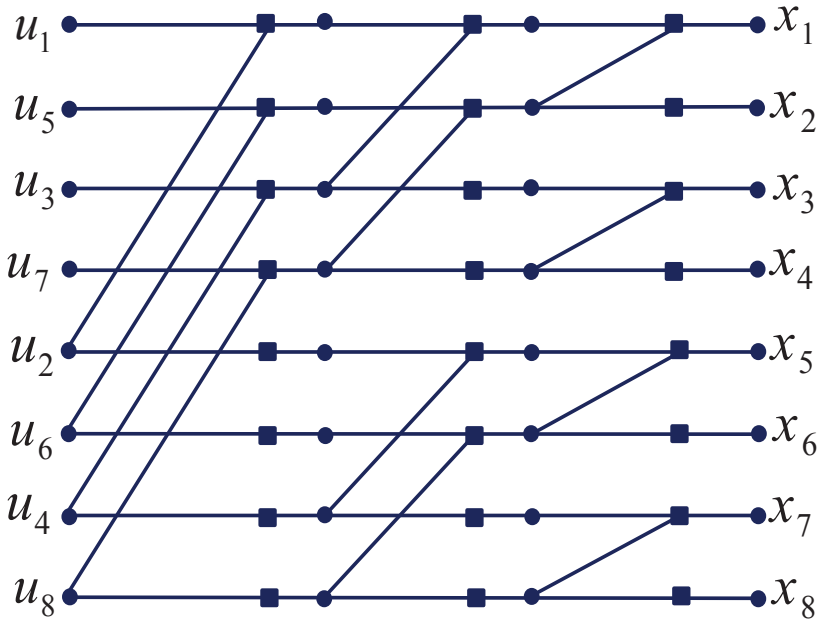


Figure 2.9. Factor graph representation of a polar code of length 8 used in SC and BP decoding. The graph is derived from the generator matrix. Input bits are on the left side and code-bits are placed on the right.

to 0. Since the fraction of channels becoming noiseless tends to $C(W)$, this scheme achieves the capacity of the channel.

2.4.3 Decoding Schemes for Polar Codes

A Successive Cancellation (SC) decoder is employed in [1] to prove the capacity-achieving property of polar codes. In such a decoder, the bits are decoded as follows. Let the estimates of the bits be denoted by $\hat{u}_1, ..., \hat{u}_N$. If a bit u_i is frozen then $\hat{u}_i = 0$. Otherwise the decoding rule is the following:

$$\hat{u}_i = \begin{cases} 0 & \text{if } \frac{\Pr(\bar{y}|\hat{u}_1^{i-1}, U_i=0)}{\Pr(\bar{y}|\hat{u}_1^{i-1}, U_i=1)} > 1; \\ 1 & \text{otherwise.} \end{cases}$$

Using the factor graph representation between $\bar{u}$ and $\bar{x}$ shown in Fig. 2.9, Arıkan showed that this decoder can be implemented with $O(N \log N)$ complexity.

However, [38] and [39] later proposed using belief propagation decoding to obtain better BER performance while keeping the decoding complexity at $O(N \log N)$. Belief propagation can be run on the factor graph representation of the code [38]. Such a representation is easily obtained by adding check nodes to the encoding graph of polar codes, as it is shown in Fig. 2.9 for a code of length 8. In the figure, variable nodes and check nodes are represented by small circles and squares, respectively. We refer to this graph as the code's *factor graph*. The most important variable nodes are the channel outputs and the information bits.The goal of BP is to recover information bits given the channel outputs and froze bits.

BP runs on the factor graph in a column-by-column fashion. That is, BP runs on each column of the adjacent variable and check nodes. The parameters are then passed to the next column. Each column, as it can be seen in Fig. 2.9, is formed of some Z-shaped subgraphs. In our proofs, we sometimes simply call a Z-shaped part a "Z". The schedule with which BP runs is very important for channels other than BEC. Here, we use the same scheduling used in [39], i.e. we update the log-likelihood ratios (LLRs) for Z parts from bottom to top for each column, starting from the rightmost one. After arriving at the leftmost column, we reverse the course and update the Zs from top to bottom for each column, moving toward the rightmost one. This makes one round of iteration, and will repeat at each round. While we tried other schedules as well, this one led to a better overall performance.

CHAPTER 3

FINITE-LENGTH ANALYSIS OF POLAR CODES

3.1 Introduction

Since their introduction, polar codes have attracted a lot of attention among researchers due to their capability to solve many problems (sometimes open problems) that could not be handled using other schemes. However, theoretical approaches have been mostly taken toward polar codes in the literature. Our goal is to study polar codes from a practical point of view to find out about properties that can be useful in real-world applications. Hence, we are mainly concerned with the performance of polar codes in the finite regime (i.e. with finite lengths) as opposed to the asymptotic case. Some of the previous work related to finite-length polar codes include [40–48]. Particularly, [41] proposes a successive cancellation list decoder that bridges the gap between successive cancellation and maximum-likelihood decoding of polar codes. Inspired by [41], [49–51] propose using CRC along with list decoding to improve the performance of polar codes. [42] presents a method to improve the finite-length performance of successive cancellation decoding by means of simple and short inner block codes. A linear program (LP) decoding for polar codes is considered in [44]. In [46], a method for efficient construction of polar codes is presented and analyzed. In addition, scaling laws are provided in [52–56] for the behavior of polar codes that, in some cases, have finite-length implications.

Since an analysis in the finite regime can be very difficult in general, we start with studying the performance of polar codes over the binary erasure channel (BEC). A binary erasure channel (shown in Fig. 3.1) is a common communications channel

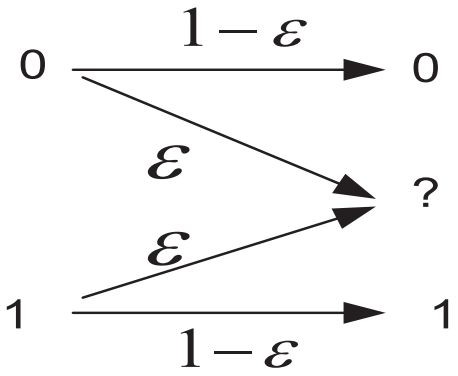


Figure 3.1. A binary erasure channel (BEC) with parameter (erasure probability) ϵ.

model in which a transmitter sends a bit (a zero or a one), and the receiver either receives the bit or it receives a message that the bit was not received ("erased"). The probability of erasure is ϵ. We might denote this channel by BEC(ϵ). While being fairly manageable, an analysis over the BEC leads to a better understanding of the behavior of polar codes. We provide an analysis of the stopping sets in the *factor graph realization* of polar codes. Factor graph of polar codes, as it is explained in next section, is a bipartite graph formed by variable and check nodes connected together through the edges. Such a realization for polar codes was first employed by [38] and [39] to run Belief Propagation (BP) as the decoding algorithm. Stopping sets are important as they contribute to the decoding failure and error floor, when BP is used for decoding [57]. A stopping set is a non-empty set of variable nodes such that every neighboring check node of the set is connected to at least two variable nodes in the set. For the BEC, it is proved [57] that the set of erasures which remain when the decoder stops is equal to the unique maximal stopping set within the erased bits. Therefore, in the case of BEC, stopping sets are the sole reason of decoding failure. A stopping set with minimum number of variable nodes is called a *minimum stopping set*. Minimum stopping sets play an important role in the decoding failure.

In addition to the above, since BP is rather well-studied in the context of LDPC codes, there are many approaches to modify BP in order to obtain better BER performance; examining such schemes in the context of polar codes is another interesting issue. Therefore, we consider a modified version of BP while employing a guessing algorithm. The algorithm was studied in [58] and [59] and was shown to be considerably helpful in the case of LDPC codes with good error floor performance.

Our main contributions in this chapter are as follows:

- We find the structure of the minimum stopping set and its size, called *stopping distance*.
- We will show that the stopping distance grows polynomially for polar codes. This is a clear advantage over capacity-approaching LDPC codes.
- We find the girth of the factor graph of polar codes, showing that polar codes hold a relatively large girth.
- Simulation results will be provided to investigate the effect of such a large girth and stopping distance on the error floor behavior of polar codes over the binary erasure and AWGN (Additive White Gaussian Noise) channels.
- We show that applying the aforementioned guessing algorithm to polar codes leads to significant improvements in the BER performance.

The rest of the chapter is organized as follows. We first explain the notations and provide a short background on belief propagation. Section 3.3 gives an analysis on the minimum stopping set of polar codes. We provide a girth analysis of polar codes in Section 3.4 where we also present simulation results for error floor performance. We will then study the improvement made in BER by employing a modified version of BP using guessing techniques. All proofs for the facts, lemmas, and theorems have

been moved to the Section 3.7 at the end of the chapter. The results of this chapter have been published in [47] and [60].

3.2 Preliminaries

In this section, we explain the notations and some preliminary concepts we will use in our analysis. As it is shown in Fig. 3.2, the factor graph is formed of columns of variable nodes and check nodes. There are, respectively, $n+1$ and n columns of variable and check nodes in the graph. We denote the variable nodes in jth column by $v(1,j), v(2,j), ..., v(N,j)$ for $j = 1, ..., n+1$. This is also shown in Fig. 3.2. Similarly, check nodes are labeled as $c(1,j), c(2,j), ..., c(N,j)$ for $j = 1, ..., n$. The rightmost column in the graph includes code-bits, while the leftmost column includes frozen and information bits. As it will become clear, our analysis does not depend on any specific choice of the frozen and information bits. Therefore, we treat all the nodes in the left-most column as variable nodes. Among $v(i,1), i = 1, ..., N$, some are associated to the information bits. We denote the index set of information bits by $\mathcal{A}$ where $\mathcal{A} \subseteq \{1, 2, ..., N\}$. Also, the row in $F^{\otimes n}$ associated with an information bit $i \in \mathcal{A}$ will be denoted by $\mathbf{r}_i = [r_{i,1}\ r_{i,2}\ ...\ r_{i,N}]$. Note that this is the ith row of $F^{\otimes n}$. We denote by $wt(\mathbf{r}_i)$ the Hamming weight of $\mathbf{r}_i$.

We denote the factor graph of a code of length $N = 2^n$ by T_n. A key observation is the symmetric structure of this graph due to the recursive way of finding the generator matrix: T_{n+1} includes two factor graphs T_n as its upper and lower halves, connected together via $v(1,1), v(2,1), ..., v(N,1)$ and $c(1,1), c(2,1), ..., c(N,1)$. We denote these two subgraphs by T^U_{n+1} and T^L_{n+1}, as it is shown in Fig. 3.2. This observation will be later used in our analysis.

In this chapter, we are particularly interested in the analysis of *stopping sets* in the factor graph of polar codes. A stopping set is a non-empty set of variable nodes such that every neighboring check node of the set is connected to at least two variable

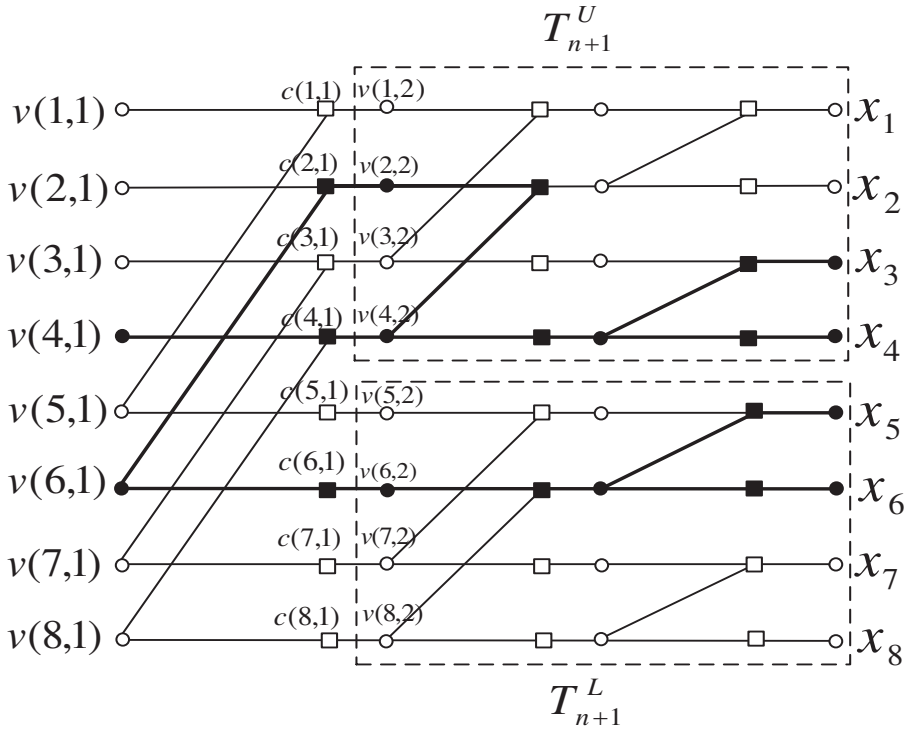


Figure 3.2. Normal realization of the encoding graph for $N = 8$. An example of a GSS is shown with black variable and check nodes. Notice the columns of variable and check nodes. The figure also depicts the two induced tanner graphs T_n in upper and lower halves.

nodes in the set. Fig. 3.2 shows an example of the stopping set in the polar codes' graph, where we have also included the corresponding set of check nodes. A stopping set with minimum number of variable nodes is called a *minimum stopping set*.

3.2.1 Stopping Trees

An important category of stopping sets in the factor graph of polar codes are *stopping trees*. A stopping tree is a stopping set that contains one and only one information bit. It can be easily seen that this sub-graph is indeed a tree, therefore justifying its name. We say that the stopping tree is rooted at its (single) information bit (on the left side of the graph), with leaves at code-bits (on the right side of the graph). An example of such a stopping set is shown in Fig. 3.3 with black variable

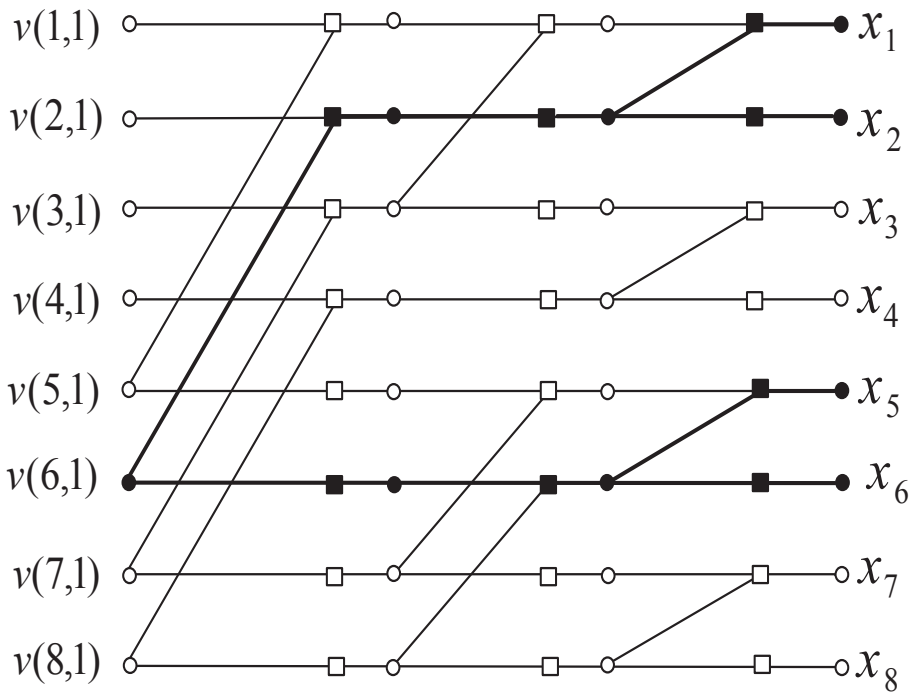


Figure 3.3. The stopping tree for $v(6,1)$ is shown with black variable and check nodes. The tree is rooted at $v(6,1)$ and has leaves at code-bits x_1, x_2, x_5, x_6.

nodes. We also included the corresponding set of check nodes in order to visualize the structure of the tree. A stopping tree like the one shown in Fig. 3.3 can be immediately realized for any information bit. As we will later see (in Lemma 2 below), this would in fact be the unique stopping tree for each information bit. We denote the stopping tree rooted at $v(i,1)$ by $ST(i)$. Among all the stopping trees, the one with minimum number of variable nodes is called a *minimum stopping tree*. We refer to the set of leaf nodes of a stopping tree as the *leaf set* of the tree. The size of the leaf set for $ST(i)$ is denoted by $f(i)$. We refer to a stopping tree with minimum leaf set as a *Minimum-Leaf Stopping Tree (MLST)*. Note that a minimum stopping tree does not necessarily have the minimum $f(i)$ among all the stopping trees.

3.2.2 Graph Stopping Sets vs. Variable-Node Stopping Sets

By looking at the factor graph of polar codes, one can observe that the middle variable nodes, i.e. $v(i,j)$ for $j = 2, ..., n$ and $i = 1, ..., N$, are always treated as erasures by the BP decoder. This is also true about information bits. Frozen bits, on the other hand, are known to the decoder. As a result, the only real "variable" nodes are the code-bits, i.e. $v(1, n+1), ..., v(N, n+1)$. These are in effect the variable nodes that if erased may cause a decoding failure. Here, we refer to a stopping set on the graph as a *Graph Stopping Set (GSS)*, while we refer to the set of code-bits on such a GSS as a *Variable-Node Stopping Set (VSS)*. In Fig. 3.2, the set $\{x_3, x_4, x_5, x_6\}$ is the VSS for the depicted GSS. As we will see later, every GSS must include some information bits and some code-bits. Thus, VSS is nonempty for each GSS. Accordingly, we define a *minimum VSS* ($MVSS$) as a VSS with minimum number of code-bits among all the VSSs. That is, a minimum VSS is the set of code-bits on a GSS with minimum number of code-bits among all GSSs. Note that a minimum VSS is not necessarily on a minimum GSS. We refer to the size of a minimum VSS as *stopping distance* of the code.

Now, for any given index set $J \subseteq \mathcal{A}$, there always exists an information bit $j \in J$ whose corresponding stopping tree has the smallest leaf set among all the elements in J. We call such an information bit a *minimum information bit* for J, denoted by $MIB(J)$. Note that there may exist more than one MIB in J. In general, any given index set $J \subseteq \mathcal{A}$ can be associated to several GSSs in the factor graph. We denote by $GSS(J)$ the set of all the GSSs that include J and only J as information bits. Each member of $GSS(J)$ includes a set of code-bits. The set of code-bits in each of these GSSs is a VSS for J. We refer to the set of these VSSs as *variable-node stopping sets* (VSSs) of J, denoted by $VSS(J)$. Among the sets in $VSS(J)$, we refer to the one with minimum cardinality as a *minimum VSS for* J, denoted by $MVSS(J)$.

3.3 Stopping Set Analysis of Polar Codes

In this section, we provide a stopping set analysis for polar codes. For the BEC, it is proved [57] that the set of erasures which remain when the decoder stops is equal to the unique maximal stopping set within the erased bits. In general, an analysis of the structure and size of the stopping sets can reveal important information about the error correction capability of the code. A minimum stopping set is more likely to be erased than larger stopping sets. Thus, minimum stopping sets play an important role in the decoding failure. In code design, codes with large minimum stopping sets are generally desired. We consider the problem of finding the minimum stopping set for a given polar code of length N. The results of this analysis may also help finding the optimal rule of choosing information bits to achieve the best error correction performance under belief propagation decoding.

3.3.1 Minimum VSS in The Graph

It is important to realize that what prevents the BP decoder from recovering a subset J of information bits is the erasure of the code-bits in one of the sets in $VSS(J)$. Therefore, what will eventually show up in any error probability analysis is the set of VSSs and their size. Particularly, $MVSS(J)$ represents the smallest set of code-bits whose erasure causes a decoding failure of J. We will find the size of $MVSS(J)$ for any given J. Furthermore, we will find the size of minimum VSS for a given polar code.

We start our analysis by stating some of the facts about the structure of stopping sets in the factor graph of polar codes. The factor graph of polar codes has a simple recursive structure which points to some useful observations. Here we mention some of these observations.

Lemma 1. *Any GSS in the factor graph of a polar code includes variable nodes from all columns of the graph. In particular, any GSS includes at least one information bit and one code-bit.* □

This implies that any given GSS includes a nonempty VSS.

Lemma 2. *Each information bit has a unique stopping tree.* □

Lemma 3. *Any GSS in T_{n+1} is formed of a GSS in T^U_{n+1} and/or a GSS in T^L_{n+1}, and a number of variable nodes $v(i,1),\ i=1,...,N$.* □

This implies that any GSS in T_{n+1} induces a GSS in T^U_{n+1} and/or T^L_{n+1}. This can be also seen in Fig. 3.2. The stopping set shown in the figure induces a stopping set in each of T^U_{n+1} and T^L_{n+1}. Now, consider size of the leaf set for different stopping trees. Note that we have $f(1)=1,\ f(2)=2,\ f(3)=2,\ f(4)=4$, so on. In general, we can state the following facts about $f(\cdot)$.

Lemma 4. *For a polar code of length $N=2^n$, the function $f(\cdot)$ can be obtained as follows:*

$$\begin{aligned} &f(2^l)=2^l \quad \textit{for} \quad l=0,1,...,n, \\ &f(2^l+m)=2f(m) \quad \textit{for} \quad 1\le m\le 2^l-1, \quad 1\le l\le n-1. \end{aligned} \tag{3.1}$$

Thus $f(\cdot)$ is not necessarily an increasing function. □

Lemma 5. *For a given polar code of length N formed by the kernel F, and for any $i\in\mathcal{A}$, we have $f(i)=wt(\boldsymbol{r}_i)$. In other word, the size of the leaf set for any stopping tree is in fact equal to the weight of the corresponding row in the generator matrix. Particularly, the leaf set of the stopping tree for any input bit represents the locations of 1's in the corresponding row of the matrix $F^{\otimes n}$.* □

Now, let us consider variable-node stopping sets for $J\subseteq\mathcal{A}$. The following theorem is proved for $MVSS(J)$ in the Appendix. The proof uses facts 1, 3, and 4.

Theorem 1. *Given any set $J \subseteq \mathcal{A}$ of information bits in a polar code of length $N = 2^n$, we have $|MVSS(J)| \geq \min_{j \in J} f(j)$.* □

Theorem 1 sets a lower bound on the size of the $MVSS$ for a subset J of information bits. It also implies that the size of the minimum VSS for a polar code is at least equal to $\min_{i \in \mathcal{A}} f(i)$. However, we already know that the leaf set of the stopping tree for any node $i \in \mathcal{A}$ is a VSS of size $f(i)$. This leads us to the following corollary.

Corollary 1. *For a polar code with information bit index $\mathcal{A}$, the size of a minimum variable-node stopping set is equal to $\min_{i \in \mathcal{A}} f(i)$, i.e. the size of the leaf set for the minimum-leaf stopping tree.* □

Corollary 1 implies that in order to find the size of the minimum VSS, we need to find the information bit with minimum leaf stopping tree among all the information bits.

