Coding for DNA-based Storage Systems

Issam Maarouf

Thesis for the degree of Philosophiae Doctor (PhD)
University of Bergen, Norway
2024

UNIVERSITY OF BERGEN

Coding for DNA-based Storage Systems

Issam Maarouf

Thesis for the degree of Philosophiae Doctor (PhD)
at the University of Bergen

Date of defense: 24.05.2024

Year: 2024

Title: Coding for DNA-based Storage Systems

Name: Issam Maarouf

Print: Skipnes Kommunikasjon / University of Bergen

Acknowledgements

First, I would like to thank my two supervisors, Eirik Rosnes and Alexandre Graell i Amat, for their support, guidance, patience, and interest in working with me over the course of six years. I have learned a lot from both of you, and I appreciate all you have done to bring me to this moment.

Second, I would like to thank all my friends and colleagues at Simula UiB. I have grown immensely as a person at Simula, and I will be leaving this place filled with joy and core memories that will stay with me all my life. I felt very comfortable being myself in the office, and I am very proud to have contributed to the excellent work environment developed at Simula. The four years I have spent there were the best years of my life. Also, I would like to thank everyone in the "enemy" group at the Selmer Center. Thank you for all the fun memories we shared. I hope I left as much of an impact on all of you as you did on me.

To my core support system. To the people who have always been there for me. To the people closest to my heart. My lovely and beautiful family. My siblings, Bilal, Ismat, and Reina, and my parents, Ahmad and Iman. I am forever grateful for all of you. You have always believed in me and have always been my number one fan. Thank you so much for supporting me throughout my Masters and PhD. I know that being abroad has been difficult for all of you, but my mind and heart were always with you. It fills my heart with joy to make you proud, so I dedicate this Achievement to you. This achievement is also dedicated to my newborn niece, Sara.

To the person who owns my heart. My amazing and beautiful girlfriend, Irati. My final thanks goes to you. I would have been unable to go through the last year of my PhD without your support. You were there for me when I needed help the most, and all the love and support you have given me gave me the strength and courage to make it to this point. I close this chapter of my life while you are by my side, and I look forward to being by your side with all the coming future chapters. My life became instantly better and more exciting since you got into it. I dedicate this achievement to you, my love.

Abstract

Storing user data in deoxyribonucleic acid (DNA) strands, referred to as DNA-based storage, has become a hot topic during the last decade and has gained the interest of several research communities due to the success of several small-scale experiments that demonstrated the viability of DNA-based storage. This increased interest is warranted due to the theoretical superiority of DNA-based storage technology compared to traditional storage methods, as storing data in DNA can potentially provide both higher storage capacity and longevity. This success also inspired the study of the DNA storage channel from an information-theoretic point of view. In turn, this paved the way for the research problem of "Reliable communication over the DNA storage channel," the most relevant problem for this thesis, or, more precisely, reliable communication over channels that introduce insertion, deletion, and substitution (IDS) errors.

In this thesis, we design and optimize concatenated coding schemes that can combat errors that arise from the DNA storage channel and hence guarantee reliable communication. In addition, we introduce decoding algorithms for the proposed concatenated coding schemes that are tailored to the DNA storage channel. The proposed coding schemes have the form of an inner synchronization block code concatenated with an outer low-density parity-check code (or a polar code). Over the course of four papers, we study three different channel models. The first channel model is a simplistic model that introduces independent and identically distributed IDS errors. Studying this simplistic channel model gave us fundamental insight into the impact of IDS errors on code performance. The second and third channel models are more realistic extensions of the first model, where the second model includes strand erasures and the loss of ordering of the stored data, while the third model introduces correlated IDS errors. The design and optimization of the coding schemes for each case was done with the aid of asymptotic achievable information rates, finite-length bounds, extrinsic information transfer chart analysis, and density evolution.

Abstrakt

DNA-basert lagring, eller lagring av brukerdata i deoksyribonukleinsyre (DNA) tråder, kan teoretisk gi dramatisk høyere lagringskapasitet og lagringsrobusthet sammenlignet med dagens lagringsteknologier. Flere småskalaeksperimenter indikerer at DNA-basert lagring kan være mulig å implementere i praksis, og internasjonalt foregår det intens forskning rundt dette temaet. I tillegg til de biologisk aspektene, er det også viktig å studere DNA-lagringskanalen fra et informasjonsteoretisk ståsted. I DNA-basert lagring blir informasjon lagret som sekvenser av nitrogenbaser (adenin (A), tymin (T), guanin (G) og cytosin (C)). Lagringsprosessen er sårbar for såkalte IDS-feil, det vil si feil som følge av tilfeldig innsetting, sletting eller substitusjoner av nitrogenbaser i sekvensene.

Fokus for denne avhandlingen er utvikling og optimalisering av kodeløsninger som kan motvirke effekten av IDS-feil og derfor gi en pålitelig lagring. I tillegg introduserer vi algoritmer for å dekode de foreslåtte kodeløsningene som er skreddersydd for en DNA-basert lagringskanal. I fire artikler studerer vi tre forskjellige kanalmodeller. Den første kanalmodellen er en forenklet modell som introduserer uavhengige og identisk distribuerte IDS-feil. Ved å studere denne forenklete modellen fikk vi grunnleggende innsikt i virkningen av IDS-feil på kodeytelse. Den andre og tredje kanalmodellen er mer realistiske utvidelser av den første modellen, der den andre modellen inkluderer trådslettinger og tap av rekkefølge på lagrede data, mens den tredje modellen introduserer korrelerte IDS-feil. Design og optimalisering av kodeløsningene for hvert enkelt tilfelle ble gjort ved hjelp av asymptotisk oppnåelige informasjonsrater, grenser for endelig lengde og analyse av dekodingsalgoritmer.

List of Papers

This thesis is based on the following publications:

Paper I
I. Maarouf, A. Lenz, L. Welter, A. Wachter-Zeh, E. Rosnes, and A. Graell i Amat, *Concatenated Codes for Multiple Reads of a DNA Sequence*, IEEE Transactions on Information Theory, vol. 69, no. 2, pp. 910–927, Feb. 2023.

Paper II
I. Maarouf, G. Liva, E. Rosnes, and A. Graell i Amat, *Finite Blocklength Performance Bound for the DNA Storage Channel*, in Proc. International Symposium on Topics in Coding (ISTC), Brest, France, Sep. 2023, pp. 1–5.

Paper III
I. Maarouf, E. Rosnes, and A. Graell i Amat, *Achievable Information Rates and Concatenated Codes for the DNA Nanopore Sequencing Channel*, in Proc. IEEE Information Theory Workshop (ITW), Saint-Malo, France, Apr. 2023, pp. 1–5.

Paper IV
L. Welter, I. Maarouf, A. Lenz, A. Wachter-Zeh, E. Rosnes, and A. Graell i Amat, *Index-Based Concatenated Codes for the Multi-Draw DNA Storage Channel*, in Proc. IEEE Information Theory Workshop (ITW), Saint-Malo, France, Apr. 2023, pp. 1–5.

Contents

Acknowledgements **i**

Abstract **iii**

Abstrakt **v**

List of Papers **vii**

A Overview **1**

1 Background **3**

1.1 Introduction 3
1.2 Outline 5
1.3 Notation 5

2 DNA-Based Storage Systems **7**

2.1 Components of a DNA-Based Storage System 7
2.1.1 Data Encoding 7
2.1.2 DNA Synthesis 8
2.1.3 DNA Storage Medium 9
2.1.4 DNA Sequencing 10
2.2 The Coding Problem 11
2.2.1 Types of Errors 11
2.2.2 On the Receiver Side 13
2.3 Experiments 14
2.3.1 Church *et al.* 14
2.3.2 Goldman *et al.* 15
2.3.3 Grass *et al.* 15
2.3.4 Bornholt *et al.* 17
2.3.5 Erlich and Zielinski 18
2.3.6 Blawat *et al.* 20
2.3.7 Yazdi *et al.* 21
2.3.8 Chanda *et al.* 22
2.3.9 Discussion 22

3 Modern Coding Theory 25
3.1 Preliminaries of Coding Theory 25
3.2 Convolutional Codes 26
3.2.1 Algebraic Representation of Convolutional Codes 27
3.2.2 Decoding of Convolutional Codes 27
3.3 Low-Density Parity-Check Codes 30
3.3.1 Decoding of LDPC Codes (Belief Propagation) 30
3.3.2 Protograph-Based Representation of LDPC Codes 32
3.3.3 Concatenated Coding Schemes 34
3.4 Tools for Modern Codes 34
3.4.1 Density Evolution of LDPC Codes 35
3.4.2 Extrinsic Information Transfer Charts 37
3.4.3 Achievable Information Rates 39
3.4.4 Finite-Length Bounds 42

4 Channel Models for DNA-Based Storage 45
4.1 I.i.d. Insertions-Deletions-Substitutions Channel 45
4.2 Multi-Draw Insertions-Deletions-Substitutions Channel 47
4.3 Memory-k Event Dependent Insertions-Deletions-Substitutions Channel 48

5 Paper Overview 51
5.1 Paper I 51
5.2 Paper II 52
5.3 Paper III 52
5.4 Paper IV 53

6 Conclusion 55
6.1 Future Work 56

Bibliography 61

B Papers 63

I Concatenated Codes for Multiple Reads of a DNA Sequence 65
1 Introduction 67
2 System Model 70
2.1 Channel Model 70
2.2 Coding Scheme 70
2.3 Symbolwise MAP Decoding for the IDS Channel (Inner Decoding) 71
3 Symbolwise MAP Decoding for Multiple Received Sequences . . . 74
4 Concatenated Code Design 76
5 Achievable Information Rates 81
5.1 BCJR-Once Rate 81
5.2 Mutual Information Rate 86

6 Simulation Results 90
6.1 Inner Code Optimization/Comparison 90
6.2 Outer Code Optimization 91
6.3 Short Sequence Transmission 93
6.4 Multiple Sequence Transmission 94
6.5 Frame Error Rate Results With Substitution Errors 97
7 Conclusion 97
A Symbolwise APPs for Memoryless Channels 98
References 104

II Finite Blocklength Performance Bound for the DNA Storage Channel 105
1 Introduction 107
2 System Model 108
2.1 Channel Model 108
2.2 Coding Scheme 109
3 Bound on the Finite Blocklength Performance 110
4 Concatenated Coding Scheme Design 112
4.1 Inner Code 112
4.2 Outer Code 112
5 Numerical Results 114
5.1 Simulation Parameters 114
5.2 Discussion 115
6 Conclusion 117
References 118

III Achievable Information Rates and Concatenated Codes for the DNA Nanopore Sequencing Channel 119
1 Introduction 121
2 The Nanopore Sequencing Channel Model 122
2.1 Channel Transition Probability Estimation 123
2.2 Multiple Reads 124
3 Coding Scheme and System Model 124
4 Symbolwise MAP Decoding for Channel/Inner Code (Inner Decoding) 125
4.1 Decoding for a Single Received Sequence 125
4.2 Decoding for Multiple Received Sequences 127
4.3 Decoding Complexity 127
5 Achievable Information Rates 127
6 Concatenated Coding Scheme Design 128
6.1 Inner Code 128
6.2 Outer Code 128
7 Numerical Results and Discussion 129
8 Conclusion 131
References 133

IV Index-Based Concatenated Codes for the Multi-Draw DNA Storage Channel 135
1 Introduction 137
2 DNA Storage Channel Model 138
2.1 IDS Channel 138
2.2 Multi-Draw IDS Channel 138
3 Index-Based Concatenated Coding Scheme 139
4 Joint Index and Inner Decoding 140
4.1 Inner MAP Decoding 141
4.2 Clustering 142
4.3 Index and Multiple DNA Strand Decoding 142
5 Achievable Information Rates 143
6 Outer Code 144
7 AIR and FER Results 145
8 Conclusion 146
References 150

Part A

Overview

Chapter 1

Background

1.1 Introduction

Over the last decade, there has been notable attention from various research communities, particularly in the fields of bioinformatics and information theory, toward the utilization of deoxyribonucleic acid (DNA) as a medium for storing user data. Inspired by how well DNA information of ancient fossils has been preserved and how advanced DNA sequencing technologies have evolved, DNA-based storage has been identified as a promising solution for the next generation of data storage technology. In short, DNA-based storage uses DNA molecules to write and store user data, while reading the data back is done by sequencing those particular DNA molecules used for storage. The user data is contained within the sequence of bases making up the double helix structure in a DNA molecule, as shown in Fig. 1.1. Since there is a one-to-one correspondence between the two helix structures (each base can only be attached to one other base), the user data is stored in one helix.

The current focus in the research community is on making DNA-based storage technology more practical over the next decades with the ultimate goal of replacing current storage technologies. It is projected that at some point, the worldwide storage demands will exceed the limits of current state-of-the-art storage technologies, as their advancements will eventually fail to keep up as fundamental limits are reached (Moore's law will eventually cease to apply) [1]. Moreover, the physical components of current storage technologies need replacing as they get worn out, which is quite costly. Storing data in DNA molecules is expected to solve many of these problems, as it offers a much higher storage density and much higher longevity [1, 2]. While hard disks (HDs), dynamic access memory (DRAM), and flash memories (FMs) have a storage density of 10 GB/mm^3, 10 GB/mm^3, and 10000 GB/mm^3, respectively, DNA has a storage density of up to 10^7 GB/mm^3, almost a thousand times higher. In fact, it is estimated that the entire world's demand for storage by 2040 can fit in around 1 kg solution of DNA [1]. Secondly, if preserved well, DNA can remain viable to read for centuries. In fact, it is not uncommon to hear in the news that the DNA of ancient fossils has been sequenced, and these fossils can go back thousands of centuries or millennia. On the other hand, HDs and FMs should be replaced around every ten years [3], which might seem like a long time; however, this is problematic for long and sustainable storage applications. Evidently, DNA-based storage, if it becomes practical, is far superior

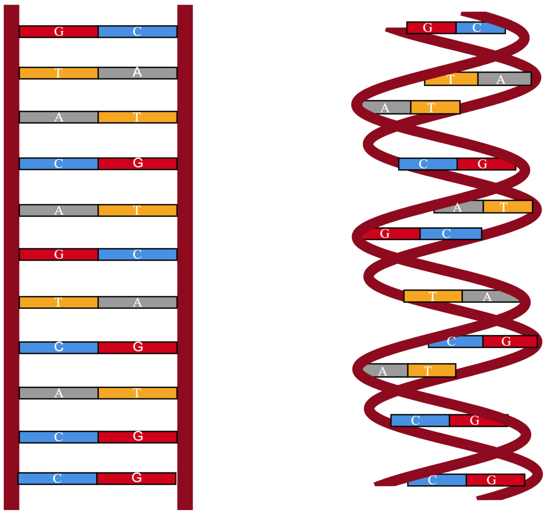


Figure 1.1: Double helix structure of a DNA molecule with user data being the singular bases.

	HD	FM	DRAM	DNA
Longevity	10 years	10 years	64 ms	$>$ 100 years
Storage density (GB/mm^3)	10	10^4	10	10^7

Table 1.1: Longevity and storage density of typical storage technologies compared to DNA storage.

to other technologies. Table 1.1 briefly summarizes the longevity and density of different storage technologies.

This research interest paved the way for many small-scale experiments demonstrating the feasibility of DNA-based storage [4–18]. The success of these experiments showed that DNA-based storage could in fact become a viable technology. This success also caught the attention of the information theory community and motivated studying the DNA storage channel from an information-theoretic point of view. In particular, several information-theoretic problems have been identified. The most important problem for this thesis is reliable communication over channels that introduce insertions, deletions, and substitutions (IDSs) [19] as the processes of DNA storage introduce errors in the form of IDSs. Furthermore, in the literature, channels that introduce IDSs have been proposed to model synchronization errors. Coding techniques are an indispensable component to cope with IDS errors and improve the reliability of DNA storage systems and channels that are prone to desynchronization. Both aspects will be explained in the next chapters.

1.2 Outline

The rest of the thesis is divided as follows. This chapter will conclude with a notation description and clarification. Chapter 2 defines and describes the components of a DNA-based storage system. It mainly focuses on providing the necessary background to identify the (coding) problem that needs to be solved by a coding scheme. Chapter 2 also contains a section summarizing the most relevant experiments for this work that showed the viability of using DNA for storing data. Chapter 3 gives the necessary preliminaries of modern coding theory, defines convolutional and low-density parity-check (LDPC) codes, introduces some decoding algorithms, and provides code performance tools. Chapter 4 introduces three DNA storage channels that are used and studied in the papers. Chapter 5 gives an overview of the 4 papers that make up this thesis. Chapter 6 concludes Part A of the thesis and gives an outline for future work. Finally, Part B includes the 4 papers titled *Concatenated Codes for Multiple Reads of a DNA Sequence*, *Finite Blocklength Performance Bound for the DNA Storage Channel*, *Achievable Information Rates and Concatenated Codes for the DNA Nanopore Sequencing Channel*, and *Index-Based Concatenated Codes for the Multi-Draw DNA Storage Channel*.