3.3.2 Size Distribution of Stopping Trees and Their Leaf Sets

We provide a method for finding the size distribution of stopping trees and their leaf sets. First, note that the recursive construction of the factor graph dictates a relationship between the size of stopping trees in T_{n+1} and T_n.

Theorem 2. *Let $\boldsymbol{A}_n$ and $\boldsymbol{B}_n$ be two vectors of length 2^n showing, respectively, the size of stopping trees and their leaf sets for all input bits in T_n. That is, $\boldsymbol{A}_n = [|ST(1)|\ |ST(2)|$*
... $|ST(2^n)|]$ and $\boldsymbol{B}_n = [f(1)\ f(2)\ ...\ f(2^n)]$. We then have

$$\begin{aligned} \boldsymbol{A}_{n+1} &= [\boldsymbol{A}_n\ 2\boldsymbol{A}_n] + \boldsymbol{1}_{n+1} \\ \boldsymbol{B}_{n+1} &= [\boldsymbol{B}_n\ 2\boldsymbol{B}_n], \end{aligned} \tag{3.2}$$

where $\boldsymbol{1}_{n+1}$ is the all-ones vector of length 2^{n+1}. □

These two recursive equations can be solved with complexity $O(N)$ to find the desired size distributions for a code of length N. Note that Lemma 4 can also be concluded from Theorem 2. Furthermore, Lemma 5 can be used to find the size of leaf set for a specific stopping tree within time $O(N)$.

3.3.3 Stopping Distance for Polar Codes

Theorem 2 gives the stopping distance for a finite-length polar code, when the set of information bits is known. However, it is not always easy to choose the optimal information set, particularly with large code-lengths. In order to approach this problem, we first show that a slight modification in the set of information bits may actually result in a larger stopping distance without a significant impact on the BER performance.

Theorem 3. *In the factor graph of a polar code of length N, the number of input bits $v(i,1)$ for which $f(i) < N^{\epsilon},\ 0 < \epsilon < \frac{1}{2}$ is less than $N^{H(\epsilon)}$.* □

The above theorem implies that, for any $0 < \epsilon < 1/2$, we can always replace $N^{H(\epsilon)}$ information bits by some frozen bits for which the stopping tree has a leaf set larger than N^{ϵ}. It is easy to show that such a replacement does not effectively change the overall BER under BP, asymptotically. When $N \to \infty$ and $\epsilon < 1/2$, $N^{H(\epsilon)}$ will be vanishing with N. In a sparse factor graph, such as the one in polar codes, erroneous decoding of a small set of information bits affects only a few number (vanishing with N as $N \to \infty$) of other information bits. Therefore, given a finite number of iterations, BER will not change asymptotically. Accordingly, we can expect such a modification to have little impact on the BER performance in the finite regime, while resulting in a better error floor performance. Fig. 3.4 is used to demonstrate this case. The BER is depicted for Arıkan's rule and its modified version introduced above (we call it *new rule*) applied to a code of length 2^{13} and rate 1/2. We replaced information bits with leaf sets smaller than 2^8, by frozen bits with minimum Bhattacharyya parameter who

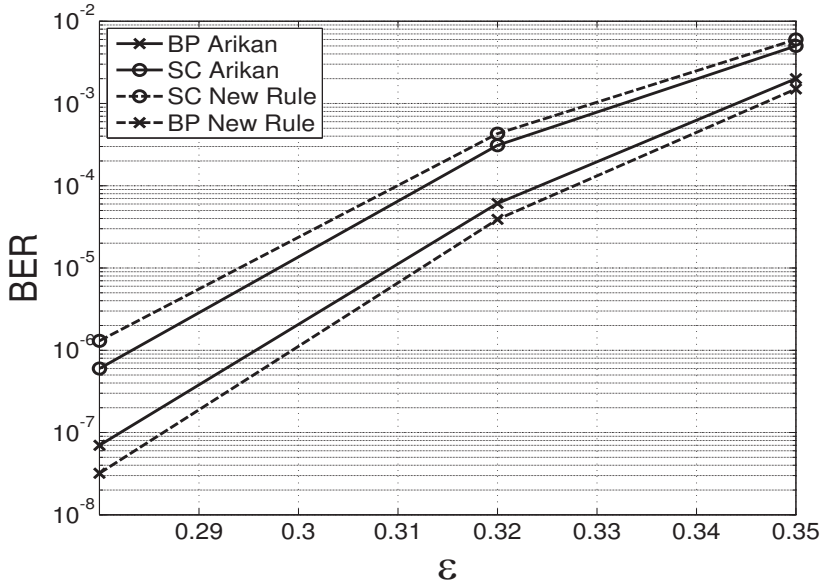


Figure 3.4. BER comparison for different methods of choosing information bits under BP and SC decoding. Code-rate and code-length are 1/2 and 2^{13}, respectively. The new rule improves the finite-length performance for BP while degrading it for SC decoding.

also had a leaf set larger than 2^8. As it can be seen, when SC decoding is used, the new rule performs slightly worse than Arıkan's rule. However, under BP decoding, it does slightly better than Arıkan's rule. While the figure only shows the BER performance in the waterfall region, We conjecture that this rule results in a superior error floor performance of the new rule due to its larger stopping distance. It is also noteworthy that if we use the new rule to pick all the information bits, i.e. if we only pick input bits with largest leaf sets as information bits, then the resulting code will be a Reed-Muller code for which BP performance is worse than polar codes [38]. Therefore, we only considered a limited use of the new rule. This apparently helps to preserve some of the good characteristics of polar codes while increasing the stopping distance. We also like to mention two points regarding the stopping distance.

3.3.3.1 Asymptotic Case

Theorem 3 asserts that given any capacity-achieving polar code and any $\sigma > 0$, we can always construct another capacity-achieving code with a stopping distance $N^{1/2-\sigma}$, by replacing some information bits by some frozen bits with larger $f(.)$. The following theorem gives the stopping distance for polar codes in the asymptotic case. Note that this only holds asymptotically and the analysis is different for finite-length codes, as we explained above.

Theorem 4. *The stopping distance for a polar code of length N grows as $\Omega(N^{1/2})$.* □

3.3.3.2 Minimum Distance vs. Stopping Distance

The following theorem states the relation between the stopping distance and minimum distance of polar codes.

Theorem 5. *The stopping distance of a polar code defined on a normal realization graph such as the one in Fig. 3.2, is equal to the minimum distance of the code, d_{min}.* □

According to Theorem 5, the number of code-bits in the minimum VSS grows as fast as the minimum distance. It is noteworthy that for linear block codes, d_{min} (i.e. the minimum Hamming weight among all codewords) puts an upper bound on the stopping distance [61–63]. This is because if all the ones in the received vector are erased, then it is impossible for the decoder to find out if an all-zero codeword has been sent or another codeword. For a code, it is a desirable property to have a stopping distance equal to its minimum distance. Therefore, Theorem 5 can be interpreted as a positive result, particularly compared to the capacity-approaching LDPC codes for which both the stopping and minimum distances are fairly small in comparison to the blocklength [61–63].

3.4 Error Floor Performance of Polar Codes

A large stopping distance is desirable in order to improve the error floor performance of a code over the BEC. After exploring the stopping sets of polar codes in the pervious section, here we focus on "girth" of polar codes as another important factor in error floor performance. Afterward, we examine the error floor performance of polar codes over the BEC and binary Gaussian channel via simulations.

3.4.1 Girth of Polar Codes

The *girth* of a graph is the length of shortest cycle contained in the graph. Cycles in the Tanner graph prevent the sum-product (BP) algorithm from converging [64]. Furthermore, cycles, especially short ones, degrade the performance of the decoder, because they affect the independence of the extrinsic information exchanged in the iterative decoding. When decoded by belief propagation, the external information at every variable node remains uncorrelated until the iteration number reaches half the girth. Hence, we are often interested in constructing large girth codes that can achieve high performance under BP decoding [65–67]. As it can be seen in the factor graph shown in Fig. 3.5, there exist two types of cycles: first, the cycles including nodes only from one of the top or bottom parts of the graph (shown by thick solid lines), and second, the cycles including nodes from both top and bottom parts of our symmetric graph (shown by thick dashed lines). The first type of cycles have the same shape in both upper and lower halves of the graph. The interesting fact about the cycles is that because the graph for a code of length 2^m is contained in the graph of a code of length 2^{m+1}, all the cycles of the shorter code are also present in the graph of the longer code. The shortest cycle appears in the graph of a length-4 polar code, as it is shown in Fig 3.5. It is a cycle of size 12, including 6 variable nodes and 6 check nodes. The shortest cycle of the second type appears first in the graph of a

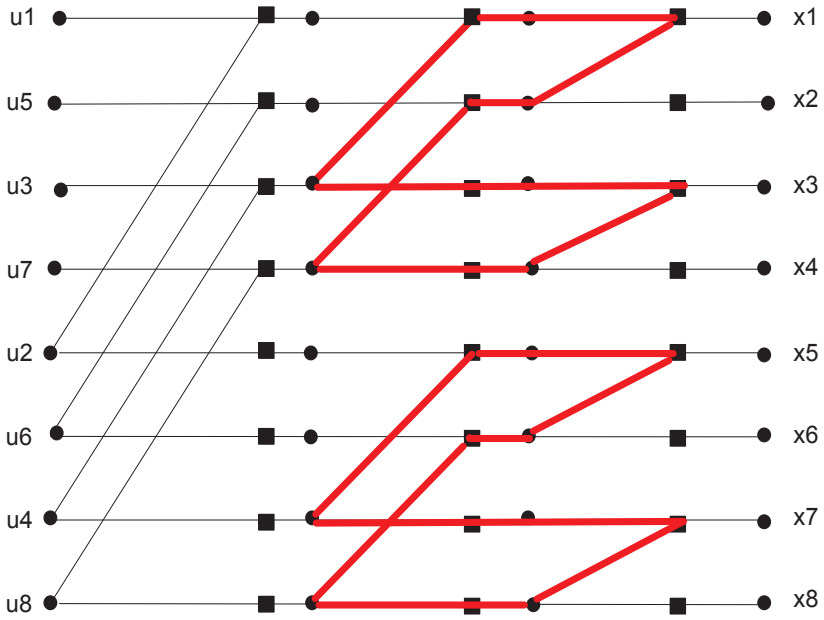
length-8 polar code, and have a size of 12 (dotted lines in Fig. 3.5). Thus, based on the above, the girth of a polar code is 12.

3.4.2 Simulation Results for Error Floor

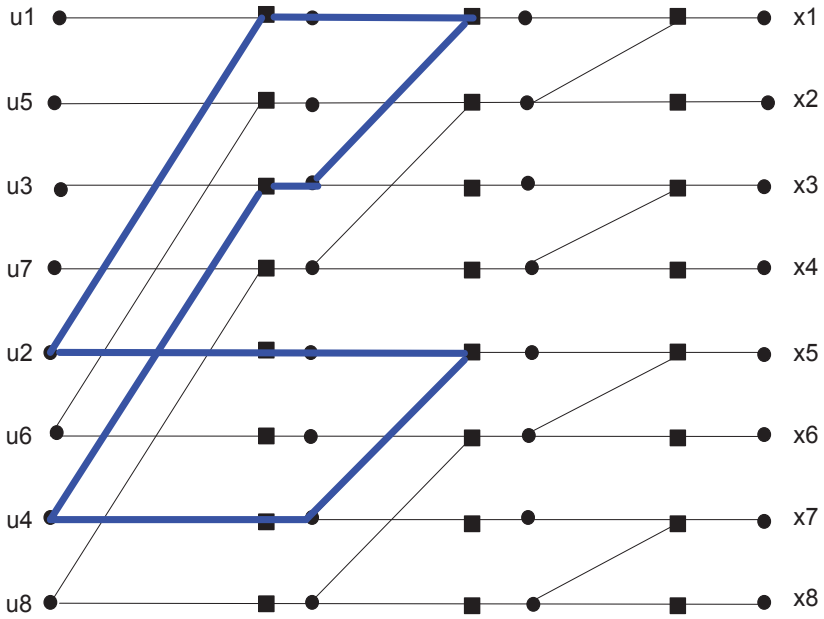
We performed simulations to examine the effect of the relatively large stopping distance and girth of the polar codes' factor graph on the error correction performance of these codes. Fig. 3.6 shows the simulation results for a code of length 2^{15} and rate 1/2 over the BEC. As can be seen, no sign of error floor is apparent. This is consistent with the relatively large stopping distance of polar codes. We indicated the 99% confidence interval for low BERs on the curve to show the precision of the simulation. Fig. 3.7 also shows the simulation results for a rate $\frac{1}{2}$ polar code of length 2^{13} over a binary-input Gaussian channel subjected to additive white Gaussian noise with zero mean and variance σ^2. The figure shows no sign of error floor down to the BERs of 10^{-9}.

3.5 Improved Decoding Using Guessing Algorithm

Fig. 3.8 provides a comparison between the bit error rate performance of BP and maximum likelihood (ML) decoding for polar codes over a BEC. As can be seen, ML decoding leads to error rates as large as four orders of magnitude lower than BP. This, along with relatively poor error rate performance of finite-length polar codes compared to LDPC codes, motivates us to find modifications to BP in order to improve its performance. Since LDPC uses belief propagation decoding, there have been various methods proposed to improve the BER performance of belief propagation in LDPC codes. Many of those ideas can be used for polar codes with a slight modification. However, as we have seen in the previous section, polar codes do not show error floor, benefiting from a large stopping distance and a relatively large girth. One of the schemes proved to be helpful for codes with such characteristics is to



(a) Shortest cycles in upper and lower halves.



(b) Shortest cycle spanning both upper and lower halves.

Figure 3.5. Different types of cycles in the factor graph of polar codes for $N = 8$. Thick solid lines show the first and second types of cycles, respectively. Figure shows a girth of 12 for polar codes.

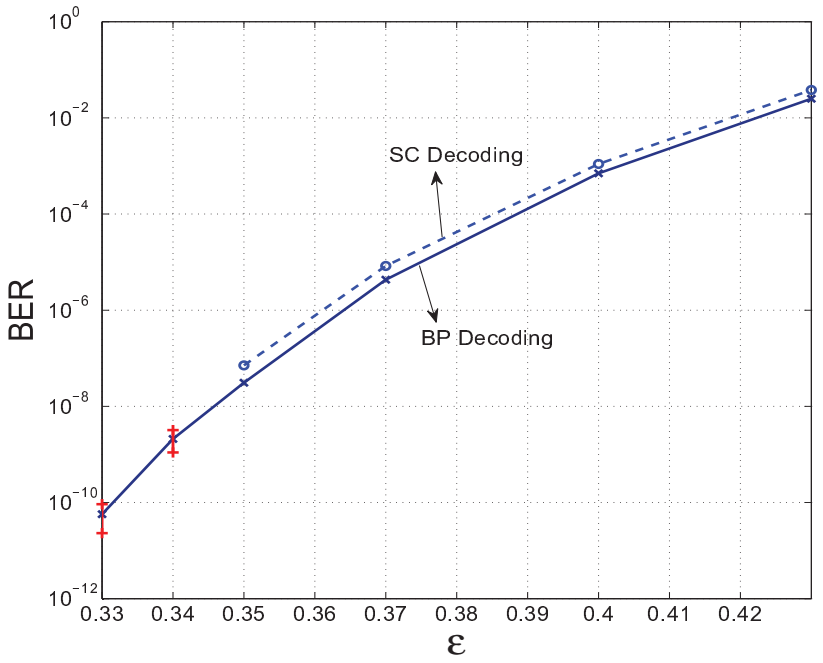


Figure 3.6. BER performance for BP and SC decoding over BEC. The code-length and code-rate are 2^{15} and $1/2$, respectively. The 99% confidence interval is shown for the two lowest BER's. No sign of error floor can be seen down to a BER of 10^{-10}.

use *guessing algorithms* alongside BP [58]. The key idea is the following observation. Consider a BEC with an erasure probability ϵ and a polar code of finite length N that has a small enough error probability. If the message-passing decoder fails to decode a received word completely, then there exist a few (usually less than or equal to 3 bits) undecoded bits such that if their values are exposed to the decoder, then the decoder can finish the decoding successfully. Note that this is true only when the BER is small enough (for example, less than 10^{-2}). Simulations and intuitive arguments strongly confirm the above statement.

In message passing algorithm basically, if the values of all but one of the variable nodes connected to a check node are known, then the missing variable bit is set to the XOR of the other variable nodes, and the check node is labeled "finished". Message passing continues this procedure until all check nodes are labeled as finished or the decoding cannot continue further. Let us call this "algorithm A". We now explain

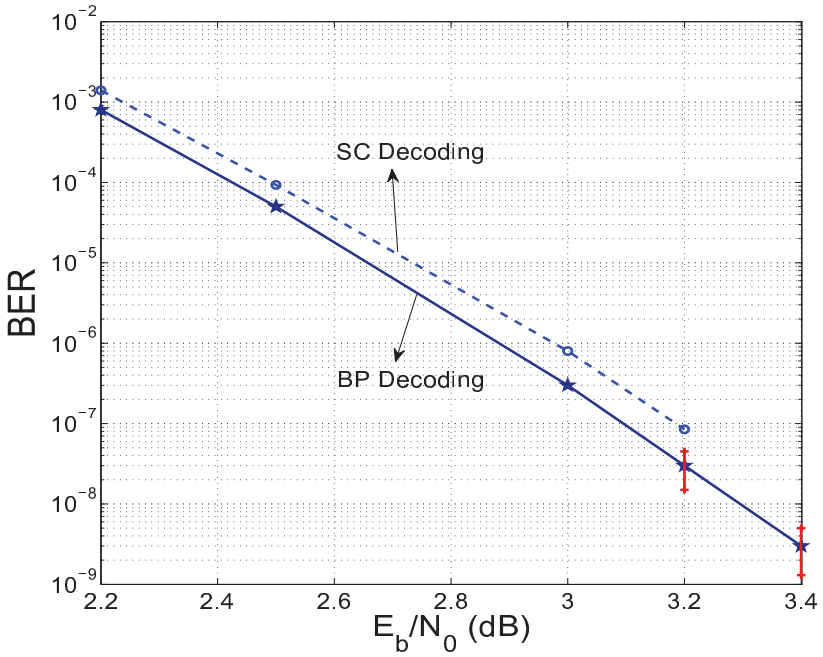


Figure 3.7. BER performance for BP and SC decoding over the Gaussian channel. The code-length and code-rate are 2^{13} and $1/2$, respectively. The 99% confidence interval is shown for the two lowest BER's. No sign of error floor can be seen down to a BER of 10^{-9}.

a modified message passing algorithm which we call "algorithm B". This algorithm continues the decoding when algorithm A fails to decode the received codeword. It chooses one of the unknown variable nodes, say w_1, and guesses its value (for example, by setting its value to zero). Intuitively, an appropriate scheme is to choose w_j that guessing its value frees as many as unknown variable nodes. In polar codes, since all variable nodes are degree 2 or 3, we choose variable nodes of degree 3 to guess their values. Then the algorithm continues to label the check nodes as in algorithm A with one more option. If all the variable nodes connected to the check node are known, then if the check node is satisfied it labels that check node "finished," otherwise the check node is labeled "contradicted." The procedure is done sequentially and the algorithm continues to run until either all check nodes are labeled or the decoding cannot continue further.

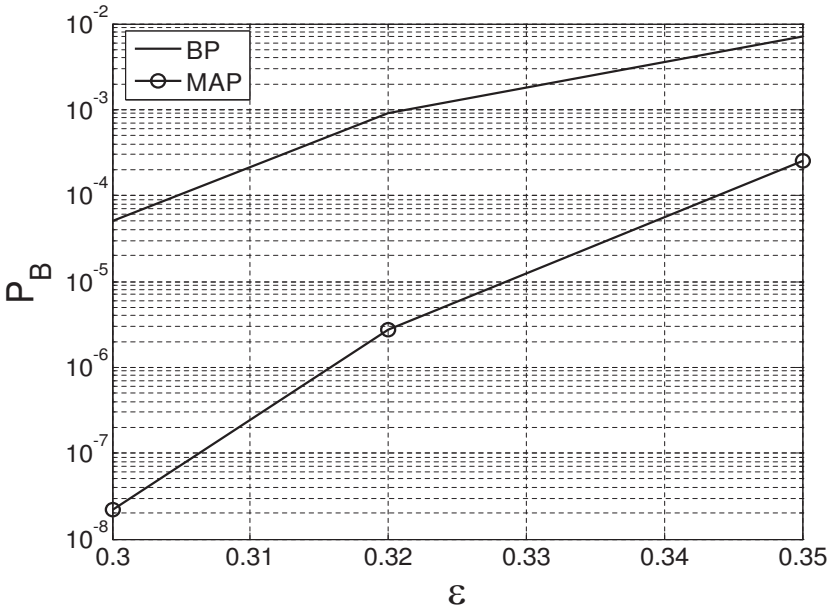


Figure 3.8. BER comparison of BP and MAP for a polar code of length 2^{10} and rate 1/2 over the binary erasure channel. BP stands far from the MAP which has the best finite-length performance. We propose a guessing algorithm to close this gap.

Once the procedure above is finished, if all of the check nodes are labeled and none of them is labeled "contradicted," the decoder outputs the resulting word as the decoded word. If all of the check nodes are labeled but some of them are labeled "contradicted," then it changes the value of w_1, the guessed variable node, and repeats the decoding from there. This time the decoding finishes successfully because we have found the actual value of w_1. But if the decoding stops again (i.e. some of the check nodes are not labeled) we have to choose another unknown variable node w_2 and guess its value to continue the decoding. Again, if some check nodes are labeled as "contradicted," we have to go back and try other values for w_1 and w_2. Obviously, Algorithm B is efficient only if the number of guesses is very small.

Algorithm B has a complexity that grows exponentially with the number of guesses. An improved algorithm called algorithm C was proposed in [58] to address this problem. Here we explain the basic idea of this algorithm. Let $w_1, w_2, ..., w_g$

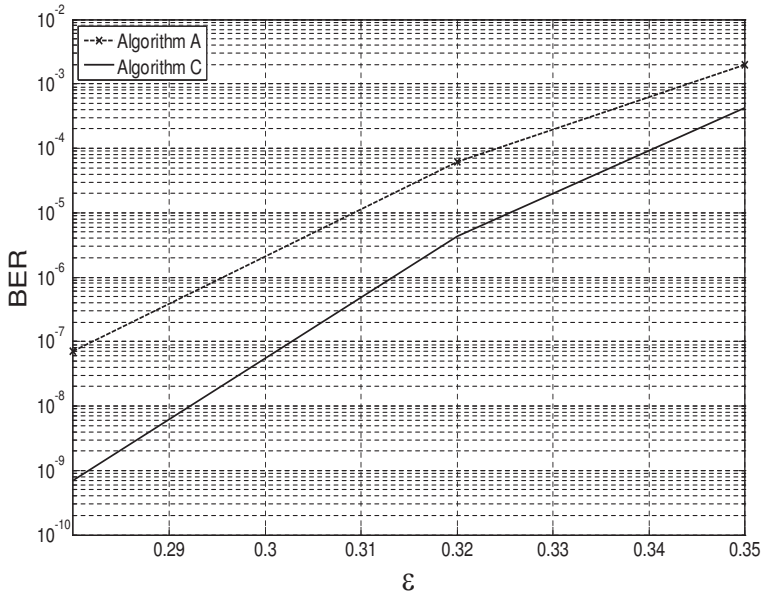
be the variable nodes that we guess and $x_1, x_2, ..., x_g$ be their values. In general, any variable node that is determined after the first guess can be represented as $a_0 \oplus a_1 x_1 \oplus a_2 x_2 \oplus ... \oplus a_g x_g$, where $a_j \in \{0, +1\}$. Algorithm C uses this parametric representation of the variable nodes to solve the set of equations obtained at the satisfied check nodes. This way, it finds the values of $x_1, x_2, ..., x_g$ and hence, the unknown variable nodes. It can be shown that this algorithm has complexity $O(g_{max}^2 N)$ where g_{max} is the maximum number of guesses [58]. We refer the reader to [58] for more details on this algorithm. Algorithm B can also be modified slightly to be used for the gaussian channel [59]. Since the basic idea is the same as the erasure channel, we omit the detailed discussion of this case here. We will show the simulation results for both cases in the next section.

3.5.1 Simulation Results

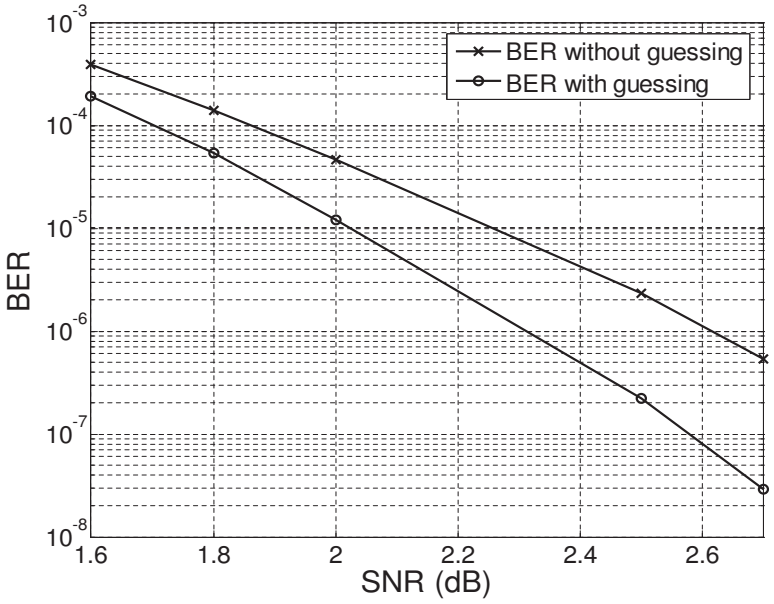
Fig. 3.9(a) shows the simulation results for BER over the BEC, where Algorithm C is compared to algorithm A. Note that Algorithms B and C show almost the same BERs. We have run our simulations for a rate 1/2 polar code of length 2^{13} while we set g_{max} to 6. As it can be seen in the figure, Algorithm C shows two orders of magnitude improvement in BER over Algorithm A. We also observed that the average running time of Algorithm C was about 1.04 times of Algorithm A. The average number of guesses is 3.07 when $\epsilon = 0.32$. Fig. 3.9(b) shows the simulation results for employing the guessing algorithm in the gaussian channel. The code we are using is of length 2^{13} and has a rate of $\frac{1}{2}$. The maximum number of guesses g_{max} is set to 6. As it can be seen, there is about 0.3 dB improvement in the BER of 2×10^{-6}.

3.6 Chapter Summary

As a first step in a practical approach to polar codes, we studied the BER performance of finite-length polar codes under belief propagation decoding. We analyzed



(a) BER performance of BP with guessing algorithm for decoding over BEC. Code-length is 2^{13} and code rate is 1/2. Guessing leads to more than an order of magnitude of improvement in BER performance.



(b) BER performance of BP with guessing algorithm for decoding over the gaussian channel. Code-length is 2^{13} and code rate is 1/2. Guessing leads to more than 0.3 dB improvement in BER performance at 10^{-6}.

Figure 3.9. BER performance of BP with guessing algorithm over the binary erasure and gaussian channels. g_{max} is set to 6, meaning that we guess 6 bits at most.

the structure of stopping sets in the factor graph of polar codes as one of the main contributors to the decoding failure and error floor over the BEC. The size of the minimum stopping set and the girth of the factor graph have been found for polar codes. We then investigated the error floor performance of polar codes through simulations where no sign of error floor was observed down to BERs of 10^{-10}. Motivated by the good error floor performance, we applied a guessing algorithm to improve the performance of BP decoding. Our simulations showed that using such a modified version of BP decoding can result in up to 2 orders of magnitude improvement in BER.

3.7 Proofs

Proof of Lemma 1: First, note that we only have degree 2 and 3 check nodes in the graph. In every Z-shaped part there are two check nodes, one at the top and one at the bottom. The top check node is always of degree 3 and the bottom one is always of degree 2. When a check node is a neighbor of a variable node or a set of variable nodes, we say that the check (variable) node is *adjacent to* that variable (check) node or the set of variable (check) nodes. We show that if a GSS is adjacent to either one of these check nodes in the ith column, then it must involve check nodes and variable nodes from both $(i-1)$th and $(i+1)$th columns. Therefore, any GSS includes variable nodes from all columns of the graph, including information bits and code-bits.

We consider two cases. Since each neighboring check node of a GSS needs to be connected to at least two variable nodes in the set, if the bottom check node is adjacent to the GSS, then both of its neighboring variable nodes must be in the set. Since all the check nodes connected to a variable node in the GSS are also adjacent to the set, this means that some of the check nodes in the $(i-1)$th and $(i+1)$th columns are also adjacent to the set. In the second case, if the upper check node (of

degree 3) is adjacent the GSS, then its neighbors in the GSS are either a variable node at its right and one at its left, or two variable nodes at its left, one at the top and one at the bottom of the Z. In the former case, the GSS clearly includes nodes from the $(i-1)$th and $(i+1)$th columns. In the latter case, the bottom variable node has the bottom check node as its neighbor in the GSS, leading to the same situation we discussed above. □

Proof of Lemma 2: Suppose an information bit i has two non-overlapping stopping trees, ST and ST'. Also, suppose ST has a form like the stopping tree shown in Fig. 3.3. That is only one variable node from each Z can participate in ST. Also, Note that a check (variable) node in the graph is adjacent to only one variable (check) node on the right (left). Thus, if a check node is adjacent to ST, it is adjacent to exactly one variable node on the left and one on the right.

Now assume that the difference between ST and ST' starts at the jth column. $j \neq 1$ Since, by definition, a stopping tree can include only one information bit; hence, $v(i,1)$ is the only variable node of column 1 participating in ST and ST'. Suppose there exists a variable node $v(k',j) \in ST', j \neq 1$, which is not part of ST. $v(k',j)$ is adjacent to $c(k',j-1)$ from left. However, $c(k',j-1)$ can not be adjacent to ST, otherwise we would have $v(k',j) \in ST$ because of what we mentioned above. But $c(k',j-1)$ must be adjacent to at least one variable node in ST' form the left since it needs to be adjacent to at least two variable nodes in ST' (definition of a stopping set). Therefore, $c(k',j-1)$ is adjacent to at least one variable node in ST' in the $(j-1)$th column, which is not part of ST. This is contradiction since we assumed ST and ST' start to differ at the jth column. □

Proof of Lemma 3: Fact 1 implies that any GSS in T_{n+1} includes at least one information bit. Consider such a GSS. According to Lemma 1, this GSS includes a set of variable nodes in T^U_{n+1} and/or T^L_{n+1}. Let us denote these sets by S^U and S^L, respec-

tively. Now, it is easy to see that the variable and check nodes in S^U and S^L, if non-empty, still satisfy the conditions of a GSS. This is because $v(1,1), v(2,1), ..., v(N,1)$ are connected to the rest of the graph only through $c(1,1), c(2,1), ..., c(N,1)$. Therefore, for any GSS in T_{n+1}, the induced non-empty subsets in T^U_{n+1} and T^L_{n+1} also form a GSS for these subgraphs. □

Proof of Lemma 4: This lemma can be concluded directly by looking at the recursive structure of the factor graph. □

Proof of Lemma 5: This is true because based on the Arıkan's paper, the encoding graph of polar codes is obtained from the matrix $F^{\otimes n}$. In fact, this graph is a representation of the recursive algebraic operations in this Kronecker product. □

Proof of Theorem 1: We prove the theorem by induction on n where $N = 2^n$ is the code-length. For $n = 1$ ($N = 2$), there are only two information bits, $v(1,1)$ and $v(2,1)$. It is trivial to check the correctness of the theorem in this case. Now suppose the hypothesis holds for a polar code of length 2^k. We prove that it also holds for a code of length 2^{k+1}. Consider a set J and let $MIB(J) = i$. In the case that there exist more than one MIB in J, without loss of generality, we pick the one with the largest index as the $MIB(J)$. That is, we pick the one which occupies the lowest place in the graph among the MIBs of J. Let VSS^* be a minimum VSS for J, and let GSS^* be the corresponding GSS for VSS^*. We also denote the upper and lower halves of the factor graph by G_U and G_L, as it is shown in Fig. 3.10(a). Note that G_U and G_L are identical in shape, and each of them includes half of the variable and check nodes in the factor graph. Without loss of generality, we assume that VSS^* includes variable nodes (code-bits in this case) from both G_U and G_L. We denote these two subsets of VSS^* by VSS^*_U and VSS^*_L, respectively. Also, GSS^* includes some variable nodes from the second column, i.e. from $v(1,2), ..., v(N,2)$. Let us denote the index set of these nodes by J'. For example, for the GSS shown in Fig.

3.2, J' is $\{2,4,6\}$. We also denote the subsets of J' in the upper and lower halves of the graph by J'_U and J'_L, respectively. Furthermore, We simply use T^U and T^L instead of T^U_{k+1} and T^L_{k+1}, since it is clear that we are dealing with the case $n = k+1$. Accordingly, we use $f_U(j')$ ($f_L(j')$) to show the size of the leaf set for the stopping tree of $j' \in J'_U$ ($j' \in J'_L$) in T^U (T^L).

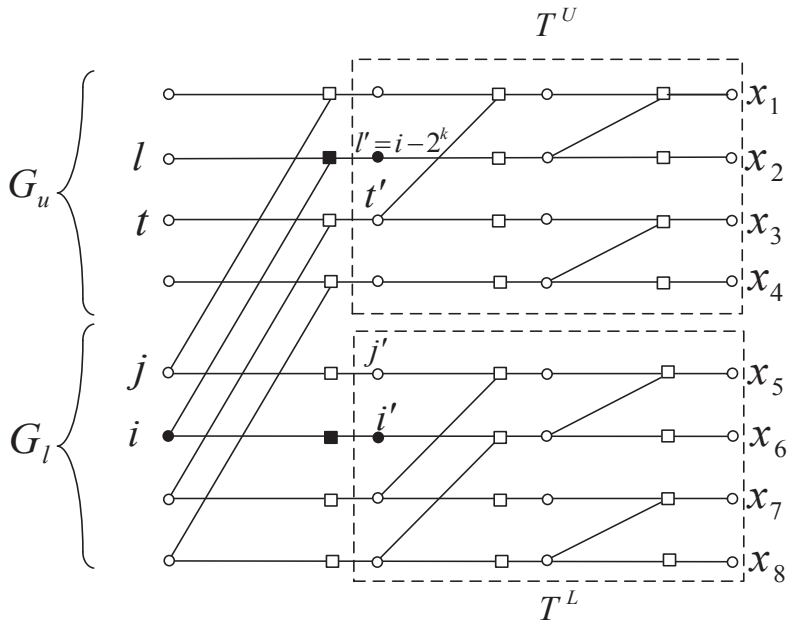
For this setting, we need to show that for bit i to be erased, at least $f(i)$ code-bits must be erased, or equivalently, $|VSS^*| \geq f(i)$. We consider two cases: 1. $i \in G_L$, and 2. $i \in G_U$.

1. $i \in G_L$: This case is depicted in Fig 3.10(a). First, note that $i - 2^k$ can not be in the VSS^*, because $f(i-2^k) = 1/2 f(i)$ and then i would not be a MIB. Now, for i to be erased, i' and $l' = i' - 2^k$ must be erased. Fact 3 asserts that J induces two stopping sets in T^U and T^L for J'_U and J'_L, respectively. We claim that i' and l' are MIB for J'_L and J'_U, respectively. If $i' \neq MIB(J'_L)$, then there exists a node j' such that $f_L(j') < f_L(i')$. Then, there exists $j \in \mathcal{A}$ such that $f(j) < f(i)$ which is in contradiction with the fact that i is a MIB.

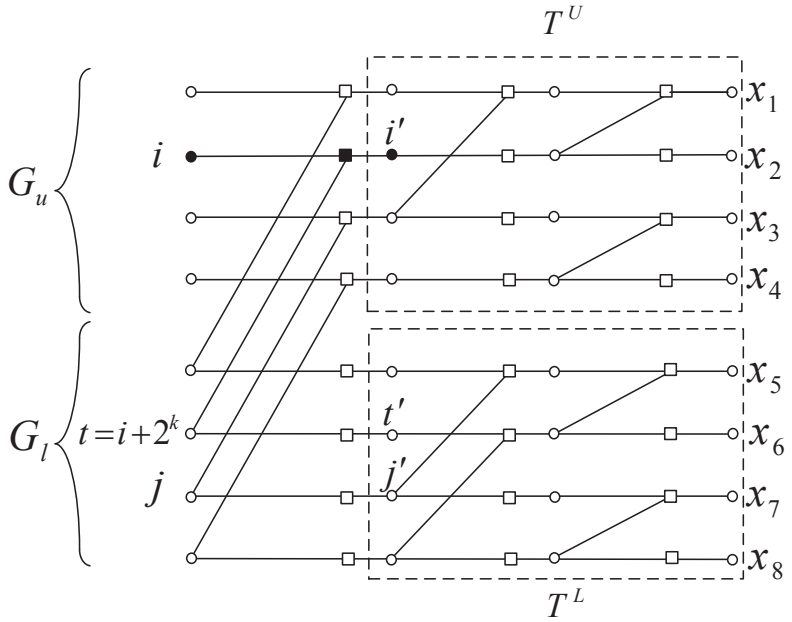
   If $l' \neq MIB(J'_U)$, then there exists t' such that $f_U(t') < f_U(l')$. This means that we have $t \in J$ and/or $t + 2^k \in J$. However, we then have $f(t) < f(i)$ and $f(t+2^k) < f(i)$, which is again a contradiction with i being a MIB. Now, since $i' = MIB(J'_L)$ and $l' = MIB(J'_U)$, then the induction hypothesis implies that $|VSS^*_L| \geq f_L(i')$ and $|VSS^*_U| \geq f_U(l')$. Therefore,

$$|VSS^*| = |VSS^*_L| + |VSS^*_U| \geq f_L(i') + f_U(l') = f(i).$$

2. $i \in G_U$: This case is depicted in Fig. 3.10(b). If $J \cap G_L = \phi$, then we can prove that $i' = MIB(J'_U)$ along the same lines as the proof of case 1 above. Then the induction hypothesis implies that $VSS^* \geq f_U(i') = f(i)$, and the proof would be complete for this case.



(a) Case 1 in Theorem 1.



(b) Case 2 in Theorem 1.

Figure 3.10. Figure is used to visualize different cases considered in the proof of Theorem 1.

Now suppose that $J \cap G_L \neq \phi$. Consider any $j \in J \cap G_L$. We show that $f(j) > f(i + 2^k)$. Let us denote $i + 2^k$ by t. First note that $f(j) > f(i)$; otherwise if $f(j) = f(i)$, then according to our definition of MIB, we would pick j as the MIB since $j \in G_L$ and $i \in G_U$. Also note that $f(.)$ only takes value as powers of 2. Hence, we have $f(j) \geq 2f(i)$. Therefore,

$$f_L(j') = 1/2 f(j) \geq f(i) = f_L(t'). \tag{3.3}$$

As a result, $|VSS^*| \geq |VSS_L^*| \geq f_L(t') = f(i)$. □

Proof of Theorem 2: The theorem becomes clear by looking at the recursive structure of the graph: T_{n+1} is formed of two copies of T_n, one at the top and one at the bottom, that are connected together. □

Proof of Theorem 3: In the matrix $F^{\otimes n}$, there are $\binom{n}{i}$ rows with weight 2^i [39]. This means that in the factor graph of a polar code, there are $\binom{n}{i}$ stopping trees with a leaf set of size 2^i. Thus the corresponding tree of these input bits is at least of size 2^i. As a result, the number of input bits with less than $2^{\epsilon n} = N^{\epsilon}$ variable nodes in their tree is less than $\sum_{i=0}^{\epsilon n} \binom{n}{i}$, which is itself upper-bounded by $2^{H(\epsilon)n} = N^{H(\epsilon)}$ for $0 < \epsilon < \frac{1}{2}$. □

Proof of Theorem 4: The block error probability for SC decoding over every B-DMC is proved to be $O(2^{-\sqrt{N}})$ [68]. Noting that the error correction performance of BP is at least as good as SC over the BEC [39], we conclude that block error probability for BP over the BEC decays as $O(2^{-\sqrt{N}})$ as well. Let us denote by $P_B(E)$ and $\Pr\{E_{MVSS}\}$, the block erasure probability and the probability of MVSS being erased. We then have

$$\begin{aligned}\Pr\{E_{MVSS}\} =& \epsilon^{|MVSS|} = (1/\epsilon)^{-|MVSS|} \leq P_B(E) \\ &= O(2^{-\sqrt{N}}) \Rightarrow |MVSS| = \Omega(\sqrt{N}),\end{aligned} \tag{3.4}$$

where ϵ is the channel erasure probability. □

Proof of Theorem 5: First note that according to Lemma 5, $f(i) = wt(\mathbf{r}_i)$ for any $i \in \mathcal{I}$. On the other hand, according to [39, 56], $d_{min} = \min_{i\in\mathcal{A}} wt(\mathbf{r}_i)$ for a polar code. Now using Corollary 1, $d_{min} = \min_{i\in\mathcal{A}} wt(\mathbf{r}_i) = \min_{i\in\mathcal{A}} f(i) = |MVSS|$. □

CHAPTER 4

APPLICATION-SPECIFIC DESIGNS FOR POLAR CODES

4.1 Introduction

There are usually limitations that are imposed by the specific applications, e.g., we might want to have rate-adaptive codes to cope with a time-varying channel. In this part of the dissertation, we aim at identifying applications that are best addressed by polar codes. It is well-known that finite-length polar codes show poor error probability performance when compared to some of the existing coding schemes such as LDPC and Turbo codes. Nevertheless, showing a set of good characteristics such as being capacity-achieving, low encoding and decoding complexity, and good error floor performance suggests that a combination of polar coding with another coding scheme could eliminate shortcomings of both, hence providing a powerful coding paradigm.

In this chapter, we consider the design of polar code-based concatenated coding schemes that can contribute to closing the gap to the capacity. Concatenated coding has been studied extensively for different combinations of coding schemes. Furthermore, there have been many applications, such as deep space communications, magnetic recording channels, and optical transport systems that use a concatenated coding scheme [69–72]. A coding scheme employed in these applications needs to show strong error correction capability. Here, we investigate the potentials of using polar codes in a concatenated scheme to achieve very low error rates while avoiding error floor. While the idea of concatenated polar codes was first introduced in [73],

the problem of designing practical concatenated schemes using polar codes is yet to be studied. In [73], the authors study the classical idea of code concatenation using short polar codes as inner codes and a high-rate Reed-Solomon (RS) code as the outer code. It is shown that such a concatenation scheme with a careful choice of parameters boosts the rate of decay of error probability to almost exponential in the blocklength with essentially no loss in computational complexity. While [73] mainly considers the asymptotic case, we are interested in improving the performance in practical finite lengths.

In addition to the above, we introduce a universally capacity achieving rate-compatible design for polar codes. As it will be shown, polar codes are inherently well-suited for rate-compatible applications. Different approaches to rate-compatible polar coding will be studied. As a relevant yet very important topic, we will introduce the design of *non-uniform polar codes* over a set of parallel channels. We particularly provide a simple design of such codes with a rather impressive performance. At the end, we further argue that because of their inherent polarization, polar codes have great potential for unequal error protection.

Our main contributions in this chapter are as follows:

- We study the combination of polar codes and LDPC codes, suggesting a polar code as the outer code and a LDPC code as the inner code.
- We investigate the performance of polar-LDPC scheme in a real-world application, by comparing it against some of the conventional schemes used in Optical Transport Networks. Our results suggest that polar codes have a great potential to be used in combination with other codes.
- We present a simple rate-compatible scheme that can universally achieve the channel capacity for a set of channels, using the same encoder and decoder.
- We will study puncturing to design rate-compatible polar codes.

- We will present the results for a simple design of polar codes over the parallel sub-channels ("non-uniform polar codes"), showing the improvement achieved over the case of using separate codes for different sub-channels.

- We will investigate the application of polar codes in unequal error protection by observing the fact that in finite-length polar codes, different information bits face different channels by the design.

The rest of the chapter is organized as follows. We propose concatenated polar codes to be used in a real-world application in Section 4.2. Section 4.3 investigates different approaches towards rate-compatible polar codes. Section 4.4 studies non-uniform polar codes and proposes a simple design for them. Finally, Section 4.5 introduces the unequal error protection using polar codes. The results of this chapter have been published in [48] and [60].