1.3 Notation

Throughout Part A of the thesis, we use the following notation:

- Lowercase letters (e.g., x) denote scalars, lowercase bold letters (e.g., $\boldsymbol{x}$) denote vectors, uppercase bold letters (e.g., $\boldsymbol{X}$) denote matrices, and uppercase calligraphic letters (e.g., $\mathcal{X}$) denote sets. We denote random vector variables as x or $\boldsymbol{x}$ unless stated otherwise.
- Vectors are row vectors unless otherwise specified.
- We denote a finite field of size q by $\mathbb{F}_q$.
- We denote set cardinality by $|\cdot|$.
- We denote vector and matrix transposition by $(\cdot)^T$.
- For natural numbers m and n, we denote the all-zero matrix of size $m \times n$ by $\mathbf{0}_{m\times n}$, and $\boldsymbol{I}$ denotes the identity matrix.
- $\Pr(\mathsf{x} = \boldsymbol{x})$ is the probability that a realization of the random vector variable x has the value $\boldsymbol{x}$, while $\Pr(\mathsf{x} = \boldsymbol{x} | \mathsf{y} = \boldsymbol{y})$ is the conditional probability. For simplicity, sometimes the probability of an event is expressed as $p(\cdot)$.
- We denote the expected value of the random vector variable x as $\mathbb{E}[\mathsf{x}]$.
- We denote mutual information by $I(\cdot;\cdot)$, and $H(\cdot)$ and $H(\cdot|\cdot)$ denote entropy and conditional entropy, respectively.
- We denote a code by $\mathcal{C}$.

For the papers that make up Part B, we introduce the notation used on a per-paper basis.

Chapter 2

DNA-Based Storage Systems

In this chapter, we describe the basic principles of DNA-based storage systems and define their individual components. We introduce and explain processes the user information goes through, from encoding to recovery, in a DNA storage system. We then introduce the types of errors that arise from these systems and define the decoding problem that needs to be solved when designing error-correction schemes. Finally, we summarize several small-scale experiments that showed the viability of DNA as a storage medium [4–10, 13, 15], as they were a great inspiration for the work in this thesis.

2.1 Components of a DNA-Based Storage System

Fig. 2.1 shows a block diagram representation of a typical DNA-based storage system. This is the state-of-the-art of DNA-based storage technology; however, this is, of course, subject to change as the technology evolves. In the following sections, we describe the tasks involved in every component and the technologies used.

2.1.1 Data Encoding

First, the digital data is encoded (mapped) into the DNA bases (alphabet of A, T, C, G), also referred to as nucleotides, where the mapping is chosen according to several constraints. There are two main constraints that can cause many errors in the later stages if not taken care of. First, homopolymer runs (runs of identical bases, e.g., $(\ldots AAAAA \ldots)$) should be avoided since they result in high error rates in DNA sequencing. As shown later, all reference experiments have targeted limited homopolymer runs, ranging from 5-6 bases as a maximum. Second, a DNA strand should have balanced GC content (percentage of GC bases in a data block should be close to 50%) [15, 20]. Other constraints include a mapping that reduces the probability of errors.

The current DNA synthesis and sequencing technologies process the digital data in blocks (strands/sequences). Usually, each block is of equal length n nucleotides (nts). We refer to a block as a strand when it exists as a DNA molecule and as a sequence when it is being processed digitally. Fig. 2.2 shows the composition of a DNA strand/sequence. The DNA sequence consists of two primer blocks (placed at

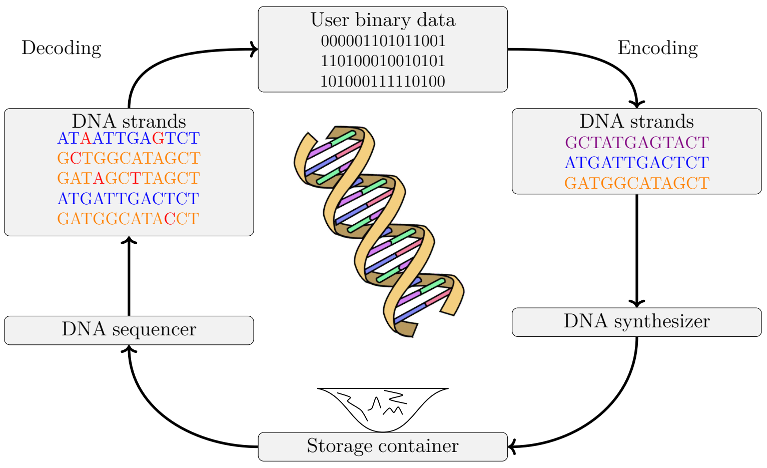


Figure 2.1: DNA-based storage system block diagram.

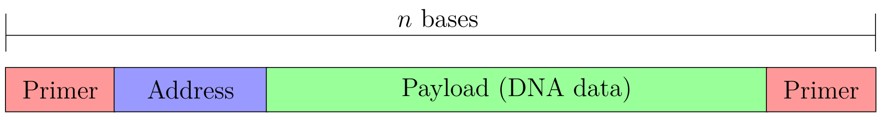


Figure 2.2: DNA data block.

both ends), an address block, and a payload (DNA data) block. Primers are used in a process called polymerase chain reaction (PCR) amplification, which creates many copies of the data block contained in between the target primers. This will be explained in more detail later. This is helpful in accessing only the target data in a DNA pool full of other strands. Primer sequences are useful in archival-based systems. The address block is used to distinguish and identify the target DNA sequence location in the input digital data. Since the order these data blocks are synthesized, on the receiver side, is not known for reasons explained later, an address sequence that conveys the order of the block is added to it. Finally, the payload block is where the actual data is encoded. Error-control codes are usually placed in this block.

2.1.2 DNA Synthesis

The synthesizing of DNA strands presented in this thesis falls under the De novo DNA synthesis umbrella that allows the creation of DNA sequences without using ready-made sequences (pre-existing templates)[15]. More specifically, this synthesis method is called chemical oligonucleotide synthesis, or more precisely, phosphoramidite oligonucleotide synthesis. This method adds nucleotides, step by

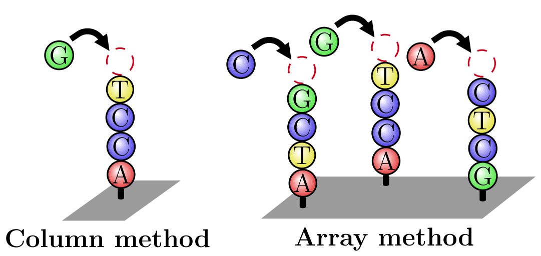


Figure 2.3: Chemical oligonucleotide synthesis (left is using the column method, and right is the array method).

step, where the first nucleotide is attached (hinged) to a solid support. The chain is grown until the target sequence is built, and the entire sequence is detached from the support and added to the gene pool [15]. There are two platforms for oligo synthesis. The first is the column-based oligo synthesis, where the strands are created column by column (one chain at a time), while the other platform, array-based oligo synthesis, can have several columns attached to the same solid support, and all of them can be grown at the same time [15, 20] (see Fig. 2.3).

Since all the current DNA synthesis technologies cannot synthesize very long sequences, the digital data is segmented into fragments that amount to 150-200 nucleotides per sequence [15, 20]. In practice, synthesis technologies cannot create single DNA strands; however, they create a set of strands, where each set is made up of around a million copies. Furthermore, each set is generated on a spot (2-D surface) of the solid support surface, and unfortunately, the number of copies per spot will not be evenly distributed [19]. In addition, some strands in each set will contain synthesis errors and will not be grown to the correct length [19]. Hence, to create single-stranded valid DNA sequences, the complete DNA strands are PCR amplified in the pool. This will increase the concentration of the valid DNA sequences and will dilute out the incomplete ones [15, 19]. As a result, the gene pool will be made up of a multi-set of DNA sequences, where the number of copies of each sequence is not evenly distributed.

2.1.3 DNA Storage Medium

The synthesized DNA strands can be either stored in living cells, referred to as in vivo, or stored in vitro, where they are placed in solutions (gene pools). The stored DNA strands should be placed in a preserving environment, where they are guaranteed to be maintained for long periods of time. Each DNA strand makes up a DNA molecule in the gene pool. The DNA data can be inside one pool or several, depending on the structure of the storgae system. In this thesis, we deal with in vitro storage.

As mentioned earlier, many strands will be floating in this gene pool, where they can come from the same or different files. Accessing specific files over others is a necessary feature in DNA-based storage systems. One way to achieve this is by increasing the concentration of the strands belonging to a specific file. In other words, these strands dilute the solution. Typically, before the sequencing process, a

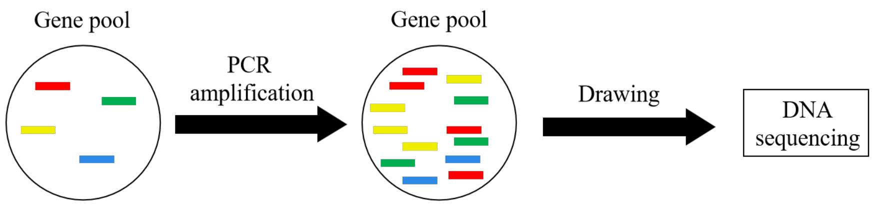


Figure 2.4: PCR amplification and sampling/drawing in a gene pool.

small sample of the gene pool will be taken, and then the strands of the target file will be PCR amplified to increase their concentration. In this way, it is guaranteed with a high probability that strands from the target file will be sequenced. Fig. 2.4 shows a simplified block diagram of the PCR amplification and sampling process in a gene pool. From a biological perspective, PCR amplification works as follows. The temperature of the sample from the gene pool, where the required data block is located, is raised and dropped between a maximum and a minimum temperature for several cycles. With each cycle, an enzyme replicates the strands containing the target primer sequences [15]. Ideally, each replication cycle should double the number of target sequences existing in the sample. However, practically, the amplification factor is 1.85 at maximum [19]. In addition, ideally, the pool sample should contain an equal concentration of all sequences; in addition, PCR amplification should amplify the target sequences equally. However, in practice, a bias toward some sequences is frequent [19]. This inherent bias might result in some sequences not being read, which is the primary source of erasure errors [19, 21].

2.1.4 DNA Sequencing

DNA sequencing is performed via state-of-the-art sequencing technologies, and in the small-scale experiments to be summarized later, the two technologies used are Illumina and the Oxford Nanopore MinION platform. Illumina is most frequently used since it is the older technology. The exact procedure for sequencing by both Illumina and Oxford nanopore is rather complicated and goes beyond the scope of this thesis, but the most important steps in this procedure are as follows.

In Illumina sequencing, a DNA strand is read in cycles, where each cycle reads one base only. A strand is attached to a flow cell and is flooded with fluorescently tagged nucleotides [15, 20]. Base by base, the tagged nucleotides compete to attach to the target base, and only the corresponding nucleotide will attach to it (e.g., if the target base is A, then only a tagged base T will attach). Once the tagged base attaches, it emits a fluorescent signal in the flow cell, characterized by its emission wavelength and light intensity. The signal is detected as an image, and the target base is determined using image processing [20]. This process is referred to as sequencing by synthesis. This is done until the entire strand is read. In practice, a target strand will have many duplicates. Strand copies will be read simultaneously, and an output sequence will be determined using the reads [20] (see Fig. 2.5(a)).

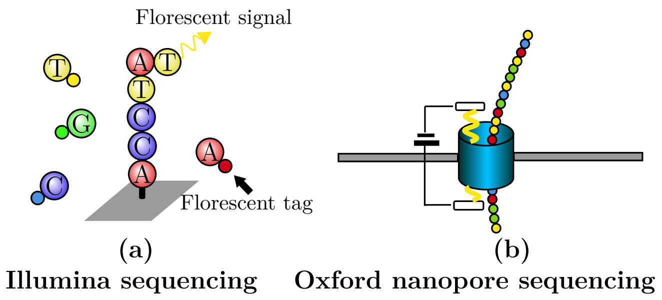


Figure 2.5: Illumina and Oxford nanopore sequencing.

In Oxford nanopore sequencing, a nanopore hole is created through a fixed voltage-biased membrane [20, 22]. Two ionic-filled-solution chambers are placed above and below the membrane. The upper chamber is called a cis, and the lower one is called trans. A voltage is applied across the chambers such that an ionic current flows through the nanopore. This ionic current will change profile (current intensity and direction) depending on the molecule that passes through the nanopore. A DNA strand will be driven into the nanopore using an enzyme placed at the top of the nanopore. The speed at which the enzyme will drive the strand can be controlled to aim for the insertion of one base at a time into the pore, targeting changing the current profile one base at a time. Ideally, each base value (A, T, C, G) will have a unique characteristic ionic current profile in the nanopore. This characteristic profile will be measured, and the target base will be determined [20, 22] (see Fig. 2.5(b)).

Illumina provides higher sequencing accuracy at the expense of higher cost and shorter base generations per read. In contrast, Oxford nanopore sequencing provides longer base reads with lower cost at the expense of lower accuracy. Illumina is still the dominant and most used technology in practice, but Oxford nanopore sequencing is gaining popularity and is expected to soon be the dominant one.

2.2 The Coding Problem

At every component of a DNA-based storage system, errors can be introduced. The job of the decoder is then to correct these errors, which is not a simple task. In this section, we will explain the different types of errors that arise at each stage of the storage procedure. In addition, we will explain the tasks the decoder has to do to recover the data.

2.2.1 Types of Errors

Substitution errors are considered the simplest errors that arise from DNA-based storage. They are the easiest errors to correct from the decoder's point of view. These errors are similar to the traditional bit flips in binary symmetric channels. Similar to the channel flipping bit 0 to bit 1 and vice versa, the DNA storage

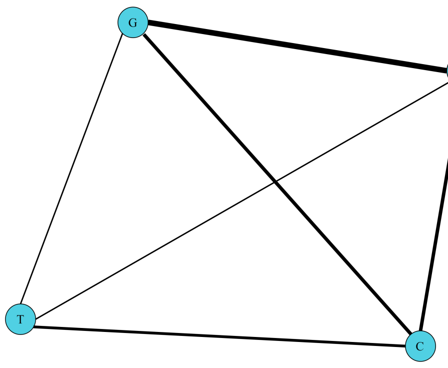


Figure 2.6: Graphical representation of substitution probabilities in DNA-based storage systems.

"channel" will substitute a particular base, for example, base A, for another base, for example, base C. As an example of a substitution error in a sequence,

$$\ldots \mathrm{CGGTG} \ldots \rightarrow \ldots \mathrm{CGGT}{\color{red}\mathrm{T}} \ldots$$

However, unlike the binary symmetric channel, substitution errors are not symmetric. Due to the chemical structure of a DNA base, some bases are more likely to be substituted with some bases than others [11]. For example, Fig. 2.6 shows a graphical representation of the substitution probabilities of bases as computed in a dataset we use in some of the work in this thesis [23]. The thicker the line appears, the higher the substitution error probability.

Deletion errors happen when a base in a sequence is deleted. In other words, it is removed from the sequence. Not to be confused with erasure errors, here, when a base is deleted, the decoder does not know where in the sequence the deletion happened. Since the length of the target sequence is known to the decoder, when the received sequence has a length less than the target one, it knows that one or several deletions have occurred. However, the location of those deletions is not known. As an example of a deletion error in a sequence,

$$\ldots \mathrm{CGGTG} \ldots \rightarrow \ldots \mathrm{CGTG} \ldots$$

Insertion errors happen when one or several bases in a sequence are inserted ahead of another base. Similarly to a deletion error, the decoder has no knowledge of where the insertion error occurred. The decoder is only aware that insertions have happened when the received sequence length is longer than expected. As an example of an insertion error in a sequence,

$$\ldots \mathrm{CGGTG} \ldots \rightarrow \ldots \mathrm{CG}{\color{red}\mathrm{A}}\mathrm{GTG} \ldots$$

When both insertion and deletion errors occur in the target sequence, the received sequence length may differ from the target one. From the decoder's perspective, if the received sequence is longer than the target one, the decoder only knows that more insertions than deletions have occurred, and vice versa. This is problematic for the decoder since, with insertion and deletion errors, the received and target sequences are out of sync. In the literature, insertion and deletion errors are referred to as synchronization errors since they induce uncertainty on the location of data in a sequence. In addition, these errors are the hardest to deal with, and they degrade the performance of the coding scheme [24].

Strand erasures, which do not happen as often, are the last type of error. Here, the entire target sequence is removed; in other words, it has not been retrieved at the decoder. This error mainly happens when sampling from a gene pool and the target sequence is not included. It also can result from a failed PCR amplification process, where, for some reason, the enzyme failed to replicate this sequence.

IDS errors and strand erasure can occur at any stage of the DNA storage process. Whether they are due to the imperfections in the chemical processes of synthesizing the strands and PCR amplification or the sequencing technology used, all three types of errors are possible. We refer the reader to [19] for a more detailed breakdown of how and when these types of errors occur.