4.2 Concatenated Polar Coding

Polar codes show a set of good characteristics that are needed in many real-world communication systems. Among these properties are good error floor performance, being capacity-achieving, and a low encoding and decoding complexity. In this section, we take advantage of these properties to design a polar code-based scheme as a solution to a practical problem. An *Optical Transport Network (OTN)*, as it is shown in Fig. 4.1, is a set of optical network elements connected by optical fiber links, able to transport client signals at data rates as high as 100 Gbit/s and beyond. These networks are standardized under ITU-T Recommendation G.709, and stand for an important part of the high data-rate transmission systems such as Gigabit Ethernet and the intercontinental communication network. A minimum BER of at least 10^{-13} is generally required in such systems [71,72]. Because of very high-rate data transmission, OTNs need to employ a low complexity coding scheme to keep the delay in a low

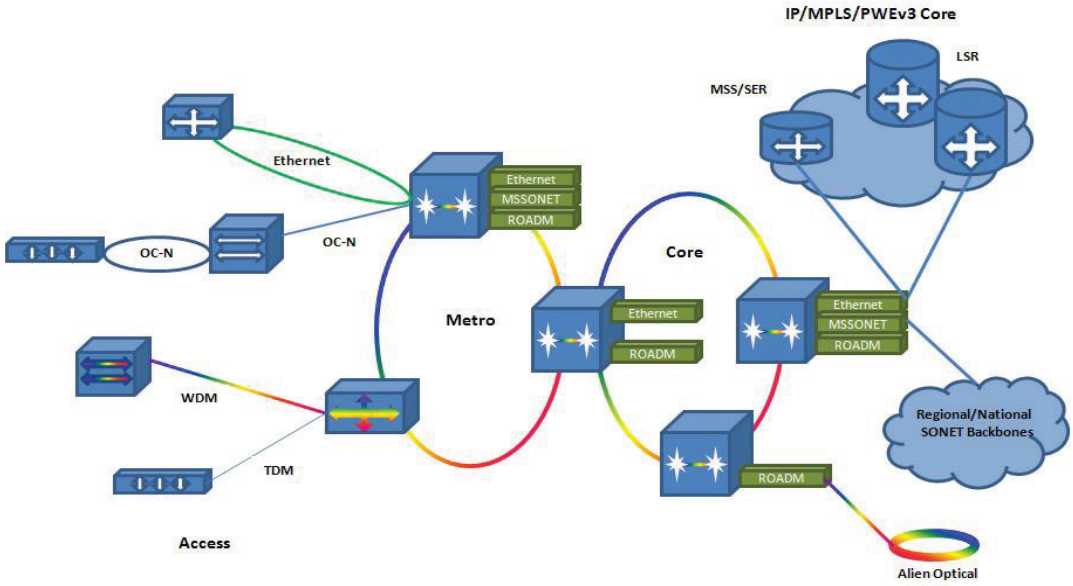


Figure 4.1. The structure of an Optical Transport Network (OTN) connecting network components using fibre optic cable. Channel coding for OTNs is standardized under OTU4 by ITU-T

level. Furthermore, these systems generally use a long frame for data transmission, which allows using large code-lengths.

We propose concatenated polar-LDPC codes to be used in OTNs. Our proposed scheme is formed of a Polar code as the outer code, and a LDPC code as the inner code. Fig. 4.2 shows the block diagram of this scheme. We consider long powerful LDPC codes as the inner code with rates close to the channel capacity. LDPC codes with good waterfall characteristics are known to mostly suffer from the error floor problem. However, the polar code plays a dominant role in the error floor region of the LDPC code. Based on the analysis provided in previous sections, the combination of polar and LDPC codes is expected to form a powerful concatenated scheme with a BER performance close to the capacity for a broad range of the channel parameter. We consider a binary polar code concatenated with a binary LDPC code. This is

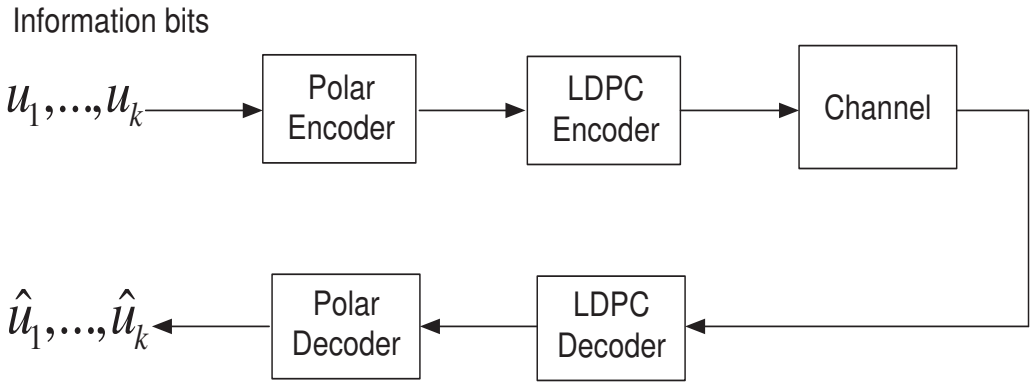


Figure 4.2. Block diagram of the proposed concatenated system of polar and LDPC codes for OTNs. We choose the LDPC code to be a capacity-approaching code.

different from the traditional concatenated schemes [5] in which a non-binary code is usually used as the outer code.

OTU4 is the standard designed to transport a 100 Gigabit Ethernet signal. The FEC (Forward Error Correction) in the standard OTU4 employs a block interleaving of 16 words of the (255, 239, 17) Reed-Solomon codes, resulting in an overall overhead of 7%. This scheme guarantees an error floor-free performance using a bounded distance decoder, and provides a coding gain of 5.8 dB at a BER of 10^{-13}. Since the approval of this standard (February 2001), several concatenated coding schemes have been proposed in the literature and some as patents, targeting to improve the performance of this standard. In most cases, these schemes propose a concatenation of two of Reed-Solomon, LDPC, and BCH codes [70–72, 74]. Here, for the first time, we consider polar-LDPC concatenation for the OTU4 setting.

4.2.1 Encoder

In order to satisfy the overhead of 7%, we adopt an effective code rate of 0.93. That is, if we denote the code-rates for the polar and LDPC codes by R_p and R_l respectively, then $R_{eff} = R_p \times R_l$ needs to be 0.93. The first problem is to find the optimal code-

rate combination for the two codes to achieve the best BER performance. While this is an interesting analytical problem, it might be a difficult problem to solve. Therefore, we find the best rate combination for our application empirically. First, note that both R_p and R_l are greater than 0.93. We are also aware of the relatively poor error rate performance of finite-length polar codes compared to LDPC codes. Therefore, in order to minimize the rate loss, we choose R_l close to the R_{eff}. As a result, R_p would be close to 1. The values of R_l and R_p can be found empirically. Fig. 4.3 shows the BER performance of three different rate couples, as a sample of all the rate couples we simulated. Code-length for the polar code is fixed to $2^{15} = 32768$ for all the rate couples. Showing a rate couple by (R_p, R_l), these three rate couples are (0.989, 0.94), (0.979, 0.95), (0.969, 0.96). We picked (0.979, 0.95) for the rest of our simulations in this section as it shows a better performance in the low-error-rate region. Fixing the code-length $2^{15} = 32768$ for the polar code and fixing the rates to (0.979, 0.95), the LDPC code-length would be 34493. We used the following optimal degree distribution pair which has a threshold value of 0.47 for the binary AWGN channel under BP [75]:

$$\begin{aligned}\lambda(x) =& 0.156935\ x + 0.138295\ x^2 + 0.325131\ x^3 \\ &+ 0.168818\ x^{11} + 0.210821\ x^{12}, \end{aligned} \tag{4.1}$$

$$\begin{aligned}\rho(x) =& 0.039239\ x^{34} + 0.144375\ x^{35} + 0.302308\ x^{70} \\ &+ 0.514078\ x^{71}. \end{aligned} \tag{4.2}$$

An interesting question here is how to design the polar code in this concatenated scheme, while the channel seen by the polar code is not an AWGN channel anymore. It is well known, that when the iterative BP decoder fails, the residual erroneous bits after decoding are organized in graphical structures (e.g. stopping sets on BEC or trapping sets for other types of channels). In order to find the distribution of

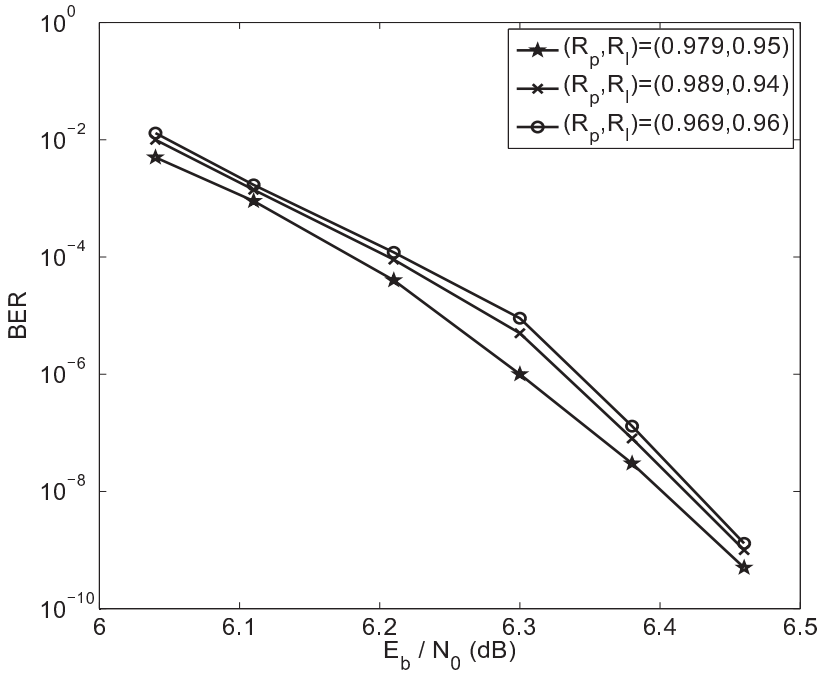


Figure 4.3. BER performance comparison for different rate combinations in a polar-LDPC concatenated scheme. We chose the rate pair of $(R_p, R_l) = (0.979, 0.95)$ for our concatenated scheme. The overall code-rate will be 0.93.

such patterns, one method is to prepare a histogram of these (post-decoding) error patterns. However, here we simply assume that the error patterns are distributed randomly (equally likely) at the output of the LDPC decoder, hence assuming the channel seen by the polar code as an AWGN channel with capacity 0.979. We then designed our polar code for this channel. The problem of designing optimal polar codes for this concatenated scheme remains as an interesting problem for further research.

4.2.2 Decoder

At the decoder side, we perform belief propagation decoding with soft-decision for both the polar and LDPC codes. Upon finishing its decoding, the LDPC decoder will pass its output vector of LLRs to the polar decoder. Polar decoder then treats this vector as the input for its belief propagation process.

4.2.3 Simulation Results

Fig. 4.4 depicts the BER performance for the concatenated scheme explained above, when using the LDPC code above. For the channel, we assumed a binary symmetric Gaussian channel as it is used by [70–72, 74]. Along with the concatenated scheme, we have shown the performance of the LDPC code when used alone with an effective rate of 0.93, which is equal to the effective rate of the concatenated scheme. As it can be seen, the concatenated scheme follows the performance of LDPC code in the waterfall region closely. Since both polar and LDPC codes here are capacity-approaching (capacity-achieving in case of polar codes), this technique does not suffer from rate-loss theoretically. Therefore, by increasing the code-length we expect the curve for polar-LDPC scheme to close the gap to capacity. The curve also shows no sign of error floor down to BERs of 10^{-10}, as opposed to the curve for LDPC code which shows error floor at around 10^{-8}. What actually happens in a polar-LDPC concatenation is that the two codes are orchestrated to cover for each other's shortcomings: LDPC plays the dominant role in its waterfall region, while polar code is dominant in the error floor region of the LDPC code.

We should also mention that a soft BP decoder is used with a 9 bit quantization (512 values) of the LLRs. We are also limiting the LLR values to the range of (-20, 20). The maximum number of iterations used in our simulations is 60; however, we counted the average number of iterations (let us call it the ANI) for LDPC and polar-LDPC schemes in order to get some ideas about their decoding latency. At a BER of 10^{-6}, the ANI for the capacity-approaching LDPC code when used alone was 11.3. On the other hand, the ANI for the LDPC and polar codes used in the polar-LDPC scheme was 13.1 and 16.7, respectively. It should be noted that the BP-Polar iterations are heavier than the iterations for LDPC due to the $N \log N$ time of each iteration in BP-Polar in comparison to the linear time of each iteration in BP-LDPC. In our simulations for the lower points in the curves, we kept sending blocks until we

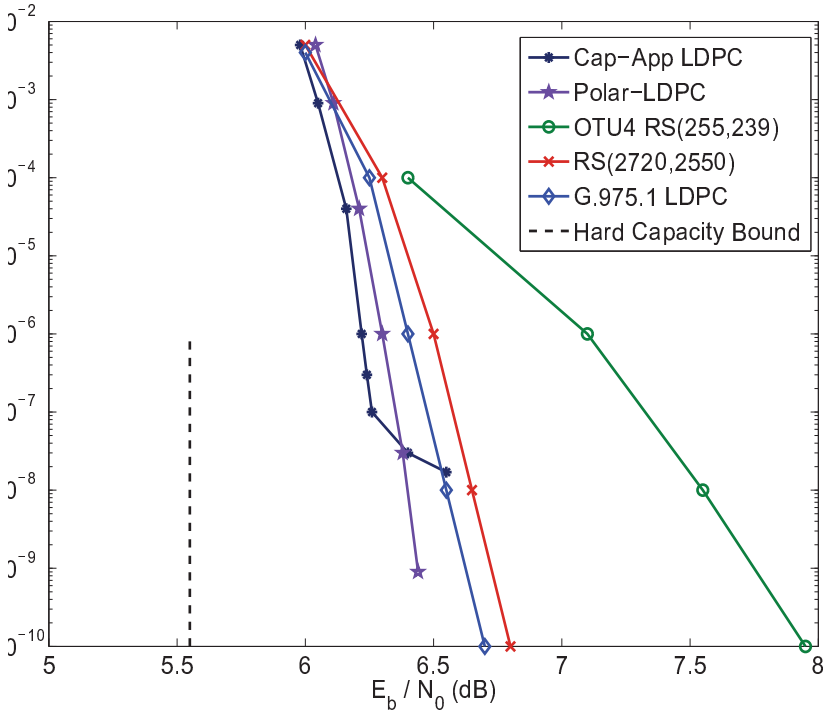


Figure 4.4. BER performance for different concatenated schemes. All the schemes have a code-length about 2^{15} and a code-rate of 0.93, as indicated by the standard. The polar-LDPC code has an edge over other schemes while showing no sign of error floor.

encounter 100 erroneous blocks. For example, for polar-LDPC curve at 6.4 dB (the lowest BER), we ended up simulating over 300 million blocks. This particular point took us the longest amongst all the simulated points. The lowest point in the cap-app LDPC curve was obtained by simulating about 30 million blocks.

In order to see the significant potential of polar codes for concatenated schemes, we compared the BER performance of the polar-LDPC approach against some of the existing coding techniques for OTNs, including the G.709 standard explained earlier in the section. We also included two "super FECs" proposed in ITU-T standard G.975.1 for high bit-rate DWDM (Dense Wavelength Division Multiplexing) submarine systems [76]. These schemes share some features, specifically the rate, blocklength, and

low decoding latency, with G.709, while achieving a much better performance. All the schemes use a code rate of 0.93. Furthermore, all of them are using codes of length around 2^{15}. We borrowed the BER curves of these schemes from [76].

As it is shown, an improvement of 1.3 dB at BER of 10^{-8} is achieved by polar-LDPC over the RS(255,239) of G.709 standard. Another scheme is an RS(2720,2550) with 12-bit symbols that has a blocklength of 32640 bits. It has been shown to achieve a significant coding gain and to have superior burst correction capabilities [76]. As it is shown, polar-LDPC concatenation achieves an improvement of 0.25 dB over this scheme. Presented in the figure is also the performance of a systematic binary LDPC code of length 32640, with 30592 information-carrying bits [76]. This LDPC code is suitable for implementation in current chip technologies for 10G and 40G optical systems offering low latency and feasibility of low power consumption in case of 40G implementation showing a significantly higher coding gain than the standardized RS code in G.709. As it can be seen, polar-LDPC shows an edge of 0.15 dB over this LDPC scheme. The decoding complexity for LDPC and RS codes is $O(N)$ and $O(N^2)$, respectively, while the polar-LDPC scheme has a complexity of $O(N \log N)$ which is closer to the LDPC code.

4.3 Rate-Compatible Polar Codes

An important practical issue is rate-compatibility over time-varying channels where error-correction codes are required to be flexible with respect to their code rates depending on the current channel state. In this section, we study polar codes for rate-compatible applications. We will show that polar codes are inherently well-suited for rate-compatible applications. We present a simple rate-compatible scheme that can universally achieve the channel capacity for a set of channels, using the same encoder and decoder. We will then study puncturing to design rate-compatible polar codes. Puncturing is widely used in the literature to generate rate-compatible

codes [59, 77, 78]. We will investigate the performance of random puncturing and stopping-tree puncturing (explained later) as used for polar codes and compare them to the universally capacity-achieving scheme. As it will be seen, the universally capacity achieving approach results in significantly better performance while having the same complexity as of puncturing.

Our goal is to provide reliable transmission over a set of channels with parameters $\theta^j, j = 1, ..., J$, using the same encoder and decoder in a rate-compatible fashion. Let W_j and $C(\theta^j)$ denote a channel with parameter θ^j and its capacity, respectively. Assume that $\theta^i < \theta^j$ and $C(\theta^i) > C(\theta^j)$ for $i > j$. We call a rate-compatible scheme *universally capacity achieving* (UCA), if the sequences of codes generated according to that scheme achieve the channel capacity $C(\theta^j)$ for $j = 1, .., J$. Assume that if $\theta^i < \theta^j$ then W_j is a degraded version of W_i, which we denote by $W_j \preceq W_i$. We may assume that the channel state information is available at both the transmitter and receiver. That is the transmitter at each time is aware of the exact value of the channel parameter and hence the channel capacity. Therefore, it can choose the appropriate code rate for communication. On the other hand, by knowing the channel capacity, receiver finds the set of frozen bits. Also, for a polar code of length N used over a B-DMC, W, we denote the N polarized channels by $W^{(i)}$, $i = 1, ..., N$. Recall from Chapter 3 that the index set of information bits is denoted by $\mathcal{A}$ where $\mathcal{A} \subseteq \{1, 2, ..., N\}$.

4.3.1 Universally Capacity Achieving Rate-Compatible Polar Codes

In this section, we present universally capacity achieving rate-compatible (UCARC) polar codes that can achieve the channel capacity for a set of channels, using a low complexity encoder. We first repeat Lemma 4.7 from [79].

Lemma 6. *Let W and $\tilde{W}$ be two symmetric BDMCs such that $\tilde{W}$ is a degraded version of W. Then, $\tilde{W}^{(i)}$ is degraded with respect to $W^{(i)}$ for $i = 1, ..., N$.*

This lemma implies that in a physically degraded setting, an order of polarization is maintained in the sense that "good" bits for the degraded channel, must also be "good" for the better channel. As a result, the set of information bits for the degraded channel is a subset of the set of information bits for the better channel, i.e. $\mathcal{A}_N(\tilde{W}) \subseteq \mathcal{A}_N(W)$.

Corollary 2. *Let $W_j,\ j = 1, ..., J$ be a set of symmetric BDMC channels such that $W_1 \preceq W_2 \preceq ... \preceq W_J$. Suppose that $\mathcal{A}(W_j)$ is known for $j = 1, ..., J$. Then, for any i and j such that $W_j \preceq W_i$, the capacity achieving polar code for W_j can be obtained from the polar code designed for W_i, by setting the input bits in $\mathcal{A}_N(W_i) \setminus \mathcal{A}_N(W_j)$ to zero in the encoder.*

This means that to implement different rates, the encoder only needs to shorten its set of information bits by switching a few of them to zero. This leads to a simple and practical structure for the UCARC polar codes. This can be considered as an important advantage of polar codes over other coding schemes such as LDPC and turbo codes for which finding a UCARC scheme can be very complicated if even possible. Fig. 4.5(a) shows the structure of encoder for a UCARC polar code of length N=8. As it is shown, the input bits can be switched by the encoder to operate either as an information bit or a frozen bit.

4.3.2 Puncturing for Rate-Compatible Polar Codes

In this section, we consider puncturing for rate-compatible polar codes. Figure 4.5(b) shows the encoder structure for the punctured rate-compatible polar codes. In this scheme, a parent code is designed for the worst channel (with largest channel parameter). In order to generate codes with higher rates for better channels, the encoder punctures some of the output bits. For every channel parameter $\theta_j,\ j = 1, ..., J$, a puncturing pattern is determined off-line and loaded into the encoder. The punctured bits will not be sent over the channel. In the decoder side, the log-likelihood

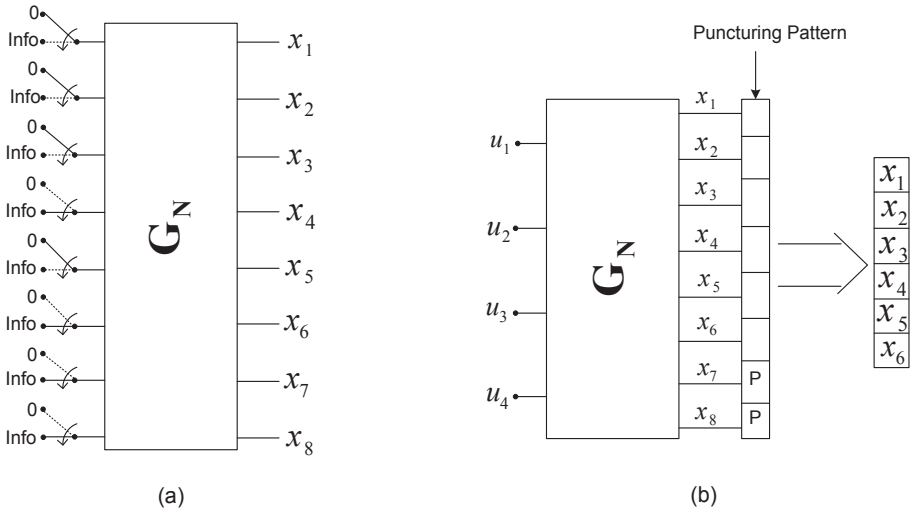
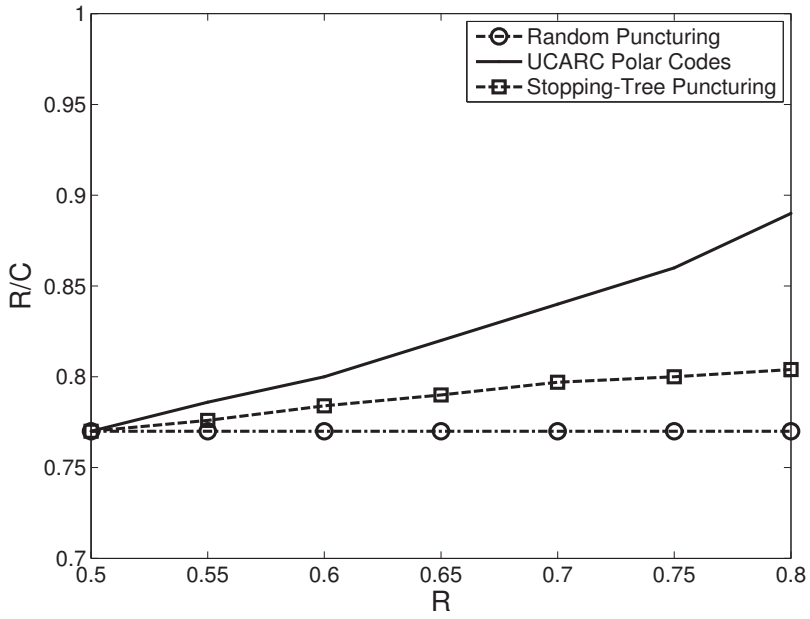


Figure 4.5. Different realizations for rate-compatible polar codes. (a)In a UCARC polar code, we can simply switch the input bits from information to frozen in order to change the code rate. (b) In punctured rate-compatible polar codes, we puncture the code-bits by not sending them over the channel, and hence changing the code rate.