2.2.2 On the Receiver Side

It becomes more evident now that the decoder's/receiver's job is very challenging when considering all the types of errors that can occur. The decoder needs to correct substitution errors, regain synchronization by solving insertion and deletion errors (including finding the locations where they occurred), and deal with entire sequences being erased. On top of that, the order of data blocks segmented for synthesis and sequencing is lost in the process. Although a target strand will have an abundance of copies in the gene pool, which is an inherent redundancy, these copies will obviously have errors, leading us to the next challenge for the decoder. Here, the decoder also has to cluster or group the strands that are similar to each other to gain from the redundancy. Recovering a target sequence from its noisy copies in a DNA storage channel is referred to as *trace reconstruction* in the literature.

Fig. 2.7 aids in visualizing the decoding problem faced by the receiver. We have a set of unordered sequences, noisy copies of the target ones. Then, the decoder has first to group similar copies together to form a cluster, where a cluster should ideally represent the target sequence. Then, the decoder will combine these copies to make a consensus on the target strand. Finally, the decoder has to find the order of these consensus strands in the original file. Some strands are high in errors and cannot be correctly clustered or grouped.

In this thesis, as shown later, we first assume an idealistic scenario where clustering and ordering are done perfectly. Studying the DNA storage problem in an idealistic setting will help to fundamentally understand the impact of IDS errors without the impact of the other problems. However, we do expand our idealistic scenario to incorporate the clustering and ordering problems. In those cases, clustering is performed on the copies by observing how similar the sequences are to each other, while ordering is performed by reading the index on the received

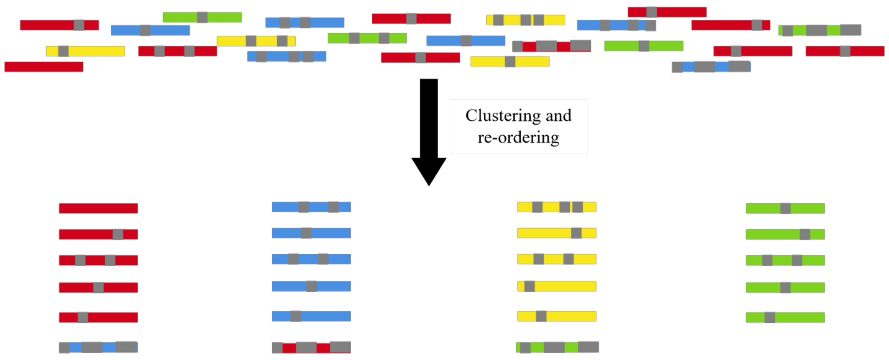


Figure 2.7: The decoding problem.

sequence. Obviously, these processes are not ideal and are not guaranteed to be perfectly done

2.3 Experiments

As mentioned earlier, several works have been done on DNA-based storage systems, both in theoretical and practical frameworks. Each paper presented its design for address and primer sequences, its own archival or random-access system design, and its design to handle all error types. Table 2.1 compares the sequencing technology used, the data size synthesized, the achieved system information capacity, error-detection/correction usage, and random-access usage of different papers on this topic.

Authors	Sequencing technology	Data size	Inf. capacity (bits/nts)	Error-detection/correction	Random access
Church *et al.* [4]	Illumina	5.27 MB	0.60	None	No
Goldman *et al.* [5]	Illumina	0.74 MB	0.19	Detection	No
Grass *et al.* [6]	Illumina	0.08 MB	0.86	Correction	No
Bornholt *et al.* [8]	Illumina	0.15 MB	0.57	None	Yes
Erlich and Zielinski [9]	Illumina	2.11 MB	1.18	None	No
Blawat *et al.* [7]	Illumina	22 MB	0.89	Correction	No
Chanda *et al.* [13]	Illumina	0.24 MB	1.52	Correction	Yes
Yazdi *et al.* [10]	Nanopore	0.003 MB	1.71	Correction	Yes

Table 2.1: Comparison of experiments in the literature.

2.3.1 Church *et al.*

Church *et al.* [4] developed the first large-scale DNA-based storage architecture. Here, every 0 data bit is encoded into an A or C base, and every 1 data bit is encoded into T or G base. A base is chosen randomly for each bit as long as no homopolymer runs of more than three bases are created. In addition, this data mapping choice allows control over the GC content balance.

The authors encoded 5.27 MB of data in DNA. The data is divided into $54,898$ blocks, each of length 159 nucleotides (bases). Each block is divided as follows: 96 nucleotides are used for information, 19 nucleotides for addressing, and 22 nucleotides are used for each primer sequence. The address sequences start from $00\ldots001$, and are appended by one in binary for each next sequence. One primer sequence is placed at the beginning of the block, referred to as the forward primer, and the reverse complement of it is placed at the end and referred to as the backward primer. The DNA data was stored in a library (gene pool).

For sequencing and data retrieval, the authors used PCR amplification to create several copies of the DNA data and chose only the sequences with the correct blocklength (115 nucleotides without primers) and a valid address sequence. A post-processing step was done on these sequences where overlapping blocks were compared, and a consensus of the output DNA sequence was decided.

2.3.2 Goldman *et al.*

In Goldman *et al.* [5], the digital data is first ASCII encoded using one byte per symbol. Afterward, each ASCII symbol is converted to 5 or 6 ternary digits (trits) using an optimal Huffman code (compression). Each trit is converted to a DNA symbol using differential encoding following the rules in Table 2.2. This type of differential encoding limits the homopolymer runs to one base.

The resulting DNA data is then segmented into blocks of length 117 nucleotides. Each block contains 100 nucleotides for information data, 2 nucleotides for file identification, 12 nucleotides for intra-file location information, and two parity-check nucleotides, one placed in the beginning and one at the end. Each segment overlaps in 75 nucleotides with its adjacent segments (this gives $4\times$ coverage for each base). This is equivalent to using a repetition code. The information content of each alternate segment is reverse complemented as well, and a parity-check nucleotide is added to indicate this, as shown in Fig. 2.8, resulting in a total of 117 nucleotides for each block. The size of the digital data file used is 739 KB.

	next		
previous	0	1	2
A	C	G	T
C	G	T	A
G	T	A	C
T	A	C	G

Table 2.2: Differential encoding used in Goldman *et al.*.

2.3.3 Grass *et al.*

Since no advanced error-correction/detection codes were used in the previous two papers, Grass *et al.* [6] proposed an architecture that uses Reed-Solomon (RS)

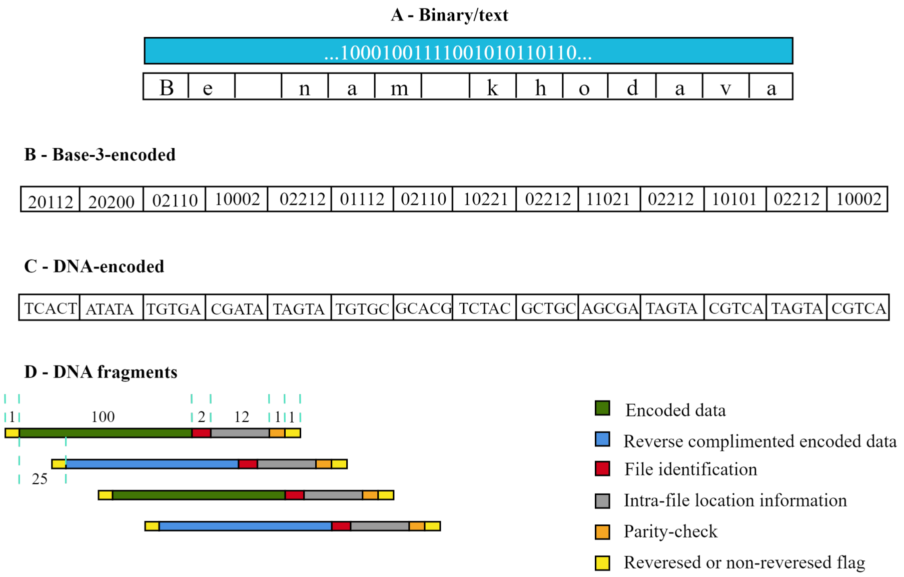


Figure 2.8: Goldman *et al.*'s encoding method using ASCII and differential encoding, Huffman compression, four-fold coverage, reverse complementation of alternate data blocks, and single parity-check coding.

codes. The authors also proposed physical storage media that prolong the DNA quality and stability.

The proposed architecture works as follows. First, every two bytes of digital data are mapped to three elements in $\mathbb{F}_{47}$ by converting data in base 256^2 to base 47^3. This information is then arranged in matrices of dimensions 39×594 (39 rows and 594 columns). To generalize, each matrix is made of m blocks (rows), each of length k (columns). Afterward, an outer RS code is applied to the beginning of each row, resulting in a total blocklength of $n = k + (n - k)$, where $n - k$ is the amount of added redundancy. Then, an index of length I is added to each column and then encoded by an inner RS code, adding a redundancy of B to each column. The dimensions of each matrix now equal $N \times n$, where $N = m + I + B$. Each column is then converted to a DNA strand by the mapping shown in Fig. 2.9C. The codon wheel works by mapping each element in $\mathbb{F}_{47}$ to a triplet of bases, where the bases are chosen in the order of least significant base to the most significant one. For example, element 44 will be mapped to AGT (not TGA). This encoding ensures that no base is repeated more than three times. Two primers are then added to the beginning and the end of each column, resulting in the DNA sequence being synthesized. In the paper, the parameters used were as follows: $k = 594$, $n = 713$, $m = 30$, $I = 3$, and $B = 6$.

Finally, the data is recovered after sequencing by first decoding the inner code (correcting individual base errors), then using the indexing information to sort the sequences, then decoding the outer code for recovering/correcting lost/whole sequences, and finally translating the sequences into $\mathbb{F}_{47}$. The entire storage architecture, from synthesis to data recovery, is shown in Fig. 2.9.

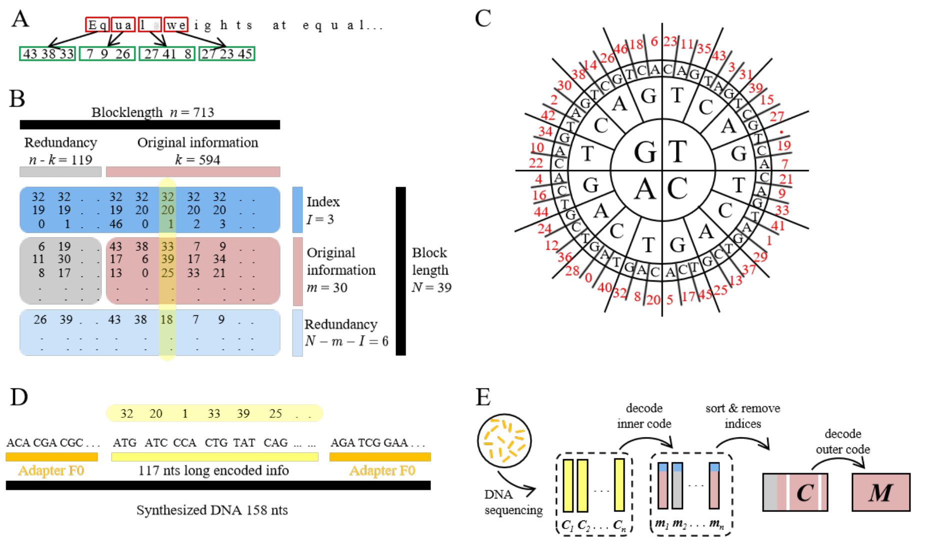


Figure 2.9: Grass *et al.*'s storage system. (A) Every two bytes of data are mapped to three elements in $\mathbb{F}_{47}$. (B) An outer RS code adds a redundancy of $n - k = 119$ to each row of the individual blocks. To each column, an index of length 3 is added, and a redundancy of $B = 6$ is generated using a second (inner) RS code. (C) Each column is mapped to DNA bases using an $\mathbb{F}_{47}$ to a DNA codon wheel. (D) Two primer sequences are added to each DNA strand for PCR amplification. (E) To recover the original information from the DNA strands, the read sequences are translated to $\mathbb{F}_{47}$ and are decoded by first decoding the inner code (correcting individual base errors), second by sorting the sequences by means of the index, followed by outer decoding, which allows the correction of whole sequences and the recovery of completely lost sequences.

2.3.4 Bornholt *et al.*

Bornholt *et al.* [8] was one of the first papers to implement random access in a DNA-based storage architecture. The authors used different primer sequences for different DNA strands, such that only the data assigned to the target primer will be PCR amplified and sequenced. They implemented the above by mapping a key to each pair of PCR primers placed in the strands. When a certain DNA strand with its corresponding primers is needed, the storage system looks for the key associated with the primers, and PCR amplifies the strand. In this storage system, the strands are separated into different pools, as shown in Fig. 2.10.

The digital data is encoded using a base three Huffman code, similar to Goldman *et al.*, where each byte in the digital data is mapped to 5 or 6 trits. The trits are then encoded into DNA data following the rules in Table 2.3. The DNA data is arranged in the same way as in Goldman *et al.* [5]. Two primer sequences are placed in the beginning and at the end of the strand, where these sequences are generated by encoding the key digital value using Table 2.3. Furthermore, an address block is added, along with two sense nucleotides ("S"), to indicate whether the payload data is reverse complemented or not. The address sequences (blocks) are placed to identify the location of the data block in the input data string. The

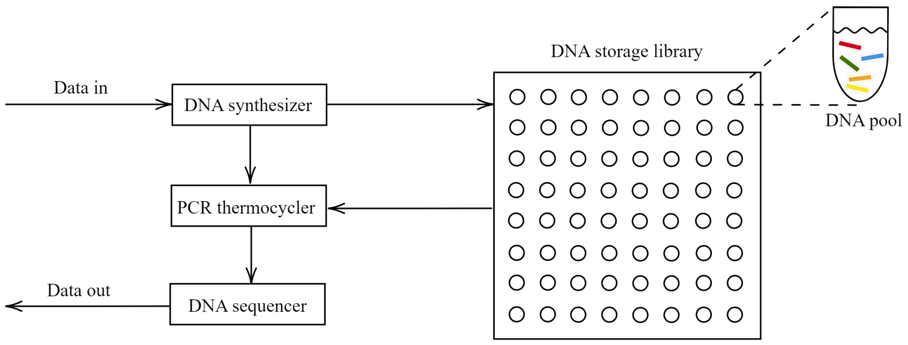


Figure 2.10: Bornholt *et al.*'s storage system.

address block is divided in two parts. The first part includes the key the block is associated with, and the second part indexes it.

To recover the data correctly, the authors used XOR decoding on the data blocks. For every two DNA data blocks, a third redundant block is created by XOR-ing all their data values. In other words, a data block $\boldsymbol{a}$ and a data block $\boldsymbol{b}$ will have a redundant block $\boldsymbol{a} \oplus \boldsymbol{b}$ associated with them. This encoding is exploited in the sequencing stage, where retrieving any two blocks from the selection $\boldsymbol{a}$, $\boldsymbol{b}$, and $\boldsymbol{a} \oplus \boldsymbol{b}$ is enough to recover $\boldsymbol{a}$ and $\boldsymbol{b}$.

	Previous nt			
	A	C	G	T
0	C	G	T	A
1	G	T	A	C
2	T	A	C	G

Table 2.3: DNA data encoding used in Bornholt *et al.*'s paper.

2.3.5 Erlich and Zielinski

Erlich and Zielinski [9] proposed a strategy for storage called DNA fountain that exploits the strength of fountain codes over channels with information dropouts. However, their strategy does not offer random access, but it does achieve the third-highest spectral efficiency with 1.18 bits/nt.

The proposed encoding of digital data is quite unique and is shown in Fig. 2.11. First, the data is divided into non-overlapping segments, each having the same fixed length, where each segment is referred to as a droplet. This process is referred to as "pre-processing." Then, the data is iterated between two computational processes, the Luby transform process and the screening process. In the Luby transform, several data segments from segmentation are chosen randomly for bitwise addition. This sets the basis for fountain codes. The summed output is then appended with a short-length seed and is given to screening. The seed is related to the

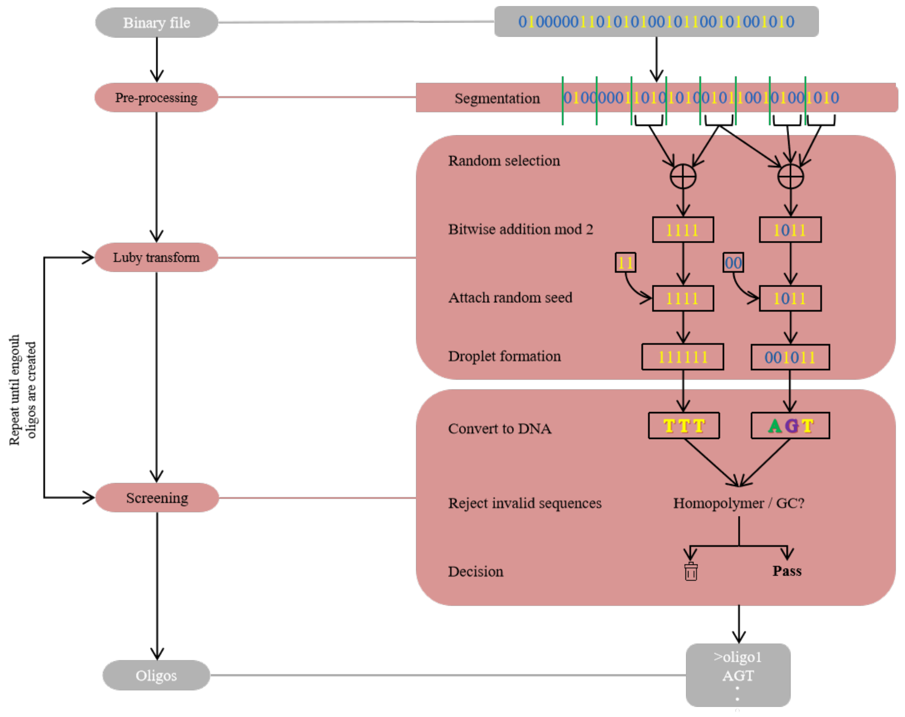


Figure 2.11: DNA fountain architecture.

random number generator of the transform during the creation of the droplet, and it allows the decoding algorithm to identify the segments in the droplet. In screening, the output data is tested for balanced GC content and homopolymer runs of more than 3 bases. If the output data does satisfy the constraints, then it is passed for synthesis; otherwise, it is discarded. The digital data in the screening process is mapped as follows: 00 $\rightarrow A$, 01 $\rightarrow C$, 10 $\rightarrow G$, and 11 $\rightarrow T$. These two processes are repeated until enough data strands are created for passing to synthesis. Fig. 2.11 shows the DNA fountain architecture.

The authors encoded 2.11 MB of data using their DNA fountain strategy. The data is split into $67,088$ segments, each one being 32 bytes in size. These segments are then iterated between the computational steps of DNA fountain until valid strands are created. Each droplet is 38 bytes in size: 4 bytes of seed, 32 bytes of payload data, and 2 bytes are used for the redundancy symbols of an RS code. The final DNA strand is made up of 200 nucleotides, 48 of which constitute primer sequences.

After sequencing using PCR amplification and Illumina sequencing technology, the decoder recovered the droplets in their binary form, discarded the droplets with errors as given by the RS code, and finally used a message-passing algorithm to recover the information from the Luby transform process.

2.3.6 Blawat *et al.*

Blawat *et al.* [7] also used a unique digital to DNA data encoding, which is summarized in Table 2.4. In this encoding scheme, each byte of digital data is transformed into a 5 nucleotides DNA symbol. The first six bits are mapped normally, where the two-bit tuples 00, 01, 10, and 11 are mapped to DNA bases A, C, G, and T, respectively. The last two bits are mapped to two DNA bases according to Table 2.4 and following some constraints. The constraints are as follows: the first three bases and the last two bases cannot be the same. In particular, in a DNA symbol, the base for the first two bits is placed as the first base, the base for the second bit tuple is placed as the second base, and the base for the third bit tuple is placed as the fourth base. This means the two bases corresponding to the last tuple are placed as the third and fifth bases in the DNA symbol. Consequently, from Table 2.4, there will be no homopolymer runs of more than three bases. If we take, for example, the byte 00011011, the mapping for the first 6 bits will be $AC - G-$. Hence, the DNA symbol can be one of the following four (according to Table 2.4): $ACAGT$, $ACCGA$, $ACGGC$, and $ACTGG$. Following the aforementioned constraints, $ACTGG$ is an invalid symbol. Generalizing the above example, every data byte sequence is guaranteed to have at least two valid DNA symbol mappings, and most of them will actually have three. In [7], the authors say that 256 out of the 256 possible data bytes have two mapping choices, while 208 out of the 256 have three mapping choices. Hence, the symbols available to be used can be grouped into three clusters: two complete ones, denoted as clusters A and B, respectively, and a partial one, denoted by cluster C. To this end, cluster C will be ignored, and each byte mapping will be chosen from either cluster A or cluster B in a controlled manner, e.g., alternatively. The clusters are used in the decoding stage, and according to the authors, this encoding scheme is useful in detecting insertions and deletions [7].

Value	Base
00	A
01	C
10	G
11	T

First 3 bit tuples

Value	Option 1	Option 2	Option 3	Option 4
00	AA	CC	GG	TT
01	AC	CG	GT	TA
10	AG	CT	GA	TC
11	AT	CA	GC	TG

Last bit tuple

Table 2.4: Balwat *et al.*'s digital to DNA data mapping.

Similar to Grass *et al.*, DNA sequences are arranged in matrices, and error-correction codes are applied to both rows and columns. More specifically, the address of every strand is protected individually with a BCH code of length 63 and dimension 39. The columns are encoded with an RS code of length 255 and dimension 223 (strands are arranged in rows). Finally, an error-detection code is placed at the end of each strand; the error-detection code used in the paper is a CCITT 16-bit cyclic redundancy check code. This triple data protection scheme further helps in the correction and detection of insertion and deletion errors.

2.3.7 Yazdi *et al.*

In their paper, Yazdi *et al.* [10] achieved the highest spectral efficiency out of all the papers. They achieved this by trying to build a storage system that is as suitable as possible for a DNA storage channel. The authors reduced the cost of synthesis by using an efficient compression algorithm, along with mapping schemes that prohibit homopolymer runs and provide balanced GC content. Furthermore, the system provides random access using so-called gBlock codewords (long DNA strands specific to nanopore sequencing), each accompanied by carefully constructed address sequences. These address sequences are used in PCR amplification to separate the different strands during retrieval. The constructed addresses are made unique by having large mutual Hamming distances along with low mutual correlations. By having a low mutual correlation, no substring of any address sequence will match with the substrings of the others, and no address sequence can be confused with erroneous received codewords. Error-detection and correction of the received strands are ensured by a construction algorithm based on consensus sequences and a deletion error-correcting code tailored for the nanopore channel. The consensus sequence construction algorithm combines classical multiple alignment methods with side information given by the address sequences; in addition, it exploits the algebraic structure of the gBlocks.

The authors performed digital to DNA data encoding in three encoding steps: the first controls the GC content balance of each DNA data block, the second controls their address sequences, and the last one is used for deletion-error correction. The total data blocklength used in the paper is 1000 nucleotides; 984 of them is for encoded information data, while 16 nucleotides are used for addressing. As mentioned earlier, the addresses are designed to have large mutual Hamming distances and low mutual correlations. This was done by combining together three codewords, two from a code $\mathcal{C}_1$ and one from a second code $\mathcal{C}_3$. In the paper, the authors chose $\mathcal{C}_1 = \{10000000, 01111111\}$, while $\mathcal{C}_3$ is a BCH code of length 16, dimension 11, and minimum distance 4. Furthermore, the addresses should not appear as substrings in the encoded information; as a result, the encoded information sequences (984 nucleotides) are constructed 8 nucleotides at a time by combining codewords from codes $\mathcal{C}_2$ and $\mathcal{C}_4$, each of length-8 bits, where $\mathcal{C}_2$ is chosen to be the set of all balanced binary sequences of length-8 bits (contains an equal number of 1's and 0's), while $\mathcal{C}_4$ is the set of all binary words of length 8.

After sequencing, the scheme performs three post-processing steps to recover the original data. First, a rough estimate of the DNA codeword is constructed, where it contains a list of the most reliable DNA strands or fragments (the ones that contain the least of number of IDS errors). This is done by using the addresses as pilot symbols since they are known to both the transmitter and the receiver. The strands corresponding to the pilot symbols with the least errors are classified as the highest-quality reads. Second, the authors use multiple sequence alignment (MSA) algorithms to construct several consensus (majority) sequences. These algorithms are performed on the high-quality reads decided earlier. The different reads are compared, and a consensus sequence is chosen with a Burrows-Wheeler alignment algorithm [25]. This final step is repeated several times, each time more reads are added to decide on the consensus until all poor-quality reads are solved or declared unsolvable. The above steps will, in some way, transform all unsolved substitution and insertion errors into deletion errors. The authors build on several codes for deletion errors in the literature to develop their coding scheme, referred to as the "homopolymer parity-check" code.

2.3.8 Chanda *et al.*

Unlike the previously mentioned experiments, where the main performance metric is the spectral efficiency in bits/nts, Chanda *et al.* [13] target two metrics referred to as the writing cost, denoted by c_{w}, and the reading cost, denoted by c_{r}, both with units of nts/bits. The writing cost is the reciprocal of the spectral efficiency, and the reading cost is loosely defined as the number of reads that need to be performed to guarantee recovery of the original data. In this work, the authors investigate the trade-off between c_{w} and c_{r} and compare their proposed scheme to previous works in this regard.

To this end, the proposed scheme works as follows. First, a binary file of size 224 KB is encoded using an LDPC code (see Chapter 3.3), where the resulting codeword is divided into $L = 12,026$ blocks, each of the same size. Each block is then mapped to the DNA alphabet, using the standard mapping of $00 \rightarrow A$, $01 \rightarrow C$, $10 \rightarrow G$, and $11 \rightarrow T$. The authors randomize the input binary file by compressing and encrypting it to avoid long runs of homopolymers [13]. Next, a marker of 3 bases, "AGT," known to both the transmitter and receiver, is added to each block. This is done to help in correcting some insertion and deletion errors after sequencing. Then, an index is attached to every block, where the index is protected by a BCH code. The DNA blocks are then synthesized into DNA strands and stored. Each DNA strand is made up of 100 nucleotides. The index takes 10 nucleotides and is protected by a BCH code that takes 6 nucleotides. The payload will take the rest of the 84 nucleotides, where the marker "AGT" takes 3 of them.

The decoding of the reads is done as follows. First, index decoding for each read is performed by decoding the BCH code. If the decoding fails, then an attempt to solve for a single insertion or deletion in the index is done. The recovery succeeds when a unique single insertion or deletion error leads to a noiseless BCH codeword. According to the authors, this additional step recovers 5-10% of the indices [13]. After index decoding, the decoder groups the reads belonging to the same index and invoke an MSA algorithm to decide on a consensus for each index. Afterward, the consensus sequence is checked for the correct target length. If the consensus length is not equal to the target one, then the decoder uses the marker sequence to recover the read. The MSA algorithm transforms insertion and deletion errors into substitution errors. Finally, the MSA consensus is sent to the LDPC decoder, where the decoder first computes the probability of each bit and then invokes a decoding algorithm. According to [13], the proposed scheme achieves a writing cost of $c_{\mathrm{w}} = 0.67$ and a reading cost of $c_{\mathrm{r}} = 3.82$.

2.3.9 Discussion

Although all these experiments show the full error-free reconstruction of their user data, they build their schemes based on heuristics only. Many of the experiments involved discarding sequences that do not fit the required length or included single insertion/deletion-correcting codes on top of substitution-correction codes that only recover a very small subset of sequences. None of these works provide a more theoretical study of the DNA storage channel (although this literature exists) nor offer a fundamental understanding of the impact of synchronization errors on coding scheme performances. In this thesis, we first offer a fundamental understanding of IDS errors and provide insight into how they hinder performance. In addition, we propose coding schemes and decoding algorithms that are tailored to handle IDS errors and the inherited multi-copy redundancy of the DNA storage channel. Furthermore, we build up to and propose a realistic channel model that captures most of the aspects of a DNA-based storage system, along with a coding scheme that outperforms all of the aforementioned experiments in most metrics.

Chapter 3

Modern Coding Theory

In this chapter, we provide preliminaries of coding theory and introduce modern codes and tools to predict their performance. Modern codes have become the standard for many applications in communications, which is attributed to their excellent performance in several channels. They have been proven to achieve the information rate capacity theorized by Shannon [26, 27], a milestone that coding theorists have sought for a long time. Hence, modern codes are used extensively in this thesis. Two classes of well-known modern codes are LDPC codes and turbo codes. Turbo codes are usually made up of a parallel or serial concatenation of two convolutional codes. LDPC and convolutional codes are both used in this thesis.

3.1 Preliminaries of Coding Theory

In the context of communication theory, which offers a mathematical model for information transfer over various channels and sets a benchmark for designing communication systems, coding theory works under the principle of adding redundancy to the transmitted message, where this redundancy is exploited in the receiver to correct errors that impaired the message. Formally, we define a code as a mapping from a vector space of dimension k, $\mathbb{F}_q^k$, to a vector space of dimension n, $\mathbb{F}_q^n$, over a finite field $\mathbb{F}_q$ of size q, where $n > k$. Formally, a code $\mathcal{C}(k, n)$ will take an information vector, or message, $\boldsymbol{u}$ of length k, and map it to a codeword $\boldsymbol{c}$ of length n. The span of $\boldsymbol{u}$ is the entire vector space $\mathbb{F}_q^k$, and we can have q^k possible messages. The mapping is one-to-one, so we have q^k codewords out of the possible q^n vectors in $\mathbb{F}_q^n$. The code rate is defined as $R = \frac{k}{n}$, and it represents the degree of redundancy the code has; the closer to one the rate is, the less redundancy the code uses.

An important and widely used subclass of codes is *linear block codes*, which exhibit very nice properties. A code is said to be a linear block code if its q^k codewords are a subspace of $\mathbb{F}_q^n$. An encoder of a linear block code $\mathcal{C}(k, n)$ will take a message $\boldsymbol{u}$ of length k and multiply it by a corresponding $k \times n$ generator matrix $\boldsymbol{G}$. The generator matrix $\boldsymbol{G}$ consists of k linearly independent row vectors that span $\mathcal{C}(k, n)$. The multiplication of the message $\boldsymbol{u}$ and $\boldsymbol{G}$ will result in a codeword $\boldsymbol{c}$ of length n, i.e., $\boldsymbol{c} = \boldsymbol{u}\boldsymbol{G}$. A linear block code has the following properties.

- The addition over $\mathbb{F}_q^n$ of any two codewords in $\mathcal{C}(k, n)$ is another codeword.

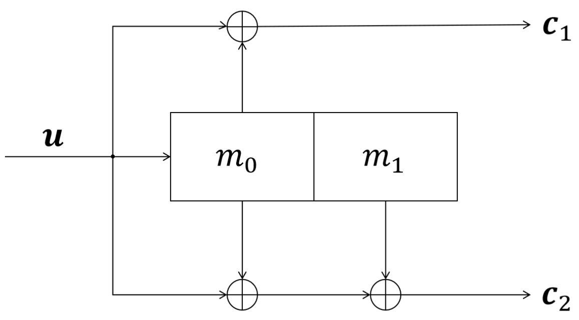


Figure 3.1: FSM representation of a convolutional code encoder.

Formally, for $\boldsymbol{c}_i \in \mathcal{C}$ and $\boldsymbol{c}_j \in \mathcal{C}$, we have $\boldsymbol{c} = \boldsymbol{c}_i + \boldsymbol{c}_j \in \mathcal{C}$.

- All linear block codes contain the all-zero codeword, i.e., $\mathbf{0} \in \mathcal{C}$.

Furthermore, a linear block code $\mathcal{C}$ has a dual (null) space mapping, $\mathcal{C}_{\text{null}}$, from $\mathbb{F}_q^{n-k}$ to $\mathbb{F}_q^n$. The code corresponding to the dual space is referred to as the dual code of $\mathcal{C}$, and has a generator matrix $\boldsymbol{H}$, of dimensions $(n-k) \times n$, referred to as the parity-check matrix of $\mathcal{C}$, where $\boldsymbol{G}\boldsymbol{H}^T = \mathbf{0}$ and $(\cdot)^T$ denotes the transpose of its argument.

3.2 Convolutional Codes

Convolutional codes are another example of popular linear codes. They have been used in several applications and standardization. Introduced in 1955 by Peter Elias, it was later shown that convolutional codes can be maximum likelihood (ML) and maximum a posteriori probability (MAP) decoded efficiently [28, 29], where soft output from the decoder is provided, unlike most linear block codes.