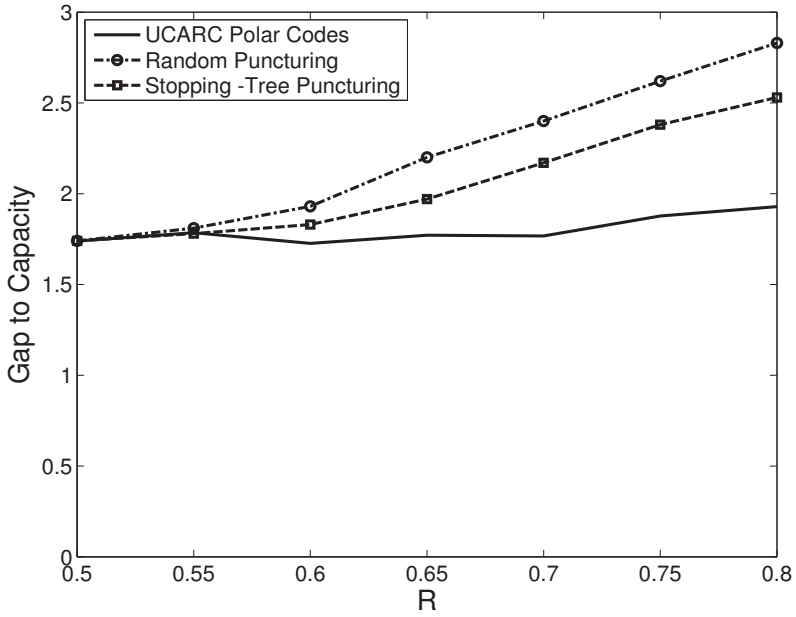
ratios for these bits will be set to zero before running belief propagation. The optimal puncturing pattern for each rate and a specific tanner graph can be found using optimization techniques [77,78]; however, it turns out it is difficult to use such techniques for polar codes.

4.3.2.1 Random Puncturing

A simple way of puncturing, which is studied in many papers, is to have the encoder choose the punctured bits for each rate randomly. Random puncturing is actually proved to be a UCARC scheme for LDPC codes over the BEC [59]. Fig. 4.6 shows the rate-to-capacity ratio (for the binary erasure channel) and gap-to-capacity (for the gaussian channel) for randomly punctured polar codes compared against the UCARC polar codes described in section 4.3.1. The rate-to-capacity ratio is constant for random puncturing over the BEC as it is expected based on the results in [59]. It is also interesting that the ratio for polar codes increases when the rate grows. As it can be seen in the figure, there is a substantial distance between the two curves.



(a) Rate-to-capacity ratio for different rate-compatible schemes over the BEC as rate increases. The UCARC scheme has the best performance.



(b) Gap-to-capacity for different rate-compatible schemes over the Gaussian channel. The UCARC scheme has the smallest gap-to-cap ratio as it follows the capacity.

Figure 4.6. Performance of different schemes when used over the BEC and the gaussian channel. Parent-code rate for punctured codes is 1/2, parent code-length is 2^{13}, and BER is fixed to 10^{-4}.

4.3.2.2 Stopping-Tree Puncturing for Polar Codes

Here, we propose an algorithm involving the stopping sets in the tanner graph to improve the performance of puncturing. Recall that a stopping set in the tanner graph was defined as a set of variable nodes such that every neighboring check node of the set is connected to at least two variable nodes in the set. Fig. 3.2 shows an example of the stopping set in the polar codes' graph. Also recall that a *stopping tree* in the polar codes' tanner graph is a stopping set shaped as a tree rooted at an information bit (on the left side of graph) and with leaves at the code-bits (on the right side of graph), such as the one shown in Fig. 3.2 with black variable and check nodes. We refer to such a tree as the stopping tree of an information bit.

For any code-bit in the tanner graph, we can find the number of stopping trees having that specific code-bit as a leaf node. Then, we pick the punctured code-bits from the ones which are present in the fewest number of stopping trees. This algorithm is based on the empirical results which show that the chance of recovery for these code-bits is higher than others in case that they are erased. In other words, these code-bits are better protected than others in the tanner graph. Since the information bits are known and the graph has a simple structure, we can easily find these bits. We call this algorithm *Stopping-Tree Puncturing*. As an example of this algorithm, suppose that we want to puncture the parent code of rate 1/2 in Fig. 4.5(b) to a code of rate 3/4. Then we need to pick 2 code-bits to puncture. Fig. 4.7 shows the factor graph representation of a polar code of length 8. It is easy to see that x_8 is the only bit that is present in only one stopping tree. Among the code-bits which are present in two stopping trees we can choose x_7.

Fig. 4.6 shows the simulation results for stopping-tree puncturing compared to other techniques. As it can be seen, the rate-to-capacity ratio has improved over the random puncturing though the distance to the UCARC scheme is still noticeable.

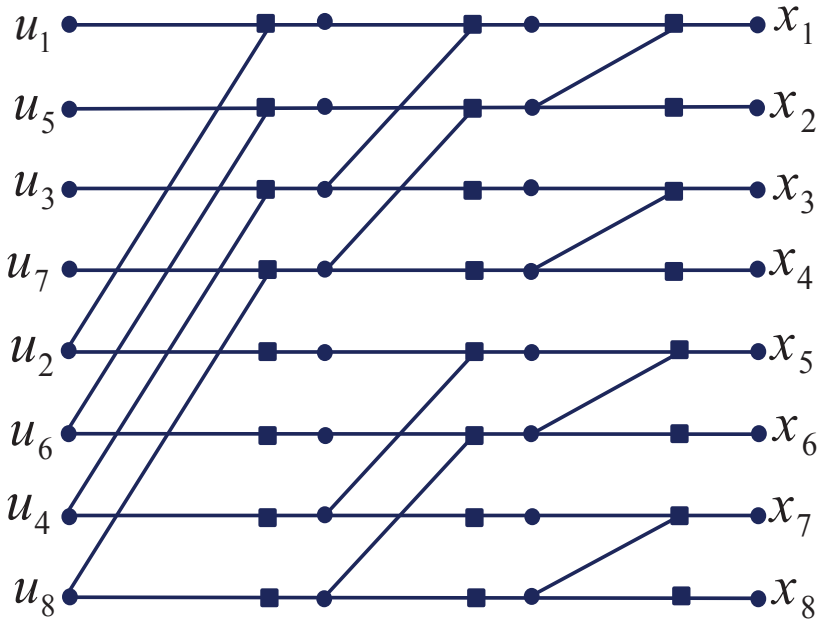


Figure 4.7. Factor graph representation of a polar code of length 8. x_8 is the only bit that is present in only one stopping tree. Among the code-bits which are present in two stopping trees we can choose x_7.

4.4 Polar Codes for Non-Uniform Channels

The puncturing problem can be thought of as coding over a set of parallel sub-channels. In puncturing, some of the code-bits are not sent over the channel. This can be modeled by two sub-channels, one is the same as the original channel and the other is a channel with capacity zero. Then, punctured bits will be assumed to be sent over the zero-capacity channel while other cod-bits are sent over the original channel. As discussed previously, coding over parallel channels (here referred to as non-uniform error correction) as well as the closely related problem of coding for unequal error protection are of practical importance.

One trivial approach to the problem of non-uniform error correction is to design a separate code for each of the channels. However, we are interested in designing only one polar code similar to the one presented in Fig. 4.8. As it is discussed in [80], this setting has several advantages over the trivial approach. It is important to note that

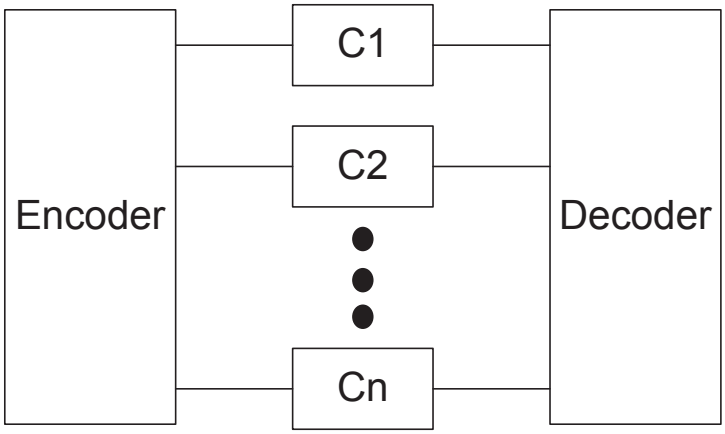


Figure 4.8. Non-uniform coding scheme for parallel sub-channels. The goal is to use only one pair of encoder-decoder for the system.

in this setting, the encoder knows that what set of code-bits are being sent over each sub-channel. This extra information in fact can be exploited to design practical polar codes for such non-uniform channels with an improved performance. Note that polar coding for permuted parallel channels is considered in [81] where different codewords are being sent over different channels. However, this is completely different from the problem of coding for non-uniform channels that we study here.

Here, we consider an approach based on the same idea used in Section 4.3.2 for stopping-tree puncturing. In other words, assuming only two sub-channels, we send more protected code-bits (code-bits that are present in more stopping trees) over the better sub-channel, while less protected bits will be sent over the worse channel. Fig. 4.9 compares the performance of this approach against using two separate decoders for the two channels. As it can be seen, despite its simplicity, this scheme achieves more than one order of magnitude improvement in BER. This idea can be considerably improved if we obtain better knowledge of how (to what degree) different code-bits are protected. Such an understanding can be achieved by analyzing the factor graph of the code. Therefore, an interesting future work is to use the results of finite analysis, discussed in Chapter 3, in finding better non-uniform schemes based on polar codes.

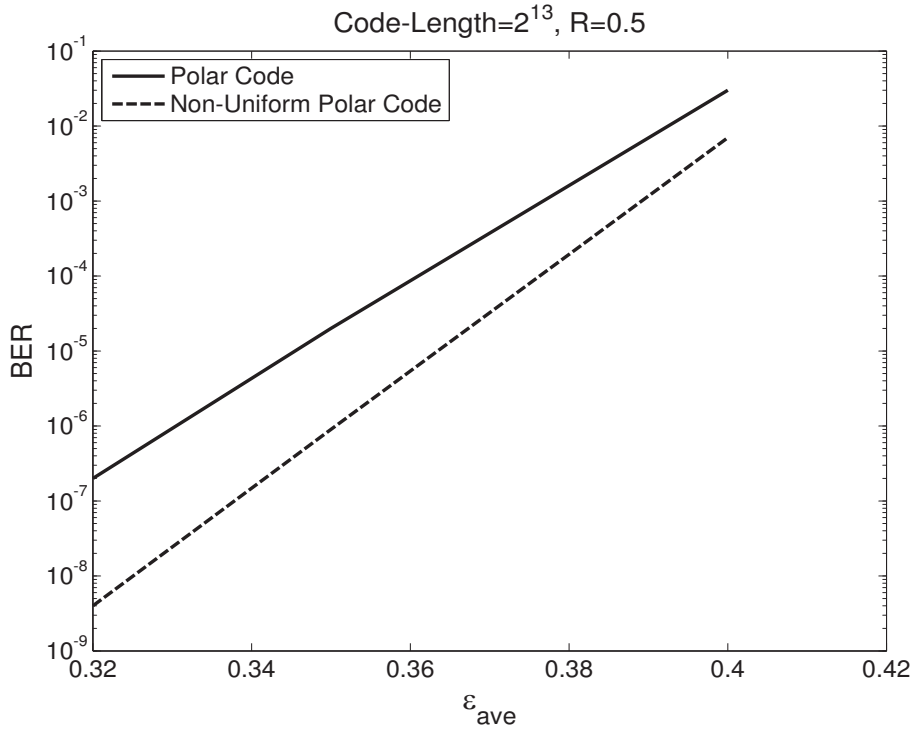


Figure 4.9. BER performance for non-uniform polar codes over two parallel channels. Non-uniform polar code performs better than the system with two separate encoder-decoders by an order of magnitude. The reason is mainly the extra information in the code design as well as the larger code-length in the non-uniform scheme.

Furthermore, because of the close connections to the puncturing problem, any scheme that can improve the performance of puncturing is expected to be used to design a non-uniform scheme. However, the proposed puncturing schemes need to be modified here since the setting is slightly different.

4.5 Unequal Error Protection Using Polar Codes

Along the same line lays the application of polar codes for unequal error protection. This problem is interesting because of the inherent characteristics of polar codes. Assume we design a polar code of length N and rate R for a given channel with capacity $C > R$. Also assume a SC decoder. In the process of design, we use bit-channels with smaller Bhattachrya parameters to send information [1]. While

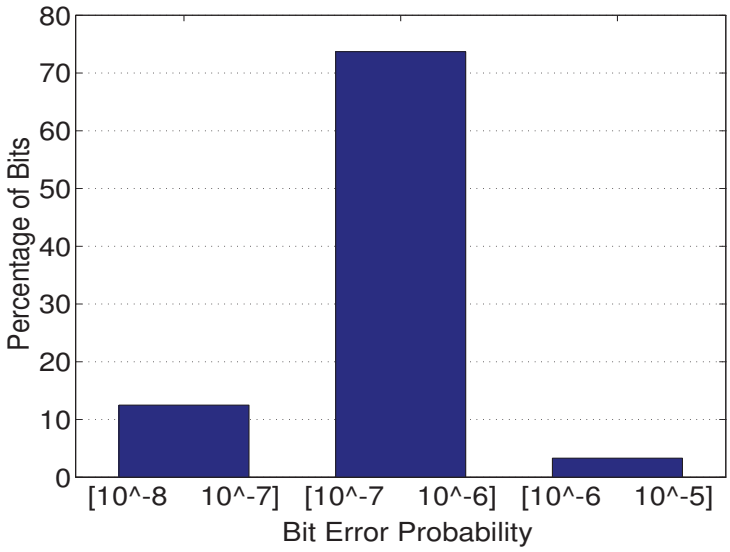


Figure 4.10. Distribution of bit error probability for the set of information bits. There is a difference of more than two orders of magnitude in the error rate for different information bits that can be used for UEP designs.

the value of Bhattachrya parameter becomes polarized when $N \to \infty$, the fact is that for finite lengths we will have a range of values spread all over the interval $[0, 1]$. Therefore, even after picking the good channels as information bits, there may still exist a considerable difference between these "good" channels. This means that some of the information bits will have lower error probability than others. In other words, polar codes inherently provide unequal protection to some degree. Fig. 4.10 shows the bit error probability distribution for the set of information bits in a polar code of length $N = 2^{15}$ and rate $1/2$, when used over a BEC. The figure shows the percentage of information bits that have a specific error probability. As it can be seen, there is a difference of more than two orders of magnitude in the error rate for different information bits. This property can be easily employed to design unequal error protection codes based on polar codes.

4.6 Chapter Summary

Motivated by good error floor performance, we proposed using polar codes in combination with other coding schemes. We particularly studied the polar-LDPC concatenation to be used in OTNs as a potential real-world application. Comparing the performance for our proposed scheme to some of the existing coding schemes for OTNs, we showed that polar-LDPC concatenation can achieve a significantly better performance.

We studied different approaches to rate-compatible polar codes. We showed that UCARC polar codes can be designed with low complexity using the inherent characteristics of polar codes. We also studied the use of puncturing to generate rate-compatible polar codes. We compared the performance of UCARC scheme against random and stopping-tree puncturing schemes through simulations. We observed that the UCARC scheme outperforms the puncturing-based methods while having a comparable complexity. The problem of non-uniform polar coding was studied in this dissertation for the first time, where we proposed a simple approach demonstrating the potentials in poalr codes for this specific application.

CHAPTER 5

FINITE WIRELESS NETWORKS

5.1 Introduction

There currently exists a vast amount of literature on the asymptotic analysis of different properties for large-scale random networks [2, 8, 9, 82–92]. However, in real world we have to face small or moderate-size networks which consist of a limited number of nodes. As it has been shown in [93, 94], asymptotic results often cease to be valid for such networks. In fact, asymptotic analyses make use of methods and approximations that can considerably simplify the analysis and results in simple and closed-form formulas for network properties. However, many of these techniques cannot generally be applied to the small-scale networks. Here, we study a model which is extensively used in analyzing the random networks: the random geometric graph. In a random geometric graph, vertices are distributed randomly according to a specific probability distribution and there exists an edge between any two vertices not more than a specific distance apart.

We first study the threshold phenomena for monotone properties in finite wireless networks modeled by random geometric graphs. A *monotone graph property* is a graph property such that if a graph H satisfies it, every graph G on the same vertex set obtained by adding edges to H also satisfies the property. In other words, a graph property is monotone if it is kept under the addition of edges. Note that many of the graph properties such as connectivity, bearing a complete subgraph of a specific size, or having a specific minimum degree are monotone properties. Fig. 5.1 shows how connectivity is a monotone property as it is preserved under the addition of edges.

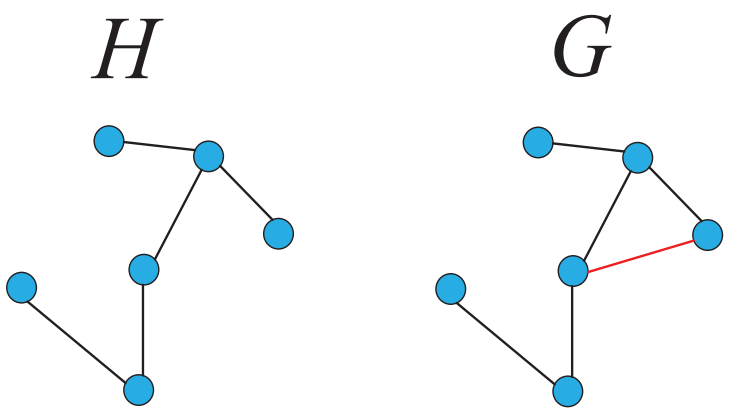


Figure 5.1. Connectivity is a monotone property. H is a connected graph because there is a path between any two nodes on the graph. As you can see, a graph obtained by adding edges to H will be connected as well.

What makes the monotone properties so interesting is that the probability of having a monotone property in a large random graph jumps from a value near 0 to a value close to 1 in a relatively short interval of the communication radius. The length of this interval- known as the threshold width- has been under close scrutiny in percolation theory, statistical physics, cluster analysis and some related issues in computer science, economics and political sciences. The asymptotic behavior of the threshold phenomena for random geometric graphs is well-studied in [2, 82, 85–87] where some upper bounds have been derived for the threshold width of the monotone properties. Here, we aim to analyze the threshold phenomena when the graph consists of a finite number of nodes. In this regard,

- we will find an upper bound for the threshold width of the monotone properties in finite one-dimensional random geometric graphs. Other models of random networks such as networks with random Poisson node deployment and unreliable sensor grids will also be considered as special cases. While previous studies on finite networks are limited to specific properties such as coverage and connectivity (see for example [93–98]), our method is a comprehensive one which leads to a bound, true for all monotone properties.

We then move on to study a non-monotone characteristic of finite wireless networks which is the MAC-layer capacity. The MAC-layer capacity is defined in [6] as the maximum possible number of concurrent transmissions at the medium access layer. Given a graph $G(V, E)$, the goal is to choose a subset of the edges on which transmissions can occur without conflicting with one another. As a MAC protocol, we adopt the simple model stated in [6] which also accounts for *virtual carrier sensing* (RTS/CTS signalling) used to resolve channel contention. In this model, if transmissions along (s, t) and (s', t') are occurring simultaneously, then none of the edges (s, s'), (s, t'), (s', t), (t, t') should be present in the graph. The set of edges that can be so chosen is called a *D2-Matching* (Distance-2 Matching). Here, we consider the problem of finding a D2-matching of maximum cardinality called *D2EMIS* [6]. MAC-layer capacity is zero when the communication radius, r, is zero and increases with r to some point after which increasing the radius leads to more interference and hence, a decline in the capacity. Therefore, MAC-layer capacity cannot be a monotone property. The problem of capacity has been investigated extensively for different models of wireless networks (see for example [8, 9]). However, almost all previous analytic results are asymptotic since they consider large-scale networks. In the second part of this chapter, we study the MAC-layer capacity in random line networks. The asymptotic MAC-layer capacity of ad hoc wireless networks is studied in [6]. However, the asymptotic result obtained there is not as precise when we consider finite networks [93]. In this chapter, we analyze the average MAC-layer capacity for finite line networks. Here,

- We obtain closed form expressions as lower and upper bounds for the MAC-layer capacity.
- We also provide an algorithm which finds the exact value for the MAC-layer capacity along with a set of active links which achieves it. This algorithm runs in linear time whereas the problem of finding the MAC-layer capacity for

the general two-dimensional case is proved to be NP-complete for the model considered in this chapter [99].

- We provide simulations showing that our bounds are good estimates for the exact values.

The rest of this chapter is organized as follows. In section 5.2, we derive upper bounds for the threshold width of one-dimensional finite networks. We follow on by analyzing the MAC-layer capacity of random line networks in section 5.3. The results of this chapter have been published in [100] and [101].

5.2 Threshold Phenomena in Finite Line Networks

In this section we provide an upper bound on the threshold width of finite wireless networks on a line. Consider n points distributed uniformly and independently in the d-dimensional unit cube $[0,1]^d$. Given a fixed distance $r > 0$, connect two points if their Euclidean distance is at most r. Such graphs are called random geometric graphs, and are denoted by $G(n,r)$, as in [102]. Fig. 5.2 shows a random geometric graph $G(n = 200, r = 0.125)$ on $[0,1]^2$.