Formally, a convolutional code is denoted as $(n_{\text{cc}}, k_{\text{cc}}, \nu_{\text{cc}})$, where n_{cc} is the number of output bits given k_{cc} input bits to the encoder; hence, $n_{\text{cc}} > k_{\text{cc}}$. Furthermore, ν_{cc} denotes the number of memory elements in the convolutional code. The convolutional code encoder is typically represented by a finite state machine (FSM) made up of shift registers and XOR gates. The encoder of a $(2, 1, 2)$ convolutional code depicted as an FSM is shown in Fig. 3.1. The rate of this convolutional code is $R_{\text{cc}} = k_{\text{cc}}/n_{\text{cc}} = 1/2$, where every bit in $\boldsymbol{u} = (u_1, u_2, \dots)$ is encoded into the odd bits in $\boldsymbol{c}_1 = (c_1, c_3, \dots)$ and the even bits in $\boldsymbol{c}_2 = (c_2, c_4, \dots)$, giving us the overall codeword $\boldsymbol{c} = (c_1, c_2, c_3, c_4, \dots)$. For clarification, here $c_1 = u_1 \oplus m_0$ and $c_2 = u_1 \oplus m_0 \oplus m_1$ for the first round of encoding, then the memory elements will shift the data one bit to the right, i.e., $m_1 = m_0$ followed by $m_0 = u_1$. In general, for this convolutional code, $c_{2j-1} = u_j \oplus m_0$ and $c_{2j} = u_j \oplus m_0 \oplus m_1$. The FSM can be generalized to $k_{\text{cc}} > 1$ input bits, $n_{\text{cc}} \geq 2$ output bits, and $\nu_{\text{cc}} \geq 2$ memory elements and for all combinations of XOR operations. Furthermore, convolutional codes are known to be stream-like codes, where essentially the information bits from $\boldsymbol{u}$ can keep being fed continuously in a stream-like fashion, and the FSM will keep producing encoded bits. However, in practice, the stream of information is finite, and the convolutional code is used as a block code. As a result, the convolutional code needs to be terminated at the end of the transmission by clearing the memory elements. This procedure will be shown later in this section.

3.2.1 Algebraic Representation of Convolutional Codes

One common way to represent convolutional codes is through polynomials. Formally, define the polynomial ring $\mathbb{F}_2[D]$ with indeterminate D, which contains polynomials with coefficients from $\mathbb{F}_2$. An $(n_{\mathrm{cc}}, k_{\mathrm{cc}}, \nu_{\mathrm{cc}})$ convolutional code has a polynomial generator matrix of the form

$$\boldsymbol{G}(D) = \begin{pmatrix} g_{1,1}(D) & g_{1,2}(D) & \dots & g_{1,n_{\mathrm{cc}}}(D) \\ g_{2,1}(D) & g_{2,2}(D) & \dots & g_{2,n_{\mathrm{cc}}}(D) \\ \vdots & \vdots & \dots & \vdots \\ g_{k_{\mathrm{cc}},1}(D) & g_{k_{\mathrm{cc}},2}(D) & \dots & g_{k_{\mathrm{cc}},n_{\mathrm{cc}}}(D) \end{pmatrix} \in \mathbb{F}_2[D]^{k \times n},$$

where ν_{cc} is the sum of the maximum degrees of the polynomials in each row, i.e., $\nu_{\mathrm{cc}} = \sum_{i=1}^{k} \max_{1 \le j \le n} \deg(g_{i,j}(D))$. Here, the coefficient of D^0 in $g_{i,j}(D)$ represents the connection of the j-th output stream to the input bit of the i-th input stream and D^l, $1 \le l \le \deg(g_{i,j}(D))$, the connection to the l-th memory element of the corresponding set of memory elements. Then, the polynomial representation of the codeword, denoted by $\boldsymbol{c}(D) = (c_1(D), c_2(D), \dots, c_{n_{\mathrm{cc}}}(D))$, is obtained by multiplying $\boldsymbol{G}(D)$ with the polynomial representation of the input vector, denoted by $\boldsymbol{u}(D) = (u_1(D), u_2(D), \dots, u_k(D))$, i.e., $\boldsymbol{c}(D) = \boldsymbol{u}(D)\boldsymbol{G}(D)$.

As an example, going back to Fig. 3.1, we have $\boldsymbol{G}(D) = (g_1(D), g_2(D))$, where $g_1(D) = 1 + D$ and $g_2(D) = 1 + D + D^2$.

3.2.2 Decoding of Convolutional Codes

The two most standard ways to decode convolutional codes are the Viterbi algorithm [28] and the BCJR algorithm [29], which are algorithms used for ML and MAP decoding, respectively. Both algorithms use the trellis representation of a convolutional code, taking advantage of the Markovian property of the FSM representation of the convolutional code encoding. A convolutional code trellis will be made of all possible combinations of the memory states at all time instances. Transmission of coded bits to the channel can be thought of as information bits entering the FSM, one by one, in a queue fashion. In this way, each information bit will take up one time instant of the encoding process. Thus, in most cases, the trellis will have identical copies of all possible memory states for each time instant. Furthermore, based on the allowed state transitions in the FSM, the memory states between each time instant and the previous one will be connected by edges following those state transitions.

The trellis representation of the convolutional code in Fig. 3.1 is shown in Fig. 3.2. The trellis is made up of four states, which represent all possible combinations of the memory elements of the convolutional encoder. One trellis section will comprise all previous states, all next states, and all possible transitions between

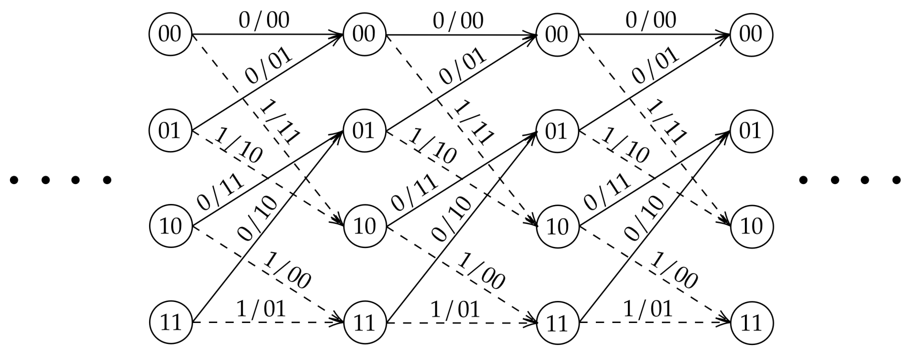


Figure 3.2: Trellis representation of the convolutional code encoder in Fig. 3.1. This trellis can be used for efficient MAP and ML decoding of the convolutional code.

them. If we take Fig. 3.2 for example, the previous state (00) can transition into either state (00) or (10) and so on. The first transition happens when a 0 is an input to the FSM, while the other transition happens when a 1 is an input. For this case, only two possible state transitions can happen per state. Also, each edge connecting one state to another will have a unique label. For example, a transition from state (00) to state (10) contains an edge labeled with 1/11. This means the FSM was in state (00), and then an input bit of 1 was encoded to 11, resulting in the FSM transitioning to state (10). As mentioned before, this trellis can be used to efficiently decode the convolutional code, as will be shown later. One key advantage of representing a convolutional code with an FSM and a trellis is that the Markovian property of the code is more apparent. With this representation, we can define a hidden Markov model (HMM) of the convolutional code encoding process and, in turn, use this HMM and its Markovian property to decode. The HMM corresponding to the convolutional code will have a state variable we denote by $\sigma_i = s_i$, where s_i denotes the state variables of the convolutional code, and i is the corresponding trellis section or time instant. For the convolutional code in Fig. 3.1, $s_i \in \{(00), (01), (10), (11)\}$ as stated earlier. In the resulting HMM, with input $\boldsymbol{u} = (u_1, u_2, \ldots, u_k)$ and output $\boldsymbol{c} = (c_1, c_2, \ldots, c_n)$, a transition from time $i-1$ to time i corresponds to a transmission of symbols $\boldsymbol{c}_{(i-1)n/k+1}^{in/k}$, where $\boldsymbol{c}_a^b = (c_a, c_{a+1}, \ldots, c_b)$.[1]

In this thesis, we mainly perform MAP decoding of our coding scheme. Hence, for a given input sequence $\boldsymbol{u}$ to an encoder and a received sequence $\boldsymbol{y}$, of length n, from the channel, we need to compute the APP of each information symbol u_i in $\boldsymbol{u}$. To this end, the a posteriori probability (APP) of symbol u_i can be computed as

$$p(u_i|\boldsymbol{y}) = \frac{p(\boldsymbol{y}, u_i)}{p(\boldsymbol{y})}.$$

[1]For simplicity, we assume here that k is a divisor of n. Otherwise, the number of symbols that are transmitted from time $i-1$ to time i will vary with i such that the average number of transmitted symbols equals n/k.

The joint probability $p(\boldsymbol{y}, u_i)$ can be computed by de-marginalizing with respect to the memory states of the convolutional code corresponding to symbol u_i. We can use the trellis and the HMM representation to marginalize the convolutional code states with respect to symbol u_i as follows,

$$p(\boldsymbol{y}, u_i) = \sum_{(\sigma,\sigma'):u_i} p(\boldsymbol{y}, \sigma, \sigma'),$$

where σ and σ' denote realizations of the random variables σ_{i-1} and σ_i, respectively. Using the Markov property, we can expand the joint probability $p(\boldsymbol{y}, \sigma, \sigma')$ into three parts as

$$p(\boldsymbol{y},\sigma,\sigma') = p\left(\boldsymbol{y}_1^{i-1}, \sigma\right) p\left(y_i, \sigma' \middle| \sigma\right) p\left(\boldsymbol{y}_i^n \middle| \sigma'\right).$$

Abbreviating the above terms by $\alpha_{i-1}(\sigma)$, $\gamma_i(\sigma, \sigma')$, and $\beta_i(\sigma')$ in order of appearance, one can deduce the forward and backward recursions

$$\begin{aligned} \alpha_i(\sigma') &= \sum_{\sigma} \alpha_{i-1}(\sigma)\gamma_i(\sigma, \sigma'), \\ \beta_{i-1}(\sigma) &= \sum_{\sigma'} \beta_i(\sigma')\gamma_i(\sigma, \sigma'). \end{aligned} \tag{3.1}$$

With convolutional codes, the initial state of the memory elements is always set to the all-zero state. Thus, in decoding, the trellis starts in this initial state before expanding into the form shown in Fig. 3.2. However, since convolutional codes are typically not used in a stream-like fashion, the transmission has to come to an end, where the convolutional code needs to be terminated. The termination is done by forcing the memory elements to the chosen final state after transmitting the last symbol u_k, where extra bits will be sent to the FSM. The total number of termination bits equals the number of memory elements, and their values are chosen such that the convolutional code will reach the chosen final state. The all-zero state is usually also chosen for that. Hence, knowing the initial and final states of the convolutional code, the initial and termination conditions of the forward and backward recursions, respectively, are

$$\alpha_0(\sigma) = \begin{cases} 1 & \text{if } \sigma = (0, 0, \ldots, 0) \\ 0 & \text{otherwise,} \end{cases}$$

$$\beta_{k+\nu}(\sigma) = \begin{cases} 1 & \text{if } \sigma = (0, 0, \ldots, 0) \\ 0 & \text{otherwise.} \end{cases}$$

The branch metric can be decomposed as

$$\gamma_i(\sigma, \sigma') = p(u_i)p(y_i, \sigma'|\sigma),$$

where $p(u_i)$ is the a priori probability of symbol u_i. All the above recursions/equations can be computed/implemented efficiently using the BCJR algorithm [29].

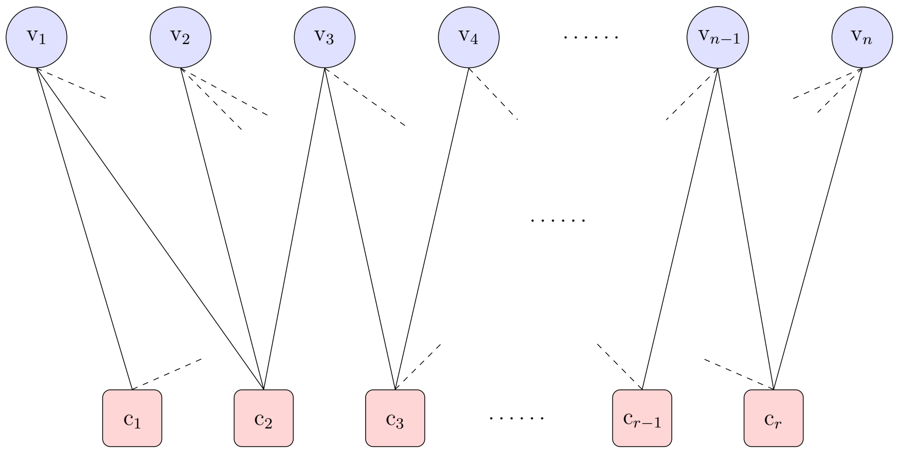


Figure 3.3: Bipartite graph representation of an LDPC code with n VNs and r CNs.

3.3 Low-Density Parity-Check Codes

LDPC codes are characterized by their LDPC matrix $\boldsymbol{H}$. That is, all rows and columns of $\boldsymbol{H}$ have at most J non-zero entries where J is very small relative to the code length n. In this case, $\boldsymbol{H}$ is said to be sparse.

A parity-check matrix $\boldsymbol{H}$ can be represented by a bipartite (Tanner) graph made up of $n-k=r$ check nodes (CNs), corresponding to the rows in $\boldsymbol{H}$, and n variable nodes (VNs) corresponding to the columns in $\boldsymbol{H}$. The CNs and VNs make up the two sets of vertices in a bipartite graph. An edge will be connected between the i-th CN, denoted as c_i, where $1 \leq i \leq r$, and the j-th VN, denoted as v_j, where $1 \leq j \leq n$, for all non-zero entries in $\boldsymbol{H}$, i.e., $h_{i,j} \neq 0$. To this end, define the neighborhood of CN c_i as the set of all VNs connecting to it, denoted as $\mathcal{N}(\mathrm{c}_i) = \{\mathrm{v}_j : h_{i,j} \neq 0, 1 \leq j \leq n\}$. The same can be defined for the VNs where $\mathcal{N}(\mathrm{v}_j) = \{\mathrm{c}_i : h_{i,j} \neq 0, 1 \leq i \leq r\}$. Finally, the total number of neighbors for a VN v_j or a CN c_i is referred to as the degree of the node, denoted as d_{v_j} and d_{c_i}, respectively. Fig. 3.3 shows a bipartite graph representation of an LDPC code. This graph representation is essential for efficient decoding of an LDPC code, as will be shown later. An LDPC code is referred to as a regular code if all CNs and VNs have the same degree d_{c} and d_{v}, respectively, i.e., $d_{\mathrm{v}_j} = d_{\mathrm{v}}$, $1 \leq j \leq n$, and $d_{\mathrm{c}_i} = d_{\mathrm{c}}$, $1 \leq i \leq r$. Otherwise, the LDPC code is referred to as an irregular code.

3.3.1 Decoding of LDPC Codes (Belief Propagation)

Decoding of LDPC codes leverages the sparsity of the parity-check matrix. The decoder uses the corresponding Tanner graph to iteratively pass messages of extrinsic beliefs or likelihoods of the encoded symbols based on the received symbols from the channel. Although using this message-passing technique on the Tanner graph is suboptimal, it is practical and can be done efficiently. The Gallager sum-product algorithm (SPA), which is the main decoder used for LDPC codes, will be described in the following. We first describe the decoder for binary

LDPC codes and then generalize the algorithm to non-binary LDPC (NB-LDPC) codes. The derivations of the equations can be found in the literature, e.g., in [30].

The Gallager SPA Decoder

The decoder receives from the channel the vector $\boldsymbol{y} = (y_1, y_2, \dots, y_n)$, which can be considered as an erroneous version of the transmitted codeword $\boldsymbol{c} = (c_1, c_2, \dots, c_n)$ unknown to the receiver. For the binary case, a codeword bit can either be a 0 or 1, i.e., $c_j = 0$ or 1. Hence, we can define the log-likelihood ratio (LLR) of a transmitted bit c_j, given the received symbol y_j, as

$$L_j = \log\left(\frac{\Pr(c_j = 0|y_j)}{\Pr(c_j = 1|y_j)}\right). \tag{3.2}$$

To this end, the SPA algorithm works as shown in Algorithm 1. Here, $L^{(\ell)}_{\mathrm{v}\to\mathrm{c}}$ and $L^{(\ell)}_{\mathrm{c}\to\mathrm{v}}$ denote the LLR message passed from VN v to CN c and from CN c to VN v in the ℓ-th iteration round, respectively, and $\ell_{\max}$ is the maximum number of iterations.

In Steps 2 and 3 in Algorithm 1, we notice that the message sent to a VN or a CN is the aggregate (product or sum) of all messages received from the neighbors of the current VN or CN, except for the node that the message is to be transmitted to. This is because the target CN or VN already contains that information itself, and it would be redundant to receive the same information again. In some cases, this may even hinder the performance. Hence, the decoding algorithm makes sure that only extrinsic messages are being passed. In other words, each node will receive new information in each iteration without unnecessary redundancy.