In this section, we adopt a more general definition of random geometric graphs which allows for an arbitrary distribution of nodes. These graphs are denoted by $G(n,r,f(u))$ where $f(u)$ is the corresponding probability distribution function (PDF) of nodes' placement. Random geometric graphs are better suited than more combinatorial classes (such as Bernoulli random graphs) to model problems where the existence of an edge between two different nodes depends on their spatial distance. As a result, random geometric graphs have received increased attention in recent years in the context of distributed wireless networks such as sensor networks (see for example, [8, 82, 88, 103]). In these graphs, the probability of a monotone property is an increasing function of r. This is because by increasing r, new edges will be formed

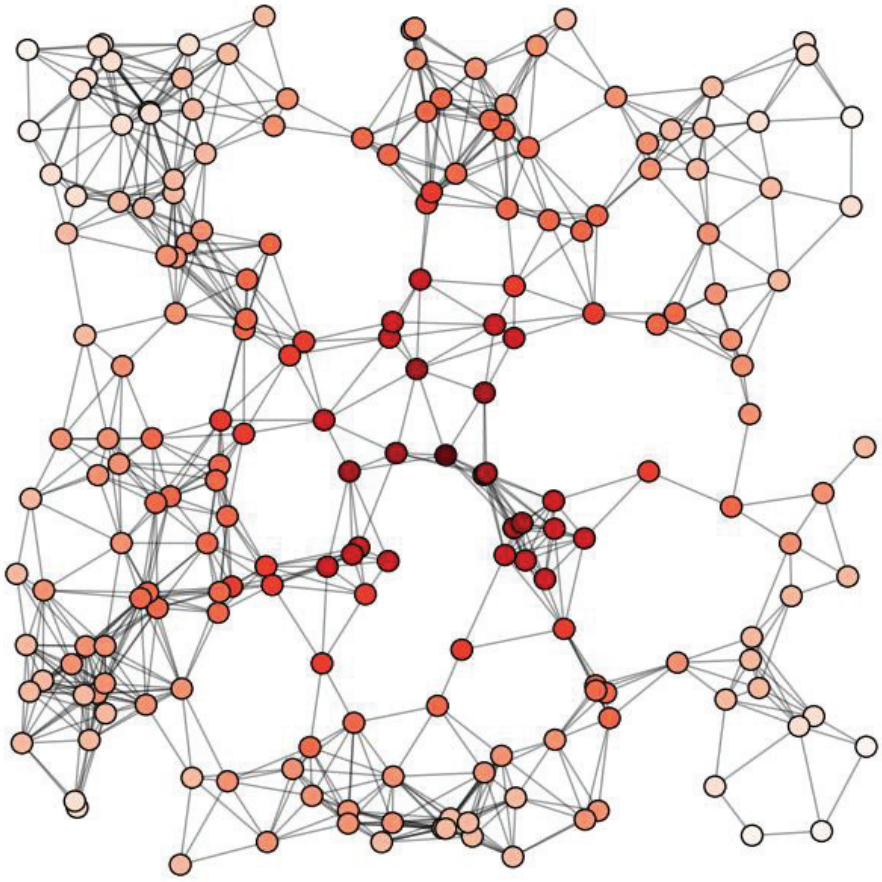


Figure 5.2. A random geometric graph $G(n = 200, r = 0.125)$ with 200 nodes distributed randomly on the 2-dimensional unit square. The nodes are colored based on their path length from the node near center.

in the graph, and this only increases the chance of having a monotone property. Further, when $r = \sqrt{d}$, the graph is a *complete graph* which satisfies every monotone property. A complete graph is a graph in which there is an edge between every pair of vertices. If A is a monotone property, then for $0 < \epsilon < 1$, let

$$r(n, \epsilon) = \inf\{r > 0 : \Pr\{G(n, r, f(u)) \text{ has property } A\} \geq \epsilon\}. \tag{5.1}$$

We define the threshold width of A as

$$\tau(A, \epsilon) = r(n, 1-\epsilon) - r(n, \epsilon), \tag{5.2}$$

when $0 < \epsilon < 1/2$. In [2], the authors show that all monotone graph properties have a sharp threshold for large random geometric graphs. In fact, the threshold width for random geometric graphs is much sharper than the one for Bernoulli random graphs. The threshold of property A is considered sharp if for every $\epsilon > 0$, we have

$$\tau(A, \epsilon) = o(\min\{r_c, 1 - r_c\}), \tag{5.3}$$

where r_c is the value of r such that

$$\Pr\{G(n, r_c, f(u)) \text{ has property } A\} = 1/2. \tag{5.4}$$

However, the goal of most of the previous studies is to address the asymptotic behavior of the threshold phenomena. In this section, we only consider finite (one-dimensional) random geometric graphs. In fact, random geometric graphs of higher dimensions are usually much more difficult to analyze. We believe that studying the threshold phenomena for finite one-dimensional networks could serve as a possible means of analyzing higher dimensions. While deriving similar results in a higher dimensional set-up might be difficult, the techniques used here may prove helpful in extending our results to higher dimensions.

In this section, we first state our result for the general case of $G(n, r, f(u))$ on a line. We will then conclude the special cases of random graphs with uniform and Poisson node distributions as well as the case of unreliable sensor grids. We now explain some notations and some definitions we need to state our results. The key idea in our analysis is to relate the behavior of monotone properties to the weight of the "bottleneck" matching (to be defined later) of the bipartite graph whose vertex sets are obtained by distributing n points independently on the line and according

to a distribution $f(u)$. Such a relation has been exploited in [2] to find an upper bound on the threshold width for random geometric graphs in the asymptotic case. Here, we describe the concept of bottleneck matching and its relation with monotone properties.

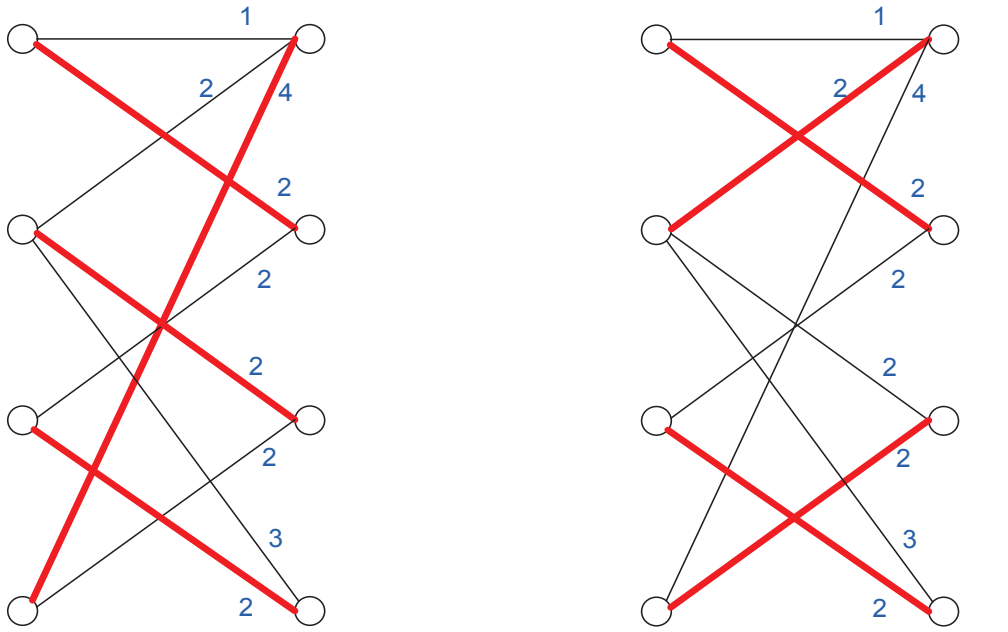
Recall that a bipartite graph is a graph whose vertices can be divided into two disjoint sets V_1 and V_2 such that every edge connects a vertex in V_1 to a vertex in V_2. In a bipartite graph with vertex sets V_1 and V_2, a *perfect matching* is a bijection (a one-to-one and onto mapping) $\phi : V_1 \to V_2$, such that each $v \in V_1$ is adjacent to $\phi(v) \in V_2$. Thus, a perfect matching is a disjoint collection of edges that covers every vertex. If the graph is weighted, then we define the weight of the matching as the maximum weight of any edge in the matching. Fig. 5.3(a) shows a possible realization of perfect matching and its corresponding weight in a bipartite graph. A *bottleneck matching* is a perfect matching with the minimum weight. Fig. 5.3(b) shows the bottleneck matching and its corresponding weight.

Let S_1 and S_2 denote two sets of n points each, where the points are i.i.d., chosen at random on the line according to $f(u)$. Form the complete bipartite graph on (S_1, S_2) and let the weight of an edge be the Euclidean distance between its endpoints. Let M_n denote the bottleneck matching weight of this graph. In [2], the authors linked the weight of the bottleneck matching with the threshold width of the monotone properties in a theorem which we repeat here.

Theorem 6. *If $Pr\{M_n > \gamma(n)\} \leq p$ for some function $\gamma(n)$ and some constant p, then $\tau(A, \sqrt{p})$ of any monotone property A is at most $2\gamma(n)$.*

According to this theorem, if we can find an upper bound on the probability $\Pr\{M_n > \gamma(n)\}$, then we can use it to find an upper bound on the threshold width. We first find the weight of the bottleneck matching for two sets of points on a line.

Lemma 7. *Let S_1 and S_2 be two sets of points each, where the points are chosen randomly and independently according to some arbitrary distribution. Let $\hat{S}_1 =$*



(a) Matching Weight =4 (b) Matching Weight =2

Figure 5.3. A weighted bipartite graph and realizations of (a) Perfect Matching, and (b) Bottleneck Matching. The matchings are shown by thick edges. The corresponding matching weights are also shown. Bottleneck matching is a perfect matching with the minimum weight.

$X_1, X_2, ..., X_n$ and $\hat{S}_2 = Y_1, Y_2, ..., Y_n$ be the points ordered according to their positions on the line, i.e. $X_1 < X_2 < ... < X_n$ and $Y_1 < Y_2 < ... < Y_n$. Then the bottleneck matching is the perfect matching $\phi : S_1 \to S_2$ such that $\phi(X_i) = Y_i$ for $i = 1, 2, ..., n$. Accordingly, the weight of the bottleneck matching is

$$M_n = \max_{i=1,...,n} |Y_i - X_i|. \tag{5.5}$$

Proof. We first show that in bottleneck matching X_1 is mapped to Y_1. So for now, assume that this is not the case and that X_1 and Y_1 have been mapped to Y_i and X_j, respectively, where $i, j \neq 1$. Without loss of generality, we assume that $X_1 < Y_1$. Then we will have 3 cases: 1. $X_1 < X_j < Y_1$, 2. $Y_1 < X_j < Y_i$, and 3. $Y_i < X_j$.

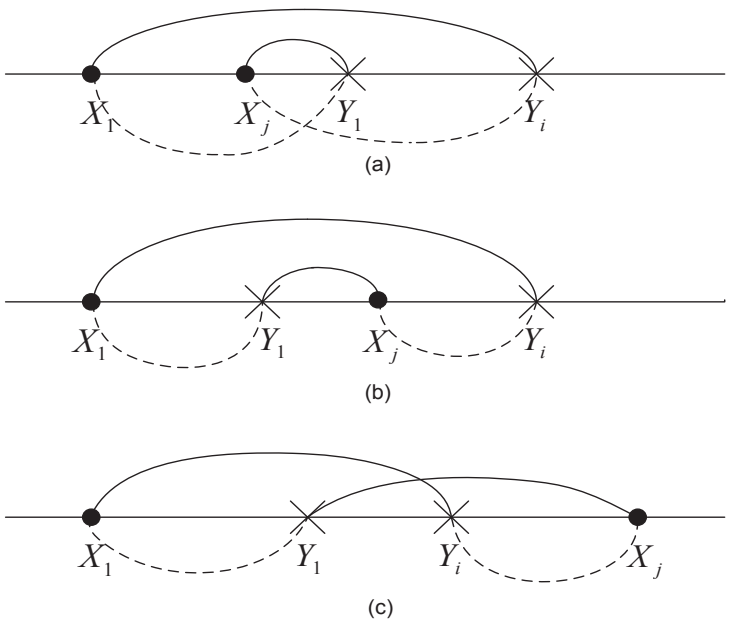


Figure 5.4. Different configurations of the two sets of random points considered to prove Lemma 7. Using dashed lines (i.e. mapping X_1 to Y_1) instead of solid lines can only lead to a perfect matching with lower weight. Hence, by mapping X_i to Y_i we will obtain the bottleneck matching.

These 3 cases are shown in Figure 5.4 where mappings between two points are shown by solid or dashed lines. It is easy to see in the figure that, in all the cases, if we map X_1 to Y_1 and X_j to Y_i (using dashed lines instead of solid lines) while not changing other mappings, we can only decrease the weight of matching. On the other hand, Y_i and X_j could be any two nodes in the graph. Therefore, by mapping X_1 to Y_1 we can get the bottleneck matching. Now, if we remove X_1 and Y_1 from our graph, we can use the same argument as above to prove that in the bottleneck matching, X_2 is mapped to Y_2, X_3 is mapped to Y_3 and so on. □

Now we need to find an upper bound for $\Pr\{\max_{i=1,\ldots n} |Y_i - X_i| > \gamma\}$ for every γ.

Theorem 7. *For the two sets of random points defined in Lemma 7 and for every* $\gamma > 0$*, we have*

$$Pr\{M_n > \gamma\} \leq \sum_{i=1}^{n} 2 \int_0^{\infty} f_i(u+\gamma) F_i(u) du, \tag{5.6}$$

where $f_i(u)$ and $F_i(u)$ are, respectively, the PDF and CDF of the ith order statistics of the underlying distribution $f(u)$.

Proof. Using Union bound, we have

$$\begin{aligned} Pr\{M_n > \gamma\} &= \Pr\{\max_{i=1,\ldots,n} |Y_i - X_i| > \gamma\} \\ &\leq \sum_{i=1}^{n} \Pr\{|Y_i - X_i| > \gamma\}. \end{aligned} \tag{5.7}$$

Note that X_i and Y_i are the ith order statistics of the underlying distribution. On the other hand, since X_i and Y_i are independent random variables, we know that

$$\Pr\{|Y_i - X_i| > \gamma\} = 2 \int_0^{\infty} f_i(u+\gamma) F_i(u) du,$$

which along with (5.7) gives (5.6). □

Note that Theorem 7 provides an upper bound for every monotone graph property and it is not limited to a specific property. An interesting point about the upper bound of Theorem 7 is that it holds for any two independent sets of random points that have the same size and the same distribution. Therefore, given the PDF and CDF of the order statistics of an arbitrary random variable, we can evaluate (5.6) for different values of γ and n, and hence, find the upper bound on the threshold width. In the case of $G(n, r)$ on the interval $[0, 1]$, the ith order statistics of the uniform distribution has a beta distribution with parameters i and $n - i + 1$ (see [104] chapter 7). Therefore, we have

$$\begin{aligned} f_i(u) &= i \binom{n}{i} u^{i-1} (1-u)^{n-i}, \\ F_i(u) &= I_u(i, n+1-i) = \sum_{j=i}^{n} \binom{n}{j} u^j (1-u)^{n-j}, \end{aligned} \tag{5.8}$$

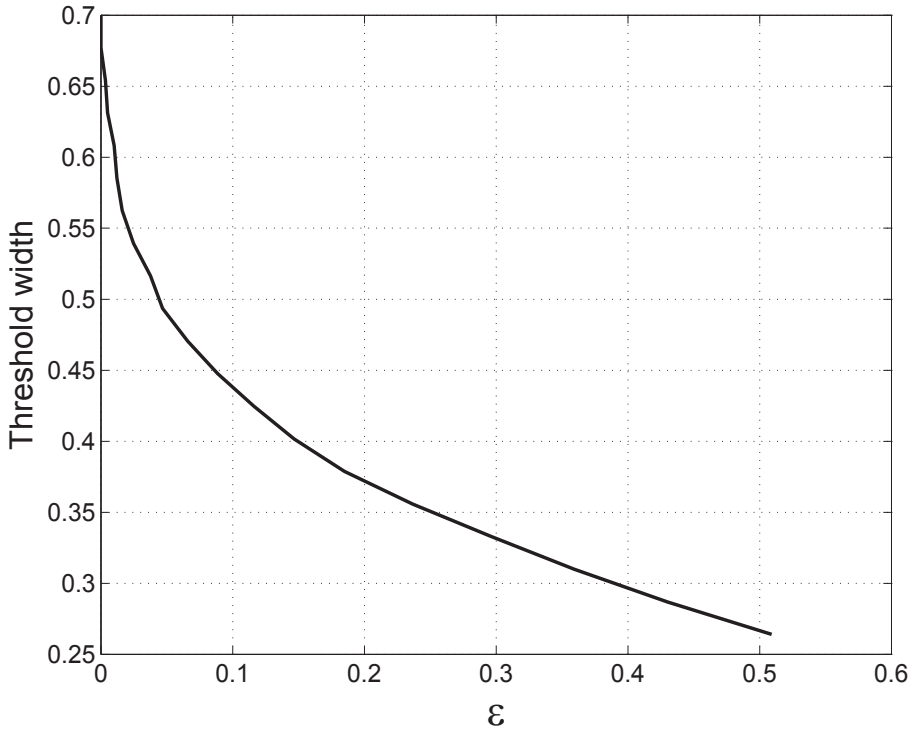


Figure 5.5. Upper bound of Theorem 7 on the threshold width of the monotone properties for $G(50, r)$ where 50 nodes are distributed uniformly at random on the unit line.

where $I_u(i, n+1-i)$ is the regularized incomplete beta function with parameters i and $n+1-i$. Substituting (5.8) in (5.6), we evaluated (5.6) to find the upper bound shown in Figure 5.5 for $G(50, r)$. Our bound is the first directly applicable to finite networks. However, comparing this bound against the actual value of the threshold width for some famous graph properties, we observed that the bound does not provide a tight approximation for them. In fact, it remains as an open problem to see whether there is any monotone property for which our bound is tight.

In [2], the authors show that for random geometric graphs with a uniform distribution of nodes we have

$$\tau(A, \epsilon) = \Theta\left(\sqrt{\frac{\log \epsilon^{-1}}{n}}\right), \tag{5.9}$$

which is *asymptotically* tight. However, this is an asymptotic result and without constant factors cannot be evaluated for specific values of n. In order to compare asymptotic results against the results from small-scale analysis, one needs to go through the underlying asymptotic analysis and extract the missing constant factors. However, due to the methods used in the asymptotic analysis, this is often hard if not impossible. We tried to minimize the upper bound obtained in [2], Lemma 5.1. While it seems impossible to find the exact value of the best constants, we derived them with a slight approximation. The bound obtained using these values is sketched in Figure 5.6 for a network with 50 nodes, along with the bound suggested by Theorem 7. As it can be seen, the asymptotic bound of [2] shows a poor performance in this case. This is mainly due to the algorithm whereby the bound is derived. We also observed that our bound outperforms the asymptotic bound even for large number of nodes such as $n = 2000$.

An asymptotic analysis of our bound is provided in Section 5.2.1, where it is shown that the bound can at least achieve

$$\tau(A, \epsilon) = O\left(\sqrt{\frac{\ln n + \ln \epsilon^{-1}}{n}}\right). \tag{5.10}$$

This is asymptotically close to the result of [2] saying that

$$\tau(A, \epsilon) = O\left(\sqrt{\frac{\log \epsilon^{-1}}{n}}\right). \tag{5.11}$$

In fact, when $\epsilon = O(\frac{1}{n^k})$ for any positive k, the two bounds will asymptotically be the same. For other values of ϵ, our bound is slightly worse due to the term $\ln n$ in the numerator. It seems that, the fact that our bound is slightly worse in the asymptotic regime is the price we pay to get a much better performance in the finite case.

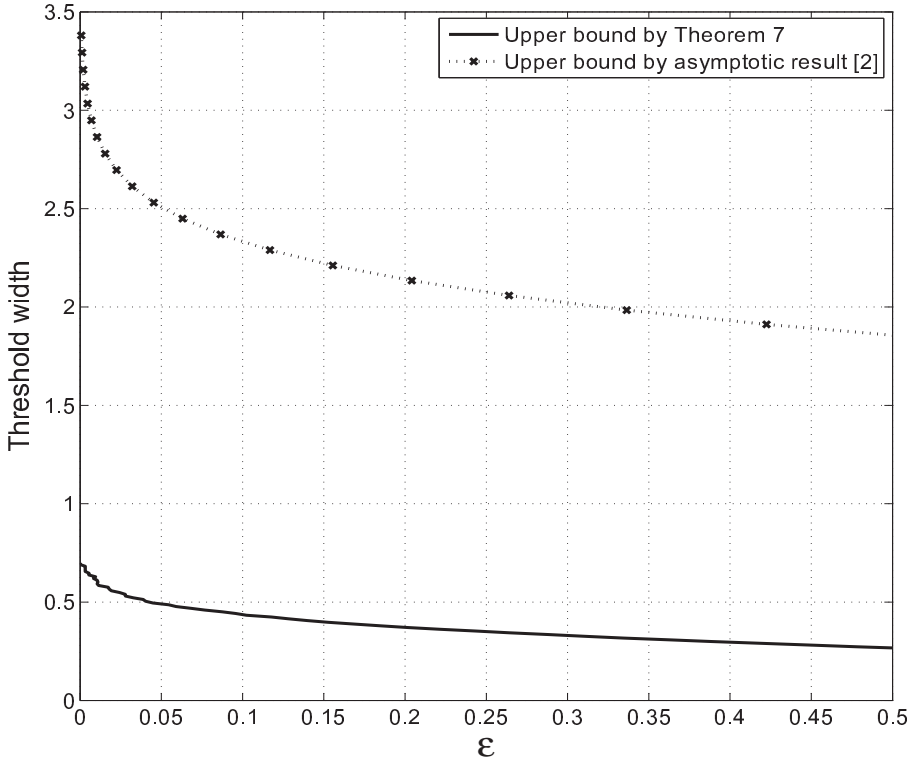


Figure 5.6. Comparing the upper bound on the threshold width implied by the asymptotic result of [2] against the upper bound obtained in this section for $G(50, r)$.

Now, suppose that the nodes in our line network are placed according to a Poisson point process with parameter λ. Then, for $i = 1, ..., n$ we can derive (see [104] chapter 7)

$$f_i(u) = \frac{\lambda^i u^{i-1} e^{-\lambda u}}{(i-1)!},$$

and

$$F_i(u) = 1 - \sum_{m=0}^{i-1} \frac{(\lambda u)^m e^{-\lambda u}}{(m-1)!}.$$

Using $f_i(u)$ and $F_i(u)$ as above, we can find the upper bound of the threshold width for the random geometric graphs generated by a Poisson point process.

Now, we consider an unreliable sensor grid on the unit interval which consists of n equidistant sensor nodes such that m of them are active. Note that all subsets of size m of the n nodes are equally probable to be active. For a given r, if the distance between two active nodes is less than r, there is a link between them. We can study the threshold phenomena for this unreliable grid when r ranges between 0 and 1. Note that as in [94], the probability curve would be piecewise constant for every graph property. Assuming that the grid nodes are located at the points $\frac{k}{n},\ k = 1, ..., n$, $f_i(u)$ can be derived as

$$f_i(u) = \sum_{k=i}^{n-(m-i)} \frac{\binom{k-1}{i-1}\binom{n-k}{m-i}}{\binom{n}{m}}\ \delta(u - \frac{k}{n}). \tag{5.12}$$

Substituting (5.12) in (5.6), we have found the upper bound shown in Figure 5.7 on $\tau(A, \epsilon)$ for $n = 100$ and $m = 35$.

5.2.1 A Note on the Asymptotic Case

Here, we perform an approximate asymptotic analysis of the bound in (5.6) when the nodes are uniformly distributed. We first find an upper bound for each term of the summation in (5.6). To find the integrals in each term, note that the beta distribution is known to be approximated by the normal distribution [105]. In fact, a beta distribution with parameters α_1 and α_2, $B(\alpha_1,\alpha_2)$, when α_1 and α_2 are both large enough (e.g. larger than 20), can be well approximated by

$$\mathcal{N}(\mu = \frac{\alpha_1}{\alpha_1 + \alpha_2}, \sigma = \sqrt{\frac{\alpha_1\alpha_2}{(\alpha_1 + \alpha_2)^2(\alpha_1 + \alpha_2 + 1)}}),$$

where μ and σ are the mean and the standard deviation of normal distribution, respectively. Thus, we consider two cases: first when either i or $n-i$ is small compared to n, e.g. less than 20, and second, when they both are large enough. In the first

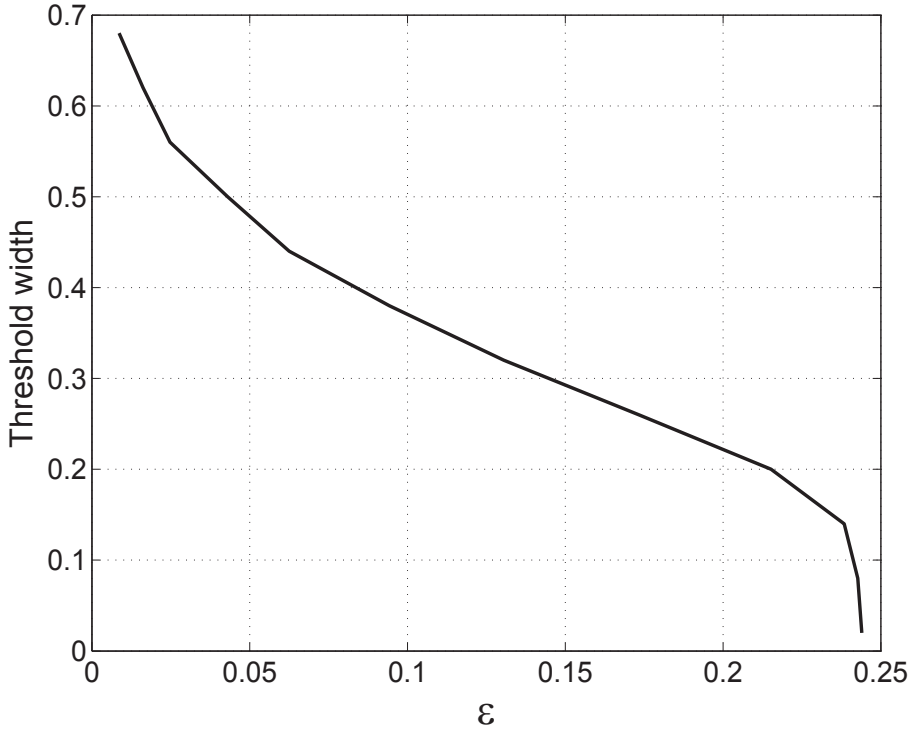


Figure 5.7. Upper bound on the threshold width of the monotone properties for a one-dimensional unreliable sensor grid with parameters $n = 100$, $m = 35$. A random subset of 35 nodes out of 100 equidistant sensor nodes are active.

case, note that in (5.8) we have $f_i(u) \leq c_1 n(1-u)^{n-1}$ for some constant c_1 and $F_i(u) \leq 1-(1-u)^n \leq 1$. Hence,

$$f_i(u+\gamma)F_i(u) < c_2\ n(1-u-\gamma)^{n-1}$$

for some constant $c_2 > 0$. As a result, we have

$$\int_0^{1-\gamma} f_i(u+\gamma)F_i(u)du < \int_0^{1-\gamma} c_2 n(1-u-\gamma)^{n-1} = c_2(1-\gamma)^n < c_2\ e^{-\gamma n}.$$

In the second case, using the approximation mentioned above, we will have

$$f_i(u) \approx \mathcal{N}(\mu_i = \frac{i}{n}, \sigma_i = \sqrt{\frac{i(n-i)}{n^3}}).$$

Note that σ_i varies between $\frac{1}{n}$ for small i's, and $\frac{1}{2\sqrt{n}}$ when $i = \frac{n}{2}$. Therefore $f_i(u)$ is a narrow normal distribution whose standard deviation is much smaller than the mean. Now assume that $\gamma > \frac{1}{\sqrt{n}}$. Then we have $\gamma > 2\sigma$ for all i's. Since the normal distribution takes on 95% of its probability within two standard deviations from the mean, we have

$$\int_0^{1-\gamma} f_i(u+\gamma)F_i(u)du < c_3\ F_i(\mu_i - \gamma)$$

for some constant c_3. Using bounds on the normal CDF as in [104], we get

$$F_i(\mu_i - \gamma) < \frac{\sigma_i}{\gamma}\frac{1}{\sqrt{2\pi}}\ e^{-\frac{\gamma^2}{2\sigma_i^2}}.$$

However, we have $\sigma_i^2 \leq \frac{1}{4n}$ which leads to

$$F_i(\mu_i - \gamma) < c_4\ e^{-2\gamma^2 n}$$

when c_4 is some positive constant. Finally we will have

$$\Pr\{M_n > \gamma\} \leq c\ n\ e^{-2\gamma^2 n}$$

for some constant c. This results in a bound on the threshold width as

$$\tau(A, \epsilon) = O(\sqrt{\frac{\ln n + \ln \epsilon^{-1}}{n}}).$$

5.3 MAC-Layer Capacity

In this section we study the MAC-layer capacity of finite networks in line deployments. As a model of the network, we consider the case of $G(n, r)$ on the interval $[0, 1]$, often called a *random interval graph* [102]. In our model of the MAC layer, if

transmissions along (s,t) and (s',t') are occurring simultaneously, then none of the edges (s,s'), (s,t'), (s',t), (t,t') should be present in the graph. The set of edges that can be so chosen is called a *D2-Matching* (Distance-2 Matching). The problem of finding a D2-matching of maximum cardinality is called D2EMIS [6]. In [6], it is shown that for a wide class of MAC protocols including IEEE 802.11, the MAC-layer capacity can be modeled as a maximum D2-matching (D2EMIS) problem in the underlying wireless network. The main result of [6] is that for a network with n nodes and communication radius r, the MAC-layer capacity is optimized at $r = \Theta(\frac{1}{\sqrt{n}})$ and is given by $\Theta(n)$. Although this is an important and valuable result, it is not as precise when we consider finite networks. For example, suppose we have a network consisting of 100 sensors and we want to choose the communication radius such that the MAC-layer capacity is optimized. The asymptotic result does not tell us what the optimum MAC-layer capacity and its corresponding communication radius are. In this section, we analyze the average MAC-layer capacity for finite line networks. Here, we define $MAC(n,r)$ as the average, over all configurations of nodes, of the cardinality of the D2EMIS for a random interval graph $G(n,r)$. Note that $MAC(n,r)$ is the average value of the maximum size of the D2-matching on $G(n,r)$. We first provide analytical lower and upper bounds on $MAC(n,r)$. Then, we propose an algorithm to find the exact value of the size of the D2EMIS for any arbitrary node configuration. Using this algorithm, we compare our bounds to the actual value of the capacity. At the end, we will consider a slightly different model of the MAC, called the "Directed Model", and extend our results to this model.

5.3.1 Lower Bound on the MAC-Layer Capacity

In this section we introduce a lower bound on the MAC-layer capacity which is a combination of two different bounds. First, recall that a connected component of size k of a graph G is a maximal connected subgraph of G with k vertices. For a random

interval graph $G(n,r)$, let us denote the number of connected components of size k by C_n^k and the total number of the connected components by C_n.

Theorem 8. *For a line network modeled by a random interval graph $G(n,r)$ we have*

$$MAC(n,r) \geq 1 + (n-3)(1-r)^n - (n-2)(1-2r)^n. \tag{5.13}$$

Proof. The proof is based on the number of connected components in the network's graph. Since transmissions in different components do not conflict, every connected component of size greater than one can contribute at least one transmission to $MAC(n,r)$. Therefore, the average number of the concurrent transmissions is always larger than the average number of the connected components of size greater than one. The average number of the total connected components and the average number of the isolated vertices (connected components of size one) for a random interval graph are calculated in [106], Theorems 1 and 4. Using this, we have

$$\begin{aligned} MAC(n,r) \geq E[C_n] - E[C_n^1] &= 1 + (n-1)(1-r)^n \\ &\quad - (n-2)(1-2r)^n - 2(1-r)^n \\ &= 1 + (n-3)(1-r)^n - (n-2)(1-2r)^n. \end{aligned}$$

□

Note that it is easy to check that the bound in (5.13) is asymptotically maximized at $r = \Theta(\frac{1}{n})$ resulting in a maximum value of $\Theta(n)$ for the lower bound on the MAC-layer capacity. Regarding [6], this asserts that our bound is asymptotically tight. It is important to note that $r = \Theta(\frac{1}{\sqrt{n}})$ in [6] is replaced by $r = \Theta(\frac{1}{n})$ here due to the one-dimensional nature of our problem. Now, we prove the following lemma which leads us to a different lower bound on $MAC(n,r)$.

Lemma 8. *Given a line network modeled by $G(n,r)$ and an interval I of length l on the line, let $P(l)$ be the probability of having at least one link in I. Then*

$$P(l)=\begin{cases} 1-(1-l)^n-nl(1-l)^{n-1} & if\, l\le r, \\ 1-\sum_{k=0}^{\min(\lceil \frac{l}{r}\rceil,n)}\binom{n}{k}[l-(k-1)r]^k(1-l)^{n-k} & if\, l>r. \end{cases} \tag{5.14}$$

Proof. If $l \le r$ then $P(l)$ is equal to the probability of having at least 2 nodes in I, which is $1-(1-l)^n-nl(1-l)^{n-1}$ for n nodes distributed uniformly and independently on $[0,1]$. In the case of $l > r$, we find the probability of having no link in an interval of length l which we denote by $P_{nl}(l)$. Then we will have $P(l) = 1-P_{nl}(l)$. For $P_{nl}(l)$, we have

$$\begin{aligned} P_{nl}(l) &= \sum_{k=0}^{\min(\lceil \frac{l}{r}\rceil,n)} \Pr\{\text{no link in } I|\ k \text{ nodes in } I\} \times \Pr\{k \text{ nodes in } I\} \\ &= \sum_{k=0}^{\min(\lceil \frac{l}{r}\rceil,n)} \Pr\{\text{no link in } I|\ k \text{ nodes in } I\} \times \binom{n}{k}\ l^k\ (1-l)^{n-k}. \end{aligned} \tag{5.15}$$

Therefore, it suffices to find the probability of having no link in I given that there are k nodes in it. For $k = 0$ and 1, this probability is trivially 1. It is easy to verify that given that $k \ge 2$ nodes are in the arbitrary interval $I = [x_i, x_i + l]$, they are distributed independently and uniformly on I. We need to find the probability of the event that these k nodes have spacings larger than r. To achieve this, we define two sets whose ratio of their volumes is the sought probability. The first set is the set of all configurations of k points in I whose volume is l^k. The other one is the set of all configurations of the k points in I, $k \le \lceil \frac{l}{r} \rceil$, for which the spacings between the points are all larger than r. This is, in fact, equivalent to the set of k points

drawn uniformly and independently from a subinterval of length $l-(k-1)r$ of I. The volume of this set is $(l-(k-1)r)^k$. Hence, substituting

$$\frac{(l-(k-1)r)^k}{l^k}$$

as

$$\Pr\{\text{no link in } I \mid k \text{ nodes in } I\}$$

in (5.15), we will find $P_{nl}(l)$ which leads us to $P(l)$ in (5.14). □

Theorem 9. *For a line network modeled by $G(n,r)$, define $P(l)$ as above and $m(l) = \lfloor \frac{1}{l+r} \rfloor$ for $0 < l < 1$. Then*

$$MAC(n,r) \geq \max_{l\in[0,1]} \{m(l)P(l) + P(1-m(l)(l+r))\}. \tag{5.16}$$

Proof. The proof is based on a constructive algorithm which finds a number of possible concurrent transmissions on the unit-length line. Consider the intervals of length l in Figure 5.8 which are a distance r apart. We have $m(l) = \lfloor \frac{1}{l+r} \rfloor$ of these intervals which are denoted by $I_1, I_2, ..., I_{m(l)}$. Also, there may be an interval of length $1-m(l)(l+r)$ at the end of the line which we denote by $I_{m(l)+1}$. Note that all these intervals do not necessarily contain an edge. However, the edges contained in $I_1, I_2, ..., I_{m(l)+1}$ are at least a distance r apart and can be in the D2-matching. Therefore, the average number of the concurrent transmissions obtained in this way is equal to the average number of the intervals containing at least one edge. Let X be the number of such intervals. To find $E[X]$, we assign an indicator random variable X_i to each interval

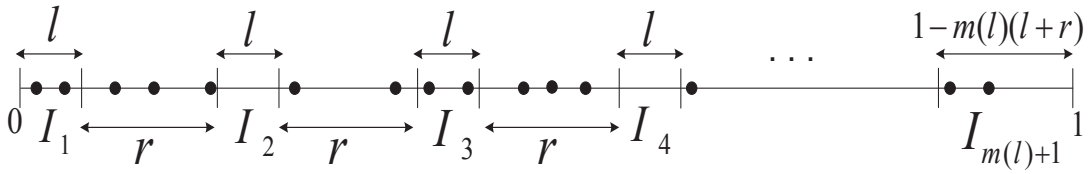


Figure 5.8. Intervals corresponding to the constructive lower bound on MAC-layer capacity. Note that in the figure above we have $X_1 = 1$, $X_2 = 0$, $X_3 = 1$, $X_4 = 0$, and $X_{m(l)+1} = 1$. Since the intervals are a distance r apart, the average number of the concurrent transmissions obtained in this setting is equal to the average number of the intervals containing at least one edge.

I_i which is one if there exists at least one edge in that interval and is zero otherwise. Then, we have

$$X = \sum_{i=1}^{m(l)+1} X_i \quad \text{and} \quad E[X] = \sum_{i=1}^{m(l)+1} E[X_i].$$

But according to Lemma 8,

$$E[X_i] = \Pr\{X_i = 1\} = P(l) \quad \text{for} \quad i = 1, 2, ..., m(l),$$

and

$$E[X_{m+1}] = \Pr\{X_{m+1} = 1\} = P(1 - m(l)(l + r)).$$

We can maximize $E[X]$ over l which gives us (5.16). □

A lower bound on $MAC(n, r)$ can be obtained from maximum of the lower bounds given by Theorems 8 and 9.

5.3.2 Upper Bound on the MAC-Layer Capacity

In this section the upper bound on the MAC-layer capacity is addressed via a theorem which results from a combination of two bounds.

Theorem 10. *For a line network modeled by $G(n,r)$ we have*

$$MAC(n,r) \leq \min(\sum_{k=1}^{n} E[C_n^k] \times \lceil \frac{k-1}{3} \rceil, \lceil \frac{1}{r} \rceil),$$

where

$$E[C_n^k] = \sum_{j=0}^{1} \binom{n-k-1}{1-j} \binom{2}{j} \sum_{i=0}^{k-1} \binom{k-1}{i} (-1)^i \times \Big(1-(2-j+i)r\Big)_+^n \qquad (5.17)$$

with $a_+ = a$ for positive a and $a_+ = 0$ otherwise.

Proof. Consider a connected component of size 2. This component contributes one transmission to $MAC(n,r)$. Now consider components of size 3 and 4. According to the definition of the set of edges in a D2-matching, these components also contribute at most 1 edge to the D2EMIS. In fact, any edge chosen for D2-matching precludes at least two other edges from participating in the matching. Therefore, a component of size k can support at most $\lceil \frac{k-1}{3} \rceil$ concurrent transmissions. Thus, the average number of the concurrent transmissions is smaller than the sum of the average number of the connected components of size k times $\lceil \frac{k-1}{3} \rceil$. The average number of the connected components of size k in a random interval graph is given in [106] as (5.17).

On the other hand, every transmission covers at least an interval r of the line. That is, if we pick an edge as a member of the D2EMIS, we can not pick any other edge for D2EMIS in a distance less than r from the first one. Therefore, there can not be more than $\lceil \frac{1}{r} \rceil$ concurrent transmissions. This completes the proof. □

Again, it can be seen that when $r = \Theta(\frac{1}{n})$, $E[C_n^2]$ and $\lceil \frac{1}{r} \rceil$ are both $\Theta(n)$ asserting the asymptotic tightness of the bound.

5.3.3 Algorithm for the Exact Value of the MAC-Layer Capacity

As we mentioned earlier, transmissions in different connected components do not conflict. Therefore, to find the D2EMIS, it suffices to give an algorithm for finding

the maximum possible number of concurrent transmissions in every component of size greater than one. We now propose an algorithm to find the maximum number of concurrent transmissions in a connected component. Assume that the first vertex of the component, v_i, is at location x_i and the second vertex, v_{i+1}, is located at x_{i+1}, as it is shown in Figure 5.9. We choose the first edge of the component, connecting v_i to v_{i+1}, to participate in D2EMIS. Thus, none of the vertices in the interval $(x_{i+1}, x_{i+1}+r]$ can participate in the D2EMIS. We call the interval $[x_i, x_{i+1} + r]$ an *interference interval*. Then, we consider the first vertex located after $x_{i+1} + r$ which is v_l, and choose the edge (v_l, v_{l+1}) as another member of the D2EMIS. Again, none of the vertices within range r of v_{l+1} can participate in D2EMIS. We repeat this until we reach the end of the component. It is easy to see that this greedy choice is optimal. Consider an optimal algorithm which does not choose the first edge of the component, hence leads to an edge in D2EMIS in an interference interval larger than $[x_i, x_{i+1}+r]$. Assume that the last vertex of the component is v_p and is located at x_p. So, this algorithm has to choose the second edge of the D2EMIS from an interval shorter than $(x_{i+1} + r, x_p]$. Note that our algorithm chooses the first edge in $(x_{i+1} + r, x_p]$ as the second edge of the D2EMIS. In fact, moving toward the end of the component, our algorithm always selects its next edge for the D2EMIS from an interval at least as large as the one for the optimal algorithm. Therefore, the optimal algorithm cannot find a larger D2-matching than our algorithm.

It can be seen that it takes a linear time, with respect to the number of nodes, to find the connected components on a line and then the algorithm above takes a linear time to find the D2EMIS. This result can be interesting since the problem of finding D2EMIS is NP-complete in the two-dimensional case [99]. Also, note that to find $MAC(n, r)$, we need to find the average size of the D2-matching obtained by the above algorithm. However, analyzing this algorithm to find the exact value of $MAC(n, r)$ might be difficult. Figure 5.10 shows the exact value of $MAC(50, r)$ resulted by

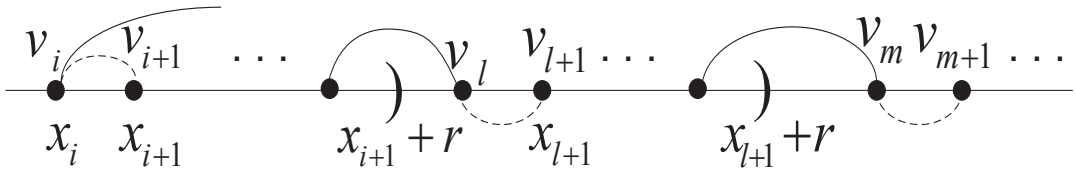


Figure 5.9. Algorithm to find the exact value of the MAC-layer capacity by finding the maximum number of concurrent transmissions in a connected component. Dashed lines are the edges chosen by the algorithm as members of the D2EMIS. We choose the first edge of the component to participate in D2EMIS. After that, other participating edges are chosen in a greedy approach.

exhaustive simulations and using the proposed algorithm, compared against the lower and upper bounds given by Theorems 8, 9 and 10. The "Directed Model", sketched by a dotted line in the figure, will be explained in the next section. Note that as r increases to 1, the capacity goes to one since the graph will become a complete graph which can support only one transmission. On the other hand, there will not be any edges in the graph when $r = 0$, hence, the capacity will be zero. As you can see in Figure 5.10, our bounds are able to predict the radius at which the capacity is maximum with a good precision. Note that this maximum occurs in a range of communication radii within which the graph is mostly formed by size-2 and size-3 connected components. This is also predicted by Theorems 8, 9 and 10. One can appropriately use this information to choose an efficient communication radius for different realizations of line networks.

5.3.4 Directed Model for MAC-Layer

In the previous sections, we studied a model of MAC-layer that can handle the undirected links in wireless networks. However, one may model the network with a directed graph in order to consider a specific network property or application. As a result, if transmissions along (s, t) and (s', t') are occurring simultaneously, then either (s, s') or (t, t') can be present in the graph. In fact, the transmitter nodes (or receiver nodes) can reside in each other's interference range but not cause destructive conflict

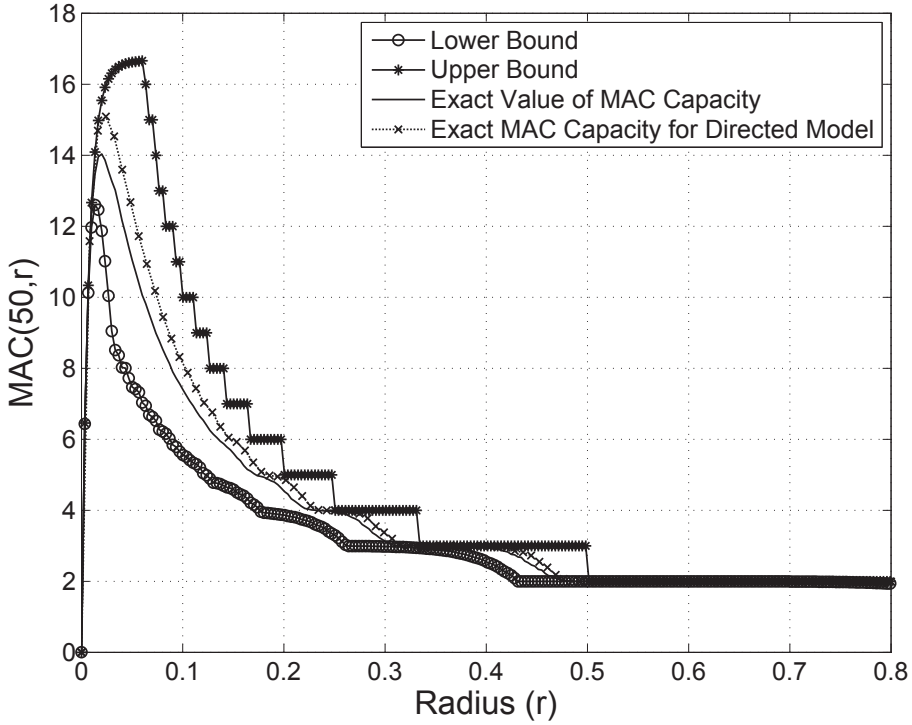


Figure 5.10. Actual value of $MAC(50, r)$ along with the lower and upper bounds of Theorems 8, 9 and 10. The exact MAC-layer capacity for the directed model has been also shown. Using the two bounds, one can predict, with a good precision, the radius at which the capacity is maximum.

in each other's data transmission sessions. We refer to this model as the "directed model" as opposed to the model studied earlier in this section which we refer to as the "undirected model". In this model, edges (x_1, y_1) and (x_2, y_2) can be in the D2EIMS if x_1 and x_2, and also y_1 and y_2 are more than a distance r apart. First, note that clearly the lower bound derived in Section 5.3.1 holds for this model as well, as the previous (undirected) model holds more restrictive constraints. Moreover, a similar upper bound as in Section 5.3.2 can be easily obtained for this model. However, we observed that this does not yield a really tight bound for the directed case. Similar to the undirected case, we can derive an algorithm to find the exact MAC-layer capacity for this relaxed model. Here is the algorithm running over a connected component (we denote a node and its coordinate by the same symbol):

1. Find the first and second nodes (from the left) of a connected component which are less than r apart, and call them x_1 and y_1, respectively. Let the edge (x_1, y_1) be in the D2EMIS.