Moving on, in this thesis, we mostly deal with NB-LDPC codes. The NB-LDPC codeword symbols are taken from $\mathbb{F}_q$, where $q > 2$. In turn, the non-zero entries of the parity-check matrix of an NB-LDPC code are also elements in $\mathbb{F}_q$. It is more convenient to work in the probability domain when decoding NB-LDPC codes; hence, the decoder will pass message probabilities. It also follows that the received vector will have elements in $\mathbb{F}_q$, and we define the APP vector for symbol c_j as

$$\boldsymbol{P}_j = \begin{pmatrix} \Pr(c_j = 0|y_j) \\ \Pr(c_j = 1|y_j) \\ \vdots \\ \Pr(c_j = q-1|y_j) \end{pmatrix} = \begin{pmatrix} P_j^0 \\ P_j^1 \\ \vdots \\ P_j^{q-1} \end{pmatrix}, \tag{3.3}$$

for all j. Now, the SPA decoder can be generalized to NB-LDPC codes as shown in Algorithm 2. Similar to the binary case, $\ell_{\max}$ is the maximum number of iterations, ℓ is the iteration number, and $\boldsymbol{P}^{(\ell)}_{\mathrm{v}\to\mathrm{c}}$ and $\boldsymbol{P}^{(\ell)}_{\mathrm{c}\to\mathrm{v}}$ denote the probability vector messages passed from VN v to CN c and from CN c to VN v in the ℓ-th iteration round, respectively.

Step 2 of Algorithm 2 is a convolution of the probability density functions (PDFs) of the messages, which can be computationally heavy. In [30], Step 2 can be done by applying a number-theoretic transform, which is the finite field arithmetic version of the fast Fourier transform (FFT), to the message probability vectors and multiplying them. The multiplication result is then transformed back by applying the inverse FFT.

Algorithm 1: The Gallager SPA Decoding Algorithm for Binary LDPC Codes

1. **Initialization:** Set the iteration counter to $\ell = 0$. Compute all the LLRs from the channel as in (3.2) for all j, and then compute all VN to CN messages in the first iteration as
$$L^{(0)}_{\mathrm{v}_j \to \mathrm{c}_i} = L_j, \ \forall j \text{ and } \forall \mathrm{c}_i \in \mathcal{N}(\mathrm{v}_j).$$
2. **CN update:** Increment ℓ by 1. Compute outgoing CN LLR messages, $L^{(\ell)}_{\mathrm{c}_i \to \mathrm{v}_j}$, for each CN c_i and VN $\mathrm{v}_j \in \mathcal{N}(\mathrm{c}_i)$ using
$$L^{(\ell)}_{\mathrm{c}_i \to \mathrm{v}_j} = 2\tanh^{-1}\left(\prod_{\mathrm{v} \in \mathcal{N}(\mathrm{c}_i) \setminus \{\mathrm{v}_j\}} \tanh\left(\frac{1}{2} L^{(\ell-1)}_{\mathrm{v} \to \mathrm{c}_i}\right)\right).$$
3. **VN update:** Compute outgoing VN LLR messages, $L^{(\ell)}_{\mathrm{v}_j \to \mathrm{c}_i}$, for each VN v_j and CN $\mathrm{c}_i \in \mathcal{N}(\mathrm{v}_j)$ using
$$L^{(\ell)}_{\mathrm{v}_j \to \mathrm{c}_i} = L_j + \sum_{\mathrm{c} \in \mathcal{N}(\mathrm{v}_j) \setminus \{\mathrm{c}_i\}} L^{(\ell)}_{\mathrm{c} \to \mathrm{v}_j}.$$
4. **Bit decision:** For $j = 1, 2, \ldots, n$, decide on bit c_j by first computing the overall LLR as
$$L^{\mathrm{total}}_j = L_j + \sum_{\mathrm{c}_i \in \mathcal{N}(\mathrm{v}_j)} L^{(\ell)}_{\mathrm{c}_i \to \mathrm{v}_j},$$
where the bit is decided as
$$\hat{c}_j = \begin{cases} 1 & \text{if } L^{\mathrm{total}}_j < 0 \\ 0 & \text{otherwise.} \end{cases}$$
5. **Stopping criteria:** Compute the syndrome of $\hat{\boldsymbol{c}} = (\hat{c}_1, \ldots, \hat{c}_n)$. If $\hat{\boldsymbol{c}}\boldsymbol{H}^{\mathrm{T}} = \boldsymbol{0}$ or $\ell = \ell_{\max}$, then stop and the decided codeword is $\hat{\boldsymbol{c}}$; else, go to Step 2.

3.3.2 Protograph-Based Representation of LDPC Codes

One of the simpler ways to construct LDPC codes is through LDPC protographs. Formally, a protograph is a small multi-edge-type graph with n_{p} VN types and r_{p} CN types. A protograph can be represented by a base matrix
$$\boldsymbol{B} = \begin{pmatrix} b_{1,1} & b_{1,2} & \ldots & b_{1,n_{\mathsf{p}}} \\ b_{2,1} & b_{2,2} & \ldots & b_{2,n_{\mathsf{p}}} \\ \vdots & \vdots & \ldots & \vdots \\ b_{r_{\mathsf{p}},1} & b_{r_{\mathsf{p}},2} & \ldots & b_{r_{\mathsf{p}},n_{\mathsf{p}}} \end{pmatrix},$$
where entry $b_{i,j}$ is an integer representing the number of edge connections from a type-i CN to a type-j VN. A parity-check matrix $\boldsymbol{H}$ of an LDPC code can then be constructed by lifting the base matrix $\boldsymbol{B}$ by replacing each non-zero (zero) $b_{i,j}$ with a $Q_{\mathsf{p}} \times Q_{\mathsf{p}}$ (zero) matrix with row and column weight equal to $b_{i,j}$. Circulant

Algorithm 2: The Gallager SPA Decoding Algorithm for NB-LDPC Codes

1. **Initialization:** Set the iteration counter to $\ell = 0$. Compute all the probability vectors from the channel as in (3.3) for all j, and then compute all VN to CN messages in the first iteration as

$$\boldsymbol{P}^{(0)}_{\mathrm{v}_j \to \mathrm{c}_i} = \boldsymbol{P}_j, \ \forall j \text{ and } \forall \mathrm{c}_i \in \mathcal{N}(\mathrm{v}_j).$$

2. **CN update:** Increment ℓ by 1. For each $a \in \mathbb{F}_q$, the outgoing message $P^{a,(\ell)}_{\mathrm{c}_i \to \mathrm{v}_j}$ from CN c_i to VN $\mathrm{v}_j \in \mathcal{N}(\mathrm{c}_i)$ is computed as

$$P^{a,(\ell)}_{\mathrm{c}_i \to \mathrm{v}_j} = \sum_{\hat{\boldsymbol{c}}:\hat{c}_j = a} \Pr\left(s_i = 0 | \hat{\boldsymbol{c}}\right) \cdot \prod_{\mathrm{v}_t \in \mathcal{N}(\mathrm{c}_i) \setminus \{\mathrm{v}_j\}} P^{h_{i,t},(\ell-1)}_{\mathrm{v}_t \to \mathrm{c}_i},$$

where s_i is the i-th element of the syndrome vector $\boldsymbol{s} = \hat{\boldsymbol{c}}\boldsymbol{H}^{\mathrm{T}}$. By computing the above for each a, we get $\boldsymbol{P}^{(\ell)}_{\mathrm{c}_i \to \mathrm{v}_j}$ for each CN c_i and VN $\mathrm{v}_j \in \mathcal{N}(\mathrm{c}_i)$.

3. **VN update:** For each $a \in \mathbb{F}_q$, the outgoing message $P^{a,(\ell)}_{\mathrm{v}_j \to \mathrm{c}_i}$ from VN v_j to CN $\mathrm{c}_i \in \mathcal{N}(\mathrm{v}_j)$ is computed as

$$P^{a,(\ell)}_{\mathrm{v}_j \to \mathrm{c}_i} = f_{\mathrm{norm}} \, P^a_j \prod_{\mathrm{c} \in \mathcal{N}(\mathrm{v}_j) \setminus \{\mathrm{c}_i\}} P^{a,(\ell)}_{\mathrm{c} \to \mathrm{v}_j},$$

where f_{norm} is chosen such that $\sum_{a \in \mathbb{F}_q} P^{a,(\ell)}_{\mathrm{v}_j \to \mathrm{c}_i} = 1$. By computing the above for each a, we get $\boldsymbol{P}^{(\ell)}_{\mathrm{v}_j \to \mathrm{c}_i}$ for each VN v_j and CN $\mathrm{c}_i \in \mathcal{N}(\mathrm{v}_j)$.

4. **Symbol decision:** For $j = 1, 2, \ldots, n$, and for each $a \in \mathbb{F}_q$, decide on symbol c_j by first computing the total probability for symbol c_j as

$$P^{a,\,\mathrm{total}}_j = f_{\mathrm{norm}} \, P^a_j \prod_{\mathrm{c}_i \in \mathcal{N}(\mathrm{v}_j)} P^{a,(\ell)}_{\mathrm{c}_i \to \mathrm{v}_j},$$

and then make a decision according to

$$\hat{c}_j = \arg\max_{a \in \mathbb{F}_q} P^{a,\,\mathrm{total}}_j.$$

5. **Stopping criteria:** Compute the syndrome of $\hat{\boldsymbol{c}}$. If $\hat{\boldsymbol{c}}\boldsymbol{H}^{\mathrm{T}} = \boldsymbol{0}$ or $\ell = \ell_{\max}$, then stop and the decided codeword is $\hat{\boldsymbol{c}}$; else, go to Step 2.

matrices are mostly used in the lifting procedure, causing the LDPC code to be quasi-cyclic, which is usually a desired feature. Picking the matrices to lift the protograph is a design parameter, and this can be used to optimize the number and length of the shortest cycles existing in $\boldsymbol{H}$. The higher the minimum cycle length, the better the code performance [31, 32]. Cycles are unavoidable in Tanner graph representations of LDPC codes, and they hurt the performance by getting the decoder stuck in finding a solution.

Moving on, the resulting lifted parity-check matrix of dimensions $Q_{\mathsf{p}}r_{\mathsf{p}} \times Q_{\mathsf{p}}n_{\mathsf{p}}$ defines an LDPC code of length $Q_{\mathsf{p}}n_{\mathsf{p}}$ and dimension at least $Q_{\mathsf{p}}(n_{\mathsf{p}} - r_{\mathsf{p}})$. As mentioned before, we use NB-LDPC codes in this thesis. To construct a non-binary code from the lifted matrix, one can randomly assign non-zero entries from $\mathbb{F}_q$ to the edges of the corresponding Tanner graph.

LDPC protographs can be regular or irregular, as defined earlier for LDPC codes in general. Here, the degrees of all VNs of all types should be equal, and the same goes for the degrees of all CNs of all types. In this case, the protograph is considered regular.

3.3.3 Concatenated Coding Schemes

Concatenated codes are a class of error-correcting codes that involve the combination of two or more simpler codes to enhance overall error-correction performance. The idea is to use one code as an outer code and another as an inner code, creating a concatenated structure. This approach leverages the strengths of each code, with the inner code typically providing strong error-correction capabilities and the outer code handling residual errors that may have escaped the inner code's correction.

The concept of concatenated codes was first introduced in 1965 by Forney in [33] to solve the problem of finding a coding scheme that has vanishing error probability as the code length goes to infinity, and where the decoding complexity only grows linearly with the code length. Later on, Berrou, Glavieux, and Thitimajshima, in 1993, developed a specific type of concatenated codes known as turbo codes [26]. Turbo codes are a groundbreaking form of concatenated codes that have played a pivotal role in advancing error-correction techniques. They consist of two or more identical convolutional codes connected in parallel, with an interleaver placed between them. This unique structure allowed turbo codes to achieve error-correction capabilities close to the theoretical Shannon limit, making them highly efficient in noisy communication channels. To this end, turbo codes do not necessarily have to be made up of two convolutional codes. In principle, any two codes can be made into a concatenated coding scheme; however, that does not always guarantee good performance. The strength of turbo codes or concatenated codes, in general, comes from the fact that the two component codes can share information or, more specifically, can send likelihood estimates of the transmitted sequences in an iterative fashion. This is essentially the two codes helping each other decode the original message with more success as opposed to each code decoding separately. The iterative decoding process is also referred to as turbo decoding.

The idea of combining multiple codes for improved performance has inspired advancements in coding theory. For instance, LDPC codes can be thought of as a serial concatenation of repetition codes with parity-check codes. In this thesis, we mainly consider a serial concatenation of an inner block code or convolutional code with an outer LDPC code (or sometimes a polar code).

3.4 Tools for Modern Codes

One important aspect of modern codes, which can be viewed as a drawback, is the probabilistic nature of the error-correction (decoding) process. Traditionally, codes have a well-defined error-correction capability, which refers to the number of errors a code is guaranteed to correct. Algebraic codes such as RS and BCH codes fall under this category. With modern codes, however, the exact error-correction capability is typically not known. Hence, we need to use performance prediction tools to get an idea of how many errors our code is expected to correct. These tools fall under two regimes: the asymptotic regime and the finite-length regime. In the former, the code blocklength is assumed to be infinite, and we get an exact error-correction capability that can also give an indication for how well the code performs in practice. The second regime gives the expected performance in the finite blocklength regime. All these tools are very useful in studying and analyzing concatenated codes and will also serve as a tool for optimizing the component codes, as we will show later.

3.4.1 Density Evolution of LDPC Codes

Density evolution (DE) is a mathematical technique used to analyze the performance of LDPC codes under iterative decoding algorithms, particularly in the context of error correction in digital communication systems.

DE involves analyzing the evolution of PDFs as they pass through the iterative decoding process. At each iteration, messages are exchanged between VNs and CNs, as described earlier. DE provides a way to track how these message probability densities change as iterations progress. The main idea behind DE is to derive and analyze the CN and VN update equations that describe the evolution of these probability densities over iterations. By doing so, it becomes possible to predict the performance of the LDPC code, specifically the error rate, under different channel conditions. DE is a powerful tool for designing and optimizing LDPC codes. By understanding how the probability densities change during decoding iterations, we can choose the LDPC code parameters and rate to achieve the desired performance characteristics. The performance of the LDPC code studied in DE is an asymptotic average performance; however, it still gives accurate insight into how the LDPC code will behave in the finite blocklength regime.

Density Evolution Equations and Algorithm

We will present DE equations and the algorithm (summarized in Algorithm 3) to optimize an LDPC protograph for some channel conditions for irregular LDPC codes in the following.

First, we need to introduce some notation. Let $f^{(\ell)}_{\mathrm{c}_i\to\mathrm{v}_j}$ and $f^{(\ell)}_{\mathrm{v}_j\to\mathrm{c}_i}$ represent the PDF of the message being passed from a CN of type i to a VN of type j and from a VN of type j to a CN of type i in the ℓ-th iteration round, respectively, and let $\ell_{\max}$ denote the maximum number of iterations. Since DE should mimic the iterative message-passing process, we define VN and CN update equations with respect to message PDFs. The DE CN type update equation at iteration ℓ becomes

$$f^{(\ell)}_{\mathrm{c}_i\to\mathrm{v}_j} = \Gamma^{-1}\left[\mathop{\circledast}_{\mathrm{v}_k\in\mathcal{N}(\mathrm{c}_i)}\left(\Gamma\left[f^{(\ell-1)}_{\mathrm{v}_k\to\mathrm{c}_i}\right]\right)^{\circledast\left(b_{i,k}-\begin{cases}1, & \text{if } k=j\\ 0, & \text{if } k\neq j\end{cases}\right)}\right], \tag{3.4}$$

where $\circledast$ represents the convolution operator and $[\cdot]^{\circledast l}$ denotes the l-fold convolution of its argument, and where Γ signifies the change in density of the message PDFs, as explained in [30], when the CN update in Algorithm 1 is computed.

Similarly, the DE VN type update equation at iteration ℓ becomes

$$f^{(\ell)}_{\mathrm{v}_j\to \mathrm{c}_i} = f_{\mathrm{ch}\to \mathrm{v}_j} \circledast \left[\mathop{\large\circledast}_{\mathrm{c}_k\in\mathcal{N}(\mathrm{v}_j)} \left(f^{(\ell)}_{\mathrm{c}_k\to \mathrm{v}_j}\right)^{\circledast\left(b_{k,j} - \left\{\begin{smallmatrix}1, & \text{if } k=i\\ 0, & \text{if } k\neq i\end{smallmatrix}\right\}\right)} \right], \tag{3.5}$$

where $f_{\mathrm{ch}\to \mathrm{v}_j}$ denotes the j-th message PDF from the channel. Since the VN update equation in Algorithm 1 is the summation of VN LLRs, the resulting PDF of the outgoing message will be the convolution of the corresponding PDFs of the neighbors' incoming messages.

Algorithm 3: Density Evolution for Irregular LDPC Protographs

1. Set the channel parameter α to some nominal value expected to be less than the threshold α^*.
2. **Initialization:** Set the iteration counter to $\ell = 0$. Compute all the channel PDFs $f_{\mathrm{ch}\to \mathrm{v}_j}$ for all j, and then compute the VN type j to CN type i message in the first iteration as $f^{(0)}_{\mathrm{v}_j\to \mathrm{c}_i} = f_{\mathrm{ch}\to \mathrm{v}_j}$ for all j and for all $\mathrm{c}_i \in \mathcal{N}(\mathrm{v}_j)$.
3. **CN update:** Increment ℓ by 1. For a given CN type i, compute the outgoing message PDF $f^{(\ell)}_{\mathrm{c}_i\to \mathrm{v}_j}$ for all j using (3.4).
4. **VN update:** For a given VN type j and corresponding channel PDF $f_{\mathrm{ch}\to \mathrm{v}_j}$, compute the outgoing message PDF $f^{(\ell)}_{\mathrm{v}_j\to \mathrm{c}_i}$ for all i using (3.5).
5. **Decision:** Calculate the average VN type to CN type message PDF at iteration ℓ as $f^{(\ell)}_{\mathrm{v}} = \sum_j \frac{1}{|\mathcal{N}(\mathrm{v}_j)|} \sum_i f^{(\ell)}_{\mathrm{v}_j\to \mathrm{c}_i}$. If $\ell < \ell_{\max}$ and
$$\int_{-\infty}^{0} f^{(\ell)}_{\mathrm{v}}(\tau)\mathrm{d}\tau \le p_e \tag{3.6}$$
for some prescribed error probability p_e (e.g., $p_e = 10^{-6}$), increment the channel parameter α by some small amount and go to Step 2. If (3.6) does not hold and $\ell < \ell_{\max}$, then go back to Step 3. If (3.6) does not hold and $\ell = \ell_{\max}$, then the previous α is the decoding threshold α^*.

Gaussian Approximation of Density Evolution

Practically, equations (3.4) and (3.5) are difficult to compute, and a quantized version is required for efficient implementation [30]. However, the DE of LDPC codes can be approximated, hence, computationally simplified, by considering the messages as Gaussian random variables, hence, having Gaussian PDFs. This approximation is referred to as the *Gaussian approximation* [34]. Since a Gaussian random variable, along with its PDF, is completely characterized by its mean and variance, the DE algorithm only needs to track these two parameters instead of the entire PDF. However, more simplifications can be made to the Gaussian assumption. Suppose the message PDF satisfies the *consistency condition*, which is a good assumption for symmetric channels. In that case, only the mean of the message PDF needs to be tracked as the variance will be directly related to it [30]. Following this assumption, we denote the mean of the PDF of the message from VN type j to CN type i and from CN type i to VN type j in the ℓ-th iteration round by $\mu^{(\ell)}_{\mathrm{v}_j \to \mathrm{c}_i}$ and $\mu^{(\ell)}_{\mathrm{c}_i \to \mathrm{v}_j}$, respectively. As a result, equations (3.4) and (3.5) become

$$\mu^{(\ell)}_{\mathrm{c}_i \to \mathrm{v}_j} = \Phi^{-1}\left(1 - \prod_{k}\left[1 - \Phi(\mu^{(\ell-1)}_{\mathrm{v}_k \to \mathrm{c}_i})\right]^{\left(b_{i,k} - \left\{\begin{smallmatrix} 1, & \text{if } k = j \\ 0, & \text{if } k \neq j \end{smallmatrix}\right\}\right)}\right) \tag{3.7}$$

and

$$\mu^{(\ell)}_{\mathrm{v}_j \to \mathrm{c}_i} = \mu_{\mathrm{ch} \to \mathrm{v}_j} + \sum_{k}\left(b_{k,j} - \left\{\begin{smallmatrix} 1, & \text{if } k = i \\ 0, & \text{if } k \neq i \end{smallmatrix}\right\}\right)\mu^{(\ell)}_{\mathrm{c}_k \to \mathrm{v}_j}, \tag{3.8}$$

respectively, where $\mu_{\mathrm{ch} \to \mathrm{v}_j}$ denotes the mean of the j-th message PDF from the channel and the function $\Phi(\cdot)$ is as defined in [30]. The DE algorithm using the Gaussian approximation follows the same steps as Algorithm 3 with equations (3.7) and (3.8) replacing equations (3.4) and (3.5), respectively. It should be noted that the channel models for DNA-based storage systems introduced in the next chapter are not symmetric; however, we do, in some cases, use the Gaussian approximation for DE, as will be shown later. Although this does not guarantee accurate results, it is satisfactory to give good results.

3.4.2 Extrinsic Information Transfer Charts

Extrinsic information transfer (EXIT) charts analysis is a graphical approach used to study the behavior and convergence properties of iterative decoding algorithms when applied to concatenated coding schemes. This technique utilizes EXIT charts to visually depict the mutual information transfer between the inner code and the outer code components of the decoding process.

In an iterative decoding algorithm, information is exchanged between the inner code and the outer code. The inner code operates on the raw received data and computes an estimate of the transmitted symbols, while the outer code performs higher-level error correction based on these estimates. The EXIT chart illustrates how the mutual information exchanged between the inner and outer decoders changes with each iteration.

The iterative decoding process is represented on the EXIT chart by curves that correspond to the information transfer characteristics of the inner and outer codes. These curves, known as the inner EXIT curve and the outer EXIT curve, illustrate how the mutual information evolves as iterations progress. The point of intersection between the inner and outer EXIT curves is of particular significance, as it represents a potential convergence point where the iterative decoding process would stabilize. Let us take an LDPC code as an example. An LDPC code can be viewed as a concatenation of two sets of codes, one representing the VNs while the other representing the CNs. Then, the extrinsic mutual information being

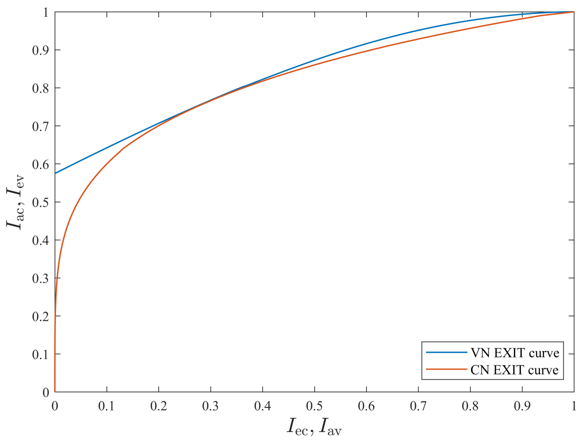
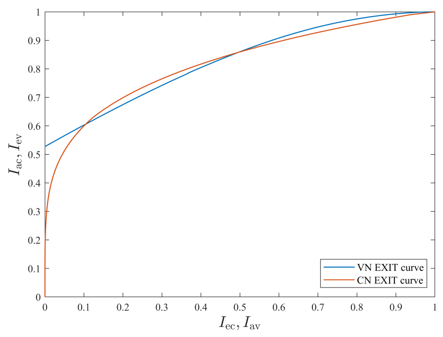


Figure 3.4: CN and VN EXIT curves for a $\boldsymbol{B} = (3, 3)$ LDPC protograph.

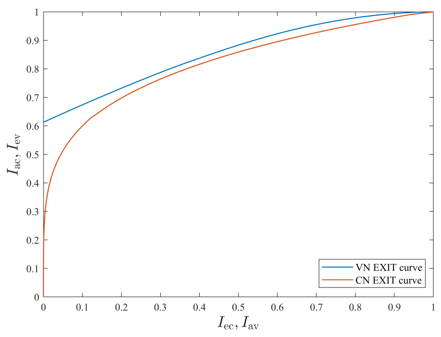
passed to each code during the iterative decoding process, as mentioned earlier with the SPA decoder, can be plotted for each one as shown in Fig. 3.4. In this figure, we impose two plots on top of each other. The first is the EXIT chart of the VNs, showing the a priori mutual information coming to the VNs from the CNs (I_{av}) versus the extrinsic mutual information coming out of the VNs toward the CNs (I_{ev}). The second is the EXIT chart of the CNs, displaying the extrinsic mutual information coming out of the CNs toward the VNs (I_{ec}) against the a priori mutual information coming to the CNs from the VNs (I_{ac}). Hence, we have the two relations, $I_{\mathrm{ac}} = I_{\mathrm{ev}}$ and $I_{\mathrm{av}} = I_{\mathrm{ec}}$.

By analyzing the EXIT chart, one can gain insights into the convergence behavior of the iterative decoding algorithm. The relative positions of the inner and outer EXIT curves and their intersection point provide valuable information about the potential error-correction performance of the code and the algorithm. Using the same LDPC code in Fig. 3.4 as an example, we can see that there is a tunnel between the two curves, which indicates that the iterative decoding algorithm will result in the correct decoding and recovery of the original message. This plot conveys that this LDPC code will achieve zero decoding error probability for this channel condition, given that the code length is infinite (asymptotic regime). However, it can happen that the channel conditions are too extreme for the code to handle, resulting in a non-zero probability of error given infinite length. This will be visible in the EXIT chart plots as a crossing between the EXIT curves of the component codes. On the other hand, the better the channel conditions are, the wider the opening of the tunnel between the curves. Fig. 3.5 depicts these phenomena.

EXIT charts can be used as an optimization tool for coding schemes. The parameters of the inner and outer codes can be adjusted for a given channel condition such that their corresponding EXIT curves will have a tunnel between them, and decoding is guaranteed to be successful asymptotically. In Paper II of this thesis, we optimize an outer LDPC code concatenated with an inner block code through EXIT charts, and the corresponding EXIT curves for a specific



(a) Crossing



(b) Opening

Figure 3.5: CN and VN EXIT curves for an LDPC code based on the protograph $\boldsymbol{B} = (3, 3)$ with bad and good channel conditions, respectively.

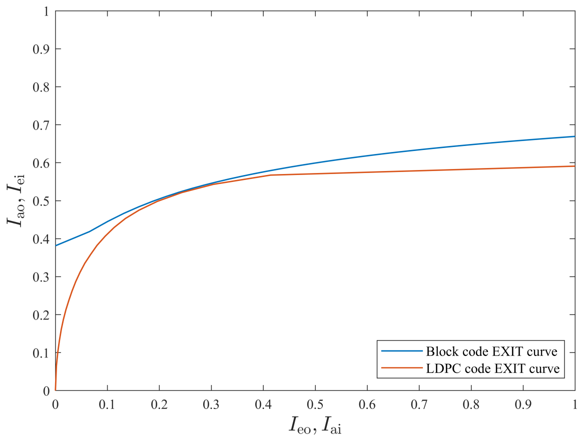


Figure 3.6: EXIT curves of inner and outer codes of Paper II.

scenario are shown in Fig. 3.6.

3.4.3 Achievable Information Rates

Another performance metric for a concatenated coding scheme is what is referred to as an achievable information rate (AIR) for a given inner code. An AIR is an outer code rate that is achievable, i.e., there exists an outer code with this rate such that the overall concatenated code can be decoded with vanishing error probability. By computing an AIR for a given inner code, we have an asymptotic performance benchmark for an outer code perfectly matched to the inner code. In other words, computing an AIR will give us the best coding rate achievable with our chosen inner code and an outer code that is perfectly matched to it. It should be noted that an AIR tells us the existence of an outer code that achieves this coding rate when concatenated with the inner code in the asymptotic regime. Finding this outer code is done by outer code optimization. Having mentioned this, we now present two standard approaches for estimating AIRs over hidden Markov channels, i.e., channels whose output is the output of an HMM. The first approach is to compute the so-called *BCJR-once* rate [35–37], which is defined as the mutual information between the information sequence and its corresponding LLRs, produced by a MAP decoder (BCJR algorithm). The second approach is based on [38, 39] and uses concentration properties of Markov chains to estimate the mutual information between channel input and output.

BCJR-Once Rate

The BCJR-once rate [35–37], which we denote by $R_{\text{BCJR-once}}$, serves as an information rate that can be achieved using an inner decoder that passes once APPs of the information symbols to an appropriate outer decoder that is unaware of possible correlations between the estimates. The outer and inner decoders hereby perform no iterations. The BCJR-once rate can be derived by computing an AIR of a decoder as follows. Let $\boldsymbol{u}$ (of length k) denote the input to the inner code (output of the (non-binary) outer code over $\mathbb{F}_q$) and $\boldsymbol{y}$ the corresponding channel output (of length n), and denote by $q(\boldsymbol{u}|\boldsymbol{y})$ an arbitrary decoding metric that is a valid distribution, i.e., $\sum_{\boldsymbol{u}} q(\boldsymbol{u}|\boldsymbol{y}) = 1$. We obtain for the mutual information between the message $\mathbf{u}$ and the received sequence from the channel $\mathbf{y}$,[2]

$$\begin{aligned} I(\mathbf{u};\mathbf{y}) &= H(\mathbf{u}) - H(\mathbf{u}|\mathbf{y}) \\ &= H(\mathbf{u}) + \sum_{\boldsymbol{u},\boldsymbol{y}} p(\boldsymbol{u},\boldsymbol{y}) \log p(\boldsymbol{u}|\boldsymbol{y}) \\ &= H(\mathbf{u}) + \sum_{\boldsymbol{u},\boldsymbol{y}} p(\boldsymbol{u},\boldsymbol{y}) \left(\log q(\boldsymbol{u}|\boldsymbol{y}) + \log \frac{p(\boldsymbol{u}|\boldsymbol{y})}{q(\boldsymbol{u}|\boldsymbol{y})} \right) \\ &\geq H(\mathbf{u}) + \sum_{\boldsymbol{u},\boldsymbol{y}} p(\boldsymbol{u},\boldsymbol{y}) \log q(\boldsymbol{u}|\boldsymbol{y}), \end{aligned} \tag{3.9}$$

where $H(\cdot)$ denotes the entropy function, and where the last inequality is due to identifying the sum over the second summand as a Kullback-Leibler divergence, which is non-negative. We can then obtain the BCJR-once rate by computing the AIR of a decoder with decoding metric

$$q_{\text{BCJR}}(\boldsymbol{u}|\boldsymbol{y}) = \prod_{i=1}^{k} q(u_i|\boldsymbol{y}).$$

Then, defining the associated LLRs

$$L_i^{\text{BCJR}}(a) = \ln \frac{q(u_i = a|\boldsymbol{y})}{q(u_i = 0|\boldsymbol{y})}, \ \forall\, a \in \mathbb{F}_q,$$

we can combine the decoding metric with the LLRs to obtain

$$q_{\text{BCJR}}(\boldsymbol{u}|\boldsymbol{y}) = \prod_{i=1}^{k} \frac{\mathrm{e}^{L_i^{\text{BCJR}}(u_i)}}{\sum_{a\in\mathbb{F}_q} \mathrm{e}^{L_i^{\text{BCJR}}(a)}},$$

[2]In order to distinguish between random variables and their realizations, $\mathbf{u}$ and $\mathbf{y}$ denote the random variables corresponding to $\boldsymbol{u}$ and $\boldsymbol{y}$, respectively.

where the denominator has been chosen such that we obtain a valid distribution. Plugging the result into (3.9) yields

$$\begin{aligned} I(\mathbf{u};\mathbf{y}) &\geq H(\mathbf{u}) + \sum_{\boldsymbol{u},\boldsymbol{y}} p(\boldsymbol{u},\boldsymbol{y}) \log q_{\mathrm{BCJR}}(\boldsymbol{u}|\boldsymbol{y}) \\ &= H(\mathbf{u}) + \sum_{i=1}^{k} \sum_{u_i,\boldsymbol{y}} p(u_i,\boldsymbol{y}) \log \frac{\mathrm{e}^{L_i^{\mathrm{BCJR}}(u_i)}}{\sum_{a\in\mathbb{F}_q} \mathrm{e}^{L_i^{\mathrm{BCJR}}(a)}}. \end{aligned}$$

For independent and uniform inputs, we obtain $H(\mathbf{u}) = k\log q$. Under the assumption that the expectation above obeys asymptotic ergodicity, we conclude that, for large k, we can estimate the BCJR-once rate by sampling a long input sequence $\boldsymbol{u}$ and corresponding output sequence $\boldsymbol{y}$ and compute the LLRs $L_i^{\mathrm{BCJR}}(a)$, allowing to conclude with the estimate

$$R_{\text{BCJR-once}} \approx R\log q + \frac{R}{k}\sum_{i=1}^{k} \log \frac{\mathrm{e}^{L_i^{\mathrm{BCJR}}(u_i)}}{\sum_{a\in\mathbb{F}_q} \mathrm{e}^{L_i^{\mathrm{BCJR}}(a)}},$$

where $R = k/n$ is the inner code rate.