2. Take the first node after $y_1 + r$, say y_2, and consider the interval $I_1 = [x_1 + r, y_1 + r]$:

   - If there exists a node in I_1 closer than r to y_2, choose the first such node in I_1, say x_2, and let the edge (x_2, y_2) be in the D2EMIS. Go to step 2 assuming that x_1, y_1 are replaced by x_2, y_2.
   - If there does not exist any node in I_1 closer than r to y_2, go to step 1 assuming that y_2 is the first node of the component.

The optimality of this algorithm can be proved along the same lines of the proof provided for the undirected model in Section 5.3.3. This algorithm finds the D2EMIS for a specific value of r and a specific node placement. To find the average MAC-layer capacity for each value of r, one needs to perform exhaustive simulations to account for different node placements according to the uniform distribution. Using the above algorithm, we sketched the exact value of the capacity achieved for this relaxed model in Figure 5.10, comparing it against the capacity obtained for the undirected model. As can be seen in the figure, these two are relatively close together such that the upper bound for the undirected model can also be used for the new model as an estimate. This suggests that the two models show a close behavior such that the bounds for the undirected model could serve as estimates for the directed case as well.

5.4 Chapter Summary

In this chapter, we studied the threshold phenomena and MAC-layer capacity in finite wireless networks on a line. We considered random geometric graphs as a model for wireless networks which is used extensively in the literature. We derived an upper

bound for the threshold width of such finite networks which holds for every monotonic graph property. We also studied the problem of MAC-layer capacity for finite line networks. MAC-layer capacity is an example of non-monotonic characteristics of networks. We provided an algorithm for finding its exact value and also derived lower and upper bounds. Through simulations, we verified that our bounds can give quite a good estimate of the actual value of the MAC-layer capacity.

CHAPTER 6

SUMMARY AND CONCLUSIONS

The driving force behind this dissertation was the importance of studying practical communication networks with real-world restrictions. As two fundamental issues we took on finite-length block codes and finite wireless networks. Our analysis opens up many theoretical and practical research possibilities that offer potential for further research. The results from our analysis of polar codes can be employed to improve the finite-length performance by making better choices for information bits. Our stopping set analysis can be extended to find the distribution of stopping sets and their size. The results can then be used to bound the bit or block erasure probability, or to find the exact values if possible.

A natural extension would be to conduct a similar analysis for the case of the Gaussian channel. In this case, the stopping set analysis needs to be expanded to "trapping set analysis". Trapping sets are the main cause of the decoding failure over the Gaussian channel. Trapping sets maintain a more general definition than stopping sets, and are formed in a variety of shapes. This makes it much harder to tackle the finite-length analysis for the Gaussian channel.

In this dissertation, we tackled the problem of rate-compatible polar codes, and briefly touched the non-uniform and UEP polar codes. The truth however is that polar coding over parallel channels is a very important open problem. The closeness of the setting for parallel channels to the one in rate-compatible polar codes makes it possible to apply different rate-compatible techniques to derive non-uniform coding schemes. While we only went over stopping-tree puncturing, the results from the

finite analysis can be employed to derive more complicated schemes for this purpose. We also pointed out the potential unequal-error-protection capabilities of polar codes. However, designing UEP codes based on polar codes which yield more than two orders of magnitude in error rate spread is an interesting practical problem.

On the finite wireless network, there is no need to justify the importance of pursuing the paths taken in this dissertation. The threshold phenomena provides us with a unified framework to approach the analysis of network properties. Such an analysis would also be mathematically wonderful and theoretically rich. An immediate research direction is to extend the results to two or higher dimensional networks. We can expect to face challenging problems in this case. On the other hand, studying each of the network properties- such as connectivity, coverage, capacity, etc- is still an open problem for many practical models of wireless networks.

BIBLIOGRAPHY

[1] E. Arikan. Channel polarization: A method for constructing capacity-achieving codes for symmetric binary-input memoryless channels. *IEEE Trans. Inf. Theory*, 55(7):3051–3073, July 2009.

[2] Ashish Goel, Sanatan Rai, and Bhaskar Krishnamachari. Monotone properties of random geometric graphs have sharp thresholds. *Annals of Applied Probability*, 15:2535–2552, 2005.

[3] H. Pishro-Nik, N. Rahnavard, and F. Fekri. Nonuniform error correction using low-density parity check codes. in Proc. of Fortieth Annual Allerton Conference, Urbana-Champaign, IL, Oct. 2002.

[4] H. Pishro-Nik, N. Rahnavard, J. Ha, F. Fekri, and A. Adibi. Low-density parity-check codes for volume holographic memory systems. *Appl. Opt.*, 42:861–870, 2003.

[5] Shu Lin and Daniel J. Costello. *Error Control Coding: Fundamentals and Applications*. Prentice-Hall, 1983.

[6] H. Balakrishan, C. L. Barrett, V. S. Anil Kumar, M. V. Marathe, and S. Thite. The distance-2 matching problem and its relationship to MAC-layer capacity of ad hoc wireless networks. *IEEE J. Select. Areas Commun.*, 22:1069 –1079, 2004.

[7] M. Grossglauser and D. Tse. Mobility increases the capacity of ad-hoc wireless networks. *IEEE Infocom, Anchorage, Alaska, USA*, April 2001.

[8] P. Gupta and P. R. Kumar. The capacity of wireless networks. *IEEE Trans. Inform. Theory*, 46(2):388–404, 2000.

[9] P. Gupta and P. R. Kumar. Towards an information theory of large networks: an achievable rate region. *IEEE Trans. Inform. Theory*, 49:1877–1894, 2003.

[10] Jinyang Li, Charles Blake, Douglas S. J. De Couto, Hu Imm Lee, and Robert Morris. Capacity of ad hoc wireless networks. In *Proceedings of the 7th ACM International Conference on Mobile Computing and Networking*, pages 61–69, Rome, Italy, July 2001.

[11] B. Liu, Z. Liu, and D. Towsley. On the capacity of hybrid wireless networks. *IEEE Infocom, San Francisco, CA, USA*, 2003.

[12] E. Perevalov and R. Blum. Delay limited capacity of ad hoc networks: Asymptotically optimal transmission and relaying strategy. *IEEE Infocom, San Francisco, CA, USA*, 2003.

[13] M. Ben-Or and N. Linial. Collective coin flipping, robust voting games, and minima of banzhaf value. In *IEEE Symposium on Foundation of Computer Science*, pages 408–416, march 1985.

[14] M. Ben-Or and N. Linial. *Randomness and Computation*. New York, NY: Academic Press, 1990.

[15] B. Bollobas and A.Thomason. Threshold functions. *Combinatorica 7*, pages 35–38, 1987.

[16] J. Bourgain and G. Kalai. Influences of variables and threshold intervals under group symmetries. *Geom. Funct. Anal.*, pages 438–461, 1997.

[17] E. Friedgut. Sharp thresholds of graphs properties, and the k-sat problem. *Journal of American Mathematical Society 12*, pages 1017–1054, 1999.

[18] J. Bourgain. Appendix on sharp thresholds of monotone properties. *Journal of American Mathematical Society 12*, pages 438–461, 1999.

[19] J.T. Chayes, L. Chayes, D. S. Fisher, and T. Spencer. Finite-size scaling and correlation length for disordered systems. *Physical Letters Review 57*, pages 2999–3002, 1986.

[20] E. Friedgut and G.Kalai. Every monotone graph property has a sharp threshold. In *Proceedings of American Mathematical Society*, pages 2993–3002, march 1996.

[21] G. Grimmett. *Percolation*. Berlin, Springer-Verlag, 1989.

[22] J. Kahn, G. Kalai, and N. Linial. The influence of variables on boolean functions. In *Proceedings of 29-th Annual Symposium on Foundations of Computer Science*, pages 68–80, march 1988.

[23] C. E. Shannon. A mathematical theory of communication. *SIGMOBILE Mob. Comput. Commun. Rev.*, 5(1):3–55, January 2001.

[24] P. Elias. Coding for noisy channels. *IRE International Convention Record*.

[25] R. G. Gallager. *Information Theory and Reliable Communication*. Wiley, New York, 1968.

[26] D. J. Costello Jr. and G. D. Forney Jr. Channel coding: The road to channel capacity. *Proceedings of the IEEE*, 95(6), June 2007.

[27] P. Elias. Error-free coding. *IEEE Trans. Inform. Theory*, 4.

[28] Jr. G. D. Forney. *Concatenated Codes*. MIT Press, 1966.

[29] A. J. Viterbi. Error bounds of convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Trans. Inform. Theory*, 13(2):260–269, Apr. 1967.

[30] F. Jelinek J. Raviv L. Bahl, J. Cocke. Optimal decoding of linear codes for minimizing symbol error rate. *IEEE Trans. Inform. Theory*, 20(2):284–287, Mar. 1974.

[31] R. M. Fano. A heuristic discussion of probabilistic decoding. *IEEE Trans. Inform. Theory*, 9(2):64–74, Apr. 1963.

[32] R. G. Gallager. *Low-Density Parity-Check Codes*. MIT Press, Cambridge, MA,USA, 1963.

[33] P. Thitimajshima C. Berrou, A. Glavieux. Near shannon limit error-correcting coding and decoding. In *IEEE ICC*, pages 1064–1070, Geneve, Switzerland, May 1993.

[34] R. Kotter N. Wiberg, H.-A. Loeliger. Codes and iterative decoding on general graphs. *European Transactions on Telecommuncations*, 6.

[35] M.G. Luby, M. Mitzenmacher, M.A Shokrollahi, and D.A. Spielman. Efficient erasure correcting codes. *IEEE Trans. Inf. Theory*, 47:569–584, 2001.

[36] T. J. Richardson and R. L. Urbanke. The capacity of low-density parity-check codes under message-passing decoding. *IEEE Trans. Inf. Theory*, 47:599–618, 2001.

[37] Jr. T. Richardson R. Urbanke S.-Y. Chung, G. D. Forney. On the design of low-density parity-check codes within 0.0045 db of the shannon limit. *IEEE Communications Letters*, 5(2):58–60, Feb. 2001.

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

[39] N. Hussami, S. Korada, and R. Urbanke. Performance of polar codes for channel and source coding. In *IEEE International Sympousiom on Information Theory (ISIT)*, 2009.

[40] Ryuhei Mori and Toshiyuki Tanaka. Non-binary polar codes using reed-solomon codes and algebraic geometry codes. In *IEEE Information Theory Workshop (ITW)*, August 2010.

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

[42] Mathis Seidl and Johannes B. Huber. Improving successive cancellation decoding of polar codes by usage of inner block codes. In *6th IEEE International Symposium on Turbo Codes and Iterative Information Processing*, 2010.

[43] Amin Alamdar-Yazdi and Frank R. Kschischang. A simplified successive-cancellation decoder for polar codes. *IEEE Communications Letters*, 15(12):1378 – 1380, Dec 2011.

[44] N. Goela, S.B. Korada, and M. Gastpar. On LP decoding of polar codes. In *IEEE Information Theory Workshop (ITW)*, August 2010.

[45] R. Pedarsani, H. Hassani, I. Tal, and E. Telatar. On the construction of polar codes. In *IEEE International Symposium on Information Theory (ISIT)*, 2011.

[46] I. Tal and A. Vardy. How to construct polar codes. arXiv:1105.6164v1 [cs.IT], May 2011.

[47] A. Eslami and H. Pishro-Nik. On bit error rate performance of polar codes in finite regime. In *48th Annual Allerton Conference on Communication, Control, and Computing*, August 2010.

[48] A. Eslami and H. Pishro-Nik. A practical approach to polar codes. In *IEEE International Symposium on Information Theory (ISIT)*, August 2011.

[49] Gregory Bonik, Sergei Goreinov, and Nickolai Zamarashkin. A variant of list plus crc concatenated polar code. arXiv:1207.4661v1 [cs.IT], Jul 2012.

[50] Bin Li, Hui Shen, and David Tse. An adaptive successive cancellation list decoder for polar codes with cyclic redundancy check. arXiv:1208.3091 [cs.IT], Aug 2012.

[51] Kai Niu and Kai Chen. Crc-aided decoding of polar codes. *IEEE Communications Letters*, pp(99).

[52] S.H. Hassani and R. Urbanke. On the scaling of polar codes: I.the behavior of polarized channels. In *IEEE International Symposium on Information Theory (ISIT)*, June 2010.

[53] S.H. Hassani, K. Alishahi, and R. Urbanke. On the scaling of polar codes: Ii.the behavior of un-polarized channels. In *IEEE International Symposium on Information Theory (ISIT)*, June 2010.

[54] S.B. Korada, A. Montanari, E. Telatar, and R. Urbanke. An emprical scaling law for polar codes. In *IEEE International Symposium on Information Theory (ISIT)*, June 2010.

[55] Ali Goli, S. Hamed Hassani, and R. Urbanke. Universal bounds on the scaling behavior of polar codes. In *IEEE International Symposium on Information Theory (ISIT)*, July 2012.

[56] S. H. Hassani, Ryuhei Mori, Toshiyuki Tanaka, , and R. Urbanke. Rate-dependent analysis of the asymptotic behavior of channel polarization. http://arxiv.org/abs/1110.0194v2, Oct 2011.

[57] C. Di, D. Proietti, I. E. Telatar, T.J. Richardson, and R.L. Urbanke. Finite-length analysis of low-density parity-check codes on the binary erasure channel. *IEEE Trans. Inf. Theory*, 48:1570 –1579, 2002.

[58] H. Pishro-Nik and F. Fekri. On decoding of low-density parity-check codes on the binary erasure channel. *IEEE Trans. Inf. Theory*, 50:439–454, 2004.

[59] H. Pishro-Nik and F. Fekri. Results on punctured low-density parity-check codes and improved iterative decoding techniques. *IEEE Trans. Inf. Theory*, 53(2):599–614, February 2007.

[60] A. Eslami and H. Pishro-Nik. On finite-length performance of polar codes: Stopping sets, error floor, and concatenated design. *IEEE Transactions on Communications, Accepted*, August 2012.

[61] C. Di, T. J. Richardson, and R. L. Urbanke. Weight distribution of low-density parity-check codes. *IEEE Trans. Inf. Theory*, 52(11):4839 – 4855, 2006.

[62] A. Orlitsky, K. Viswanathan, and J. Zhang. Stopping set distribution of ldpc code ensembles. *IEEE Trans. Inf. Theory*, 51(3):929 – 953, 2005.

[63] A. Orlitsky, R. Urbanke, K.Viswanathan, and J. Zhang. Stopping sets and the girth of tanner graphs. In *IEEE International Symposium on Information Theory (ISIT)*, June 2002.

[64] F. R. Kschischang, B. J. Frey, and H. A. Loeliger. Factor graphs and the sum-product algorithm. *IEEE Transactions on Information Theory*, 47(2):498 – 519, Feb. 2001.

[65] M. Gholami and M. Esmaeili. Maximum-girth cylinder-type block-circulant ldpc codes. *IEEE Transactions on Communications*, 60(4):952 – 962, April 2012.

[66] Irina E. Bocharova, Florian Hug, Rolf Johannesson, Boris D. Kudryashov, and Roman V. Satyukov. Searching for voltage graph-based ldpc tailbiting codes with large girth. *IEEE Transactions on Information Theory*, 58(4):2265 – 2279, April 2012.

[67] Jie Huang, Lei Liu, Wuyang Zhou, and Shengli Zhou. Large-girth nonbinary qc-ldpc codes of various lengths. *IEEE Transactions on Communications*, 58(12):3436 – 3447, Dec. 2010.

[68] E. Arikan and E. Telatar. On the rate of channel polarization. In *IEEE International Symposium on Information Theory (ISIT)*, 2009.

[69] E. M. Kurtas, A.V. Kuznetsov, and I. Djurdjevic. System perspectives for the application of structured LDPC codes to data storage devices. *IEEE Transactions on Magnetics*, 42(2):200 – 207, 2006.

[70] Chang Wu and J.R. Cruz. RS plus LDPC codes for perpendicular magnetic recording. *IEEE Transactions on Magnetics*, 46(16):1416 – 1419, 2010.

[71] Xie Ningde, Xu Wei, Zhang Tong, E. F. Haratsch, and Moon Jaekyun. Concatenated low-density parity-check and BCH coding system for magnetic recording read channel with 4 kb sector format. *IEEE Transactions on Magnetics*, 44(12):4784 – 4789, 2008.

[72] T. Mizuochi, Y. Konishi, Y. Miyata, T. Inoue, K. Onohara, S. Kametani, T. Sugihara, K. Kubo, H. Yoshida, T. Kobayashi, and T. Ichikawa. Experimental demonstration of concatenated LDPC and RS codes by FPGAs emulation. *IEEE Photonics Technology Letters*, 21(18):1302 – 1304, 2009.

[73] M. Bakshi, S. Jaggi, and M. Effros. Concatenated polar codes. In *IEEE International Symposium on Information Theory (ISIT)*, June 2010.

[74] H. Griesser and J. P. Elbers. Forward error correction coding. U.S. Patent, Jan 2009. US 7,484,165 B2.

[75] http://sigpromu.org/ldpc/.

[76] ITU-T. Forward error correction for high bit-rate dwdm submarine systems. INTERNATIONAL TELECOMMUNICATION UNION, Feb 2004. Series G.975.

[77] J. Ha, J. Kim, and S. McLaughlin. Rate-compatible puncturing of low-density parity-check codes. *IEEE Transactions on Information Theory*, 50(11):2824–2836, 2004.

[78] J. Ha, J. Kim, and S. McLaughlin. Rate-compatible punctured low-density parity-check codes with short block lengths. *IEEE Transactions on Information Theory*, 52(2):729–738, 2006.

[79] S. B. Korada. *Polar codes for channel and source coding*. PhD thesis, EPFL, 2009.

[80] H. Pishro-Nik, N. Rahnavard, and F. Fekri. Non-uniform error correction using low-density parity-check codes. submitted to IEEE Trans. Inf. Theory, May 2003.

[81] Eran Hof, Igal Sason, and Shlomo Shamai. Polar coding for reliable communications over parallel channels. In *IEEE Information Theory Workshop*, August 2010.

[82] Massimo Franceschetti and Ronald Meester. *Random Networks for Communication: From Statistical Physics to Information Systems*. Cambridge University Press, 2008.

[83] Ehud Friedgut. Sharp thresholds of graph properties, and the k-sat problem. *J. Amer. Math. Soc*, 12:1017–1054, 1999.

[84] Raphal Rossignol. Threshold for monotone symmetric properties through a logarithmic sobolev inequality. *Annals of Probability*, 34:1707–1725, 2005.

[85] Bhaskar Krishnamachari, Stephen B. Wicker, Rmon Bjar, and Marc Pearlman. Critical density thresholds in distributed wireless networks. In *Communications, Information and Network Security*. Kluwer Publishers, 2002.

[86] Gregory L. Mccolm. Threshold functions for random graphs on a line segment. *Combinatorics, Probability and Computing*, 13:373–387, 2004.

[87] S. Muthukrishnan and Gopal Pandurangan. The bin-covering technique for thresholding random geometric graph properties. In *Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 989–998, 2005.

[88] Piyush Gupta and P. Kumar. Critical power for asymptotic connectivity in wireless networks. *Stochastic Analysis, Control, Optimization and Applications: A Volume in Honor of W.H. Fleming, W.M. McEneaney, G. Yin and Q. Zhang (Eds.)*, 1998.

[89] F. Xue and P. R. Kumar. The number of neighbors needed for connectivity of wireless networks. *Wireless Networks*, 10(2):169–181, 2004.

[90] L. Booth, J. Bruck, M. Franceschetti, and R. Meester. Covering algorithms, continuum percolation and the geometry of wireless networks. *Annals of Applied Probability*, 13(2), May 2003.

[91] S. Shakkottai, R. Srikant, and N. Shroff. Unreliable sensor grids: Coverage, connectivity and diameter. *In the proceedings of IEEE INFOCOM'03, San Francisco, CA, April 2003.*

[92] S. Kumar, T. H. Lai, and J. Balogh. On k-coverage in mostly sleeping sensor network. In *MobiCom*, September 2004.

[93] H. Pishro-Nik and F. Fekri. Analysis of wireless ad-hoc and sensor networks in finite regime. In *5th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON)*, pages 73–81, 2008.

[94] H. Pishro-Nik. Analysis of finite unreliable sensor grids. In *WiOpt'06*, pages 243–254, April 2006.

[95] Paul Balister, Béla Bollobas, Amites Sarkar, and Santosh Kumar. Reliable density estimates for coverage and connectivity in thin strips of finite length. In *MobiCom '07: Proceedings of the 13th annual ACM international conference on Mobile computing and networking*, pages 75–86, 2007.

[96] M. Desai and D. Manjunath. On the connectivity in finite ad hoc networks. *Communications Letters, IEEE*, 6(10):437–439, October 2002.

[97] Nikhil Karmachandani, D. Manjunath, D. Yogeshwaran, and Srikanth K. Iyer. Evolving random geometric graph models for mobile wireless networks. In *Proceedings of the Fourth International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks, (WIOPT 2006)*, 2006.

[98] A. Ghasemi and S. Nader-Esfahani. Exact probability of connectivity in one-dimensional ad hoc wireless networks. *IEEE Communications Letters*, 10:251–253, 2006.

[99] L. Stockmeyer and V. Vazirani. Np-completeness of some generalizatiuons of the maximum matching problem. *Inform. Process. Lett.*, 15(1):14–19, 1982.

[100] A. Eslami, M. Nekoui, and H. Pishro-Nik. Results on finite wireless networks on a line. In *IEEE Information Theory Workshop (ITW)*, January 2010.

[101] A. Eslami, M. Nekoui, and H. Pishro-Nik. Results on finite wireless networks on a line. *IEEE Transactions on Communications*, 58(8):2204–2211, 2010.

[102] M. Penrose. *Random Geometric Graphs*. Oxford University Press, 2003.

[103] Piyush Gupta and P. R. Kumar. Internets in the sky: the capacity of three dimensional wireless networks. *Communications in Information and Systems*, 1:33–50, 2001.

[104] A. Papoulis and S. U. Pillai. *Probability, Random Variables and Stochastic Processes*. McGraw-Hill, 4th edition, 2001.

[105] A. K. Gupta and S. Nadarajah. *Handbook of Beta Distribution and Its Applications*. CRC, first edition, 2004.

[106] Erhard Godehardt and Jerzy Jaworski. On the connectivity of a random interval graph. *Random Struct. Algorithms*, 9(1-2):137–161, 1996.