Mutual Information Rate

A method to compute the mutual information for a given coding scheme was introduced in [38, 39], where the mutual information between an input process $\mathbf{u} = (\mathsf{u}_1, \mathsf{u}_2, \dots)$ and an output process $\mathbf{y} = (\mathsf{y}_1, \mathsf{y}_1, \dots)$, $I(\mathbf{u};\mathbf{y})$, is computed via trellis-based simulations. Here, the input process is the input to an inner code in a concatenated coding scheme, while the output process is the output of an arbitrary channel. Given that a source/channel decoding trellis exists for a coding scheme, the mutual information point

$$I(\mathsf{u};\mathsf{y}) \triangleq \lim_{k\to\infty} \frac{1}{k} I(\mathbf{u};\mathbf{y}),$$

which is an AIR, can be computed using the forward recursion of the BCJR algorithm on the given trellis. Since

$$I(\mathbf{u};\mathbf{y}) = H(\mathbf{u}) + H(\mathbf{y}) - H(\mathbf{u},\mathbf{y})$$

and $-\log p(\boldsymbol{u})$, $-\log p(\boldsymbol{y})$, and $-\log p(\boldsymbol{u},\boldsymbol{y})$ converge with probability 1 to $H(\mathbf{u})$, $H(\mathbf{y})$, and $H(\mathbf{u},\mathbf{y})$, respectively, for long sequences, $I(\mathsf{u};\mathsf{y})$ can be estimated by

$$\hat{I}(\mathsf{u};\mathsf{y}) = -\frac{1}{k}\log p(\boldsymbol{u}) - \frac{1}{k}\log p(\boldsymbol{y}) + \frac{1}{k}\log p(\boldsymbol{u},\boldsymbol{y})$$

when k is very large.

Given an input sequence $\boldsymbol{u}$ to the inner code and a corresponding output sequence $\boldsymbol{y}$ from the channel, $\log p(\boldsymbol{y})$, $\log p(\boldsymbol{u},\boldsymbol{y})$, and $\log p(\boldsymbol{u})$ can be computed as follows. First, compute $\log p(\boldsymbol{y})$ from

$$p(\boldsymbol{y}) = \sum_{\sigma} p\Big(\boldsymbol{y}_1^n, \sigma\Big) \overset{(a)}{=} \sum_{\sigma} \alpha_k(\sigma),$$

where (a) follows since $\alpha_i(\sigma) = p\Big(\boldsymbol{y}_1^{in/k}, \sigma\Big)$.[3] Hence, it can be computed using the recursion in (3.1). Second, $\log p(\boldsymbol{u})$ and $\log p(\boldsymbol{u}, \boldsymbol{y})$ can be computed using a recursion in a similar manner. In particular, the recursion for computing $p(\boldsymbol{u}, \boldsymbol{y})$ is

$$\alpha_i^{(\mathsf{u},\mathsf{y})}(\sigma) = \sum_{\hat{\sigma}} \alpha_{i-1}^{(\mathsf{u},\mathsf{y})}(\hat{\sigma})\gamma_i(\hat{\sigma}, \sigma),$$

where the summation is over all states $\hat{\sigma}$ with an outgoing edge to σ labeled with the input sequence symbol u_i at time i. In other words, the recursion for $p(\boldsymbol{u}, \boldsymbol{y})$ does not marginalize the input sequence $\boldsymbol{u}$ like in $p(\boldsymbol{y})$. Then,

$$p(\boldsymbol{u}, \boldsymbol{y}) = \sum_{\sigma} \alpha_k^{(\mathsf{u},\mathsf{y})}(\sigma).$$

The recursion for computing $p(\boldsymbol{u})$ is

$$\alpha_i^{(\mathsf{u})}(\sigma) = \sum_{\hat{\sigma}} \alpha_{i-1}^{(\mathsf{u})}(\hat{\sigma}) p(u_i, \sigma | \hat{\sigma}),$$

where again the summation is over all states $\hat{\sigma}$ with an outgoing edge to σ labeled with the input sequence symbol u_i at time i. Then,

$$p(\boldsymbol{u}) = \sum_{\sigma} \alpha_k^{(\mathsf{u})}(\sigma).$$

However, since we consider an input sequence of independent and uniformly distributed symbols, $H(\mathbf{u})$ is equal to $k \log q$, and hence we do not need to run the recursion.

The achievable rates $R_{\mathrm{MI}} \approx \hat{I}(\mathsf{u};\mathsf{y})$ give insights on the inner-outer iterative decoding performance of our coding scheme. In other words, this AIR gives us the asymptotic performance of the coding scheme after a large number of inner-outer iterations. This is the reason why there is a difference between mutual information and BCJR-once rates (as will be shown later), and in general $R_{\mathrm{MI}} \geq R_{\text{BCJR-once}}$.

3.4.4 Finite-Length Bounds

As mentioned earlier, the AIRs introduced in the previous subsection can serve as a bound for performance for coding schemes; however, those bounds are asymptotic, i.e., the coding scheme in question will achieve this performance with the blocklength going to infinity. Unfortunately, this will never be the case practically in DNA-based storage systems. As mentioned earlier, the blocklength of a stored DNA strand can range from 100-2000 nucleotides; hence, the asymptotic bounds will fail to capture the exact behavior of coding schemes designed for DNA-based storage. As a result, exploring other bounds, mainly the ones that apply with short and finite-length coding schemes, is helpful. Two finite-length performance bounds we will use in this thesis are the dependency testing (DT) bound and the outage probability bound, where the former uses the computation of R_{MI} and the latter uses $R_{\text{BCJR-once}}$.

[3] As noted previously, for simplicity, we assume here that k is a divisor of n. Otherwise, the number of symbols that are transmitted from time $i-1$ to time i will vary with i such that the average number of transmitted symbols equals n/k.

Dependency Testing Bound

The first finite-length bound provides an upper bound to the frame error probability, denoted by $P_{\mathsf{f}}(e)$, achievable over any given channel in the finite blocklength regime. In particular, we consider the DT bound [40]. The bound we provide is tailored to concatenated coding schemes with inner and outer codes, which is the main code construction we will use throughout this thesis. Hence, it can be used to guide its choice and serves as a benchmark to compare coding schemes.

The DT bound for the combination of an inner code in a concatenated coding scheme (with input length k) operating over an alphabet of size q and the output process from an arbitrary channel is given by

$$P_{\mathsf{f}}(e) \leq \mathbb{E}\left[2^{-\left(i(\mathbf{u};\mathbf{y})-\log_2 \frac{q^k-1}{2}\right)^+}\right], \tag{3.10}$$

where $(x)^+ \triangleq \max(x, 0)$ and

$$i(\mathbf{u};\mathbf{y}) \triangleq \log_2 \frac{p(\mathbf{y}|\mathbf{u})}{p(\mathbf{y})}$$

is the so-called *information density* with expected value equal to the mutual information between $\mathbf{u}$ and $\mathbf{y}$. The distribution of the information density $i(\mathbf{u};\mathbf{y})$ is not known in closed form for many channels. However, the right-hand-side of (3.10) can be accurately estimated using the Monte-Carlo approach proposed in [38, 39], which exploits concentration properties of Markov chains to estimate the mutual information between an input process $\boldsymbol{u}$ and an output process $\boldsymbol{y}$ via trellis-based simulations. We can then approximate (3.10) by

$$P_{\mathsf{f}}(e) \lesssim \frac{1}{V} \sum_{(\boldsymbol{u},\boldsymbol{y})} 2^{-(i(\boldsymbol{u};\boldsymbol{y})-(k\log_2 q-1))^+},$$

where V is the number of pairs $(\boldsymbol{u}, \boldsymbol{y})$ considered in the computation.

Outage Probability Bound

As will be shown in the next chapter, due to strand erasures and replication processes during the storage and sequencing phases, the outer decoder will have some missing sequences to recover. As such, in a way, the DNA storage channel, as seen by the outer code, resembles a block-fading channel when considering a finite number of strands existing in a gene pool [41]. Hence, it also shares its afflictions, most importantly its non-ergodic property. In particular, an *outage* event occurs when not enough (possibly zero) strand copies are drawn from a specific target strand. Formally, an outage event occurs when the instantaneous mutual information between the input and output of an arbitrary channel is lower than the transmission rate R. We consider the BCJR-once version of the information-outage probability adapted from [42], to which we refer as the *information-outage probability* q_{out}. This metric can incorporate any decoding approach, the fixed independent and uniformly distributed input $\boldsymbol{u}$ of length k, and the dispersion due to the finite blocklength phenomena. Let $r_{\text{BCJR-once}}$ denote the instantaneous BCJR-once information density for the finite-length regime, where

$$r_{\text{BCJR-once}} = \frac{1}{k}\sum_{i=1}^{k} I\left(u_i; L_i^{\text{BCJR}}(u_i)\right).$$

Then, q_{out} is formally defined as

$$q_{\text{out}} = \Pr\left(r_{\text{BCJR-once}} < R\right) \geq p_{\text{out}},$$

where p_{out} is the true outage probability. q_{out} gives a lower bound on the frame error rate for a given encoder and decoder pair for a fixed finite blocklength and fixed channel parameters, and can be approximated by the Monte-Carlo method.

Chapter 4

Channel Models for DNA-Based Storage

Designing a good error-correction scheme for a DNA-based storage system requires proper modeling of the DNA storage channel. A DNA storage channel model should capture all essential aspects that affect the DNA strands in storage and retrieval. This includes good modeling of all types of errors that can impair the data stored in DNA. Hence, the channel model should account for the strands having IDS errors. It also should account for the erasure of some strands and the loss of ordering they go through. Having a proper channel model will aid in studying the impact of these errors and will, in turn, benefit the code design. However, studying the effect of all these errors jointly is complex, as several parameters must be considered simultaneously. Hence, it is more efficient and insightful to start with a simplistic model that only considers a subset of these errors, gaining a fundamental understanding of the nature of the channel, then building on top of that to approach a model closer to reality.

4.1 I.i.d. Insertions-Deletions-Substitutions Channel

The simplest way to model errors that arise from DNA-based storage systems is with a channel model that introduces independent and identically distributed (i.i.d.) IDS errors. Studying the impact of the three types of errors, especially insertion and deletion errors, in an idealistic scenario can give great insight and provide a benchmark for designing error-correction schemes tailored to the DNA storage channel. Hence, in this idealistic channel model, clustering and ordering of reads are assumed to be perfect, forcing the model and analysis to be done for IDS errors only. Isolating IDS errors from the other types of errors will provide a simpler framework to gain insight into how to design codes that can handle IDS errors.

Parallels can be drawn between the above idealistic DNA storage scenario and the problem of maintaining synchronization at the receiver side with the transmitter. Codes designed to maintain synchronization are referred to as synchronization error-correction codes in the literature [27, 43–49, 49–56, 56–58]. Hence, we can use code constructions and, more importantly, channel models designed for synchronization errors.

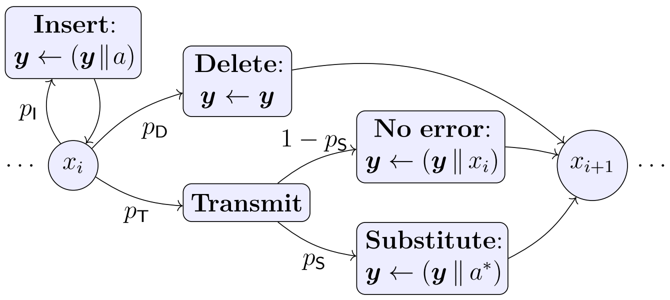

Insert: $\boldsymbol{y} \leftarrow (\boldsymbol{y} \| a)$
Delete: $\boldsymbol{y} \leftarrow \boldsymbol{y}$
No error: $\boldsymbol{y} \leftarrow (\boldsymbol{y} \| x_i)$
Transmit
Substitute: $\boldsymbol{y} \leftarrow (\boldsymbol{y} \| a^*)$
p_I
p_D
p_T
$1 - p_\mathsf{S}$
p_S
$\cdots$ x_i
x_{i+1} $\cdots$

Figure 4.1: Channel model for i.i.d. IDS errors.

To this end, the first work to present a modern coding scheme and channel model incorporating IDS errors is by Davey and MacKay [24]. The channel model introduced in this work is shown in Fig. 4.1. The channel is modeled as an FSM with defined transition probabilities. Here we have the encoded data $\boldsymbol{x} = (x_1, x_2, \ldots, x_i, \ldots x_n)$ of length n to be transmitted over the channel, where each symbol x_i is placed in a queue. When symbol x_i is enqueued, the channel enters state x_i, and three possible events can occur. First, with probability p_I, an insertion event occurs where a uniformly random symbol a is appended to the received sequence. In this case, x_i remains in the queue, and the channel returns to state x_i. Second, symbol x_i is deleted with probability p_D. Consequently, the queued symbol x_i is not appended to $\boldsymbol{y}$, the next symbol x_{i+1} is enqueued, and the channel enters state x_{i+1}. Third, x_i is transmitted with probability $p_\mathsf{T} = 1 - p_\mathsf{I} - p_\mathsf{D}$. In this case, the symbol x_i is substituted with a uniformly random symbol $a^* \neq x_i$ with probability p_S, the next transmit symbol x_{i+1} is enqueued, and the channel enters state x_{i+1}. When the last transmit symbol x_n leaves the queue, the procedure finishes, and the channel outputs $\boldsymbol{y}$. Note that the length of the output sequence n' is random and depends on the probabilities p_D and p_I. In addition, for the most part, when using this channel model, we assume $p_\mathsf{D} = p_\mathsf{I}$ unless stated otherwise. Here, we can define a random variable that incorporates insertions and deletions and leverages the fact that, in most cases, $n' \neq n$. This random variable is introduced in [24] as the *drift*. The drift d_i, $0 \leq i < n$, is defined as the number of insertions minus the number of deletions that occurred before symbol x_i is enqueued, while d_n is defined as the number of insertions minus deletions that occurred after the last symbol x_n has been processed by the channel. It is clear that we only have knowledge of the beginning and end drifts, i.e., $d_0 = 0$ and $d_n = n' - n$, while the values of the drift random variables within the sequence remain unknown. However, drift values within the sequence can be inferred, and the likelihood of the drift value can be calculated. This will be shown later in the papers.

For each time instant, the total drift can either be incremented by one (insertion), decremented by one (deletion), or stay the same (transmission). Hence, the drift random variable can be thought of as performing a random walk. Since, in the channel model, the probabilities of insertions and deletions are typically assumed to be equal, the random walk will have zero mean. In general, the variance of the

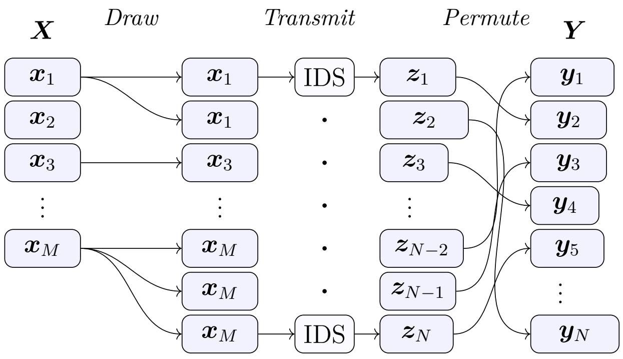


Figure 4.2: Multi-draw IDS channel model.

drift d_n is simply $n \cdot \frac{\max(p_\mathsf{I}, p_\mathsf{D})}{1-\max(p_\mathsf{I}, p_\mathsf{D})}$, giving a standard deviation of $\sqrt{n \cdot \frac{\max(p_\mathsf{I}, p_\mathsf{D})}{1-\max(p_\mathsf{I}, p_\mathsf{D})}}$. The drift, in theory, can go up to infinity, minus or plus, although it is very unlikely. However, from the point of view of the receiver/decoder, it is impractical to have infinite limits on the minimum and maximum drift values. Therefore, when decoding, a finite limit is set. In the papers included in this thesis, the drift limits are usually set to be five to ten times the standard deviation.

In the DNA pool, we will most likely have multiple copies of the target DNA strand. In order to model this feature, we can have M copies of the DNA strand $\boldsymbol{x}$ and transmit them through M parallel i.i.d. IDS channels. As a result, we get multiple noisy copies of the target DNA strand. The receiver/decoder can leverage this redundancy by implementing a multiple-received sequences decoding technique, which is discussed more thoroughly in the papers included in this thesis.

As a final note, the error probabilities can, of course, be set to any desired values, which means that p_I is not necessarily equal to p_D. With this channel model, we assume $p_\mathsf{I} = p_\mathsf{D}$ for simplicity of analysis and understanding.

4.2 Multi-Draw Insertions-Deletions-Substitutions Channel

As previously mentioned, the receiver/decoder has to deal with the loss of ordering of the stored data, along with strand erasures due to some of them not being drawn. In addition, there will exist multiple copies of the target strands in the drawn pool, and therefore the receiver has to cluster the reads accordingly. In the previous model, we mentioned that clustering of reads and ordering them is assumed to be perfect, which practically is not the case. Hence, a DNA storage channel model should also account for errors due to the drawing and shuffling nature of the storage process.

Inspired by [59, 60], the multi-draw IDS channel model is shown in Fig. 4.2. In this channel model, the information or encoded data is split into M blocks of length L, giving us $\boldsymbol{X} \triangleq (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_M)$, which represents the target strands to be synthesized. These synthesized strands then go through the multi-draw IDS channel, where they go through several steps as follows.

1. *Draw:* N draws are performed from the list $\boldsymbol{X}$ uniformly at random with replacement. Let D_i be the random variable corresponding to the number of times strand $\boldsymbol{x}_i$ is drawn and define the random vector $\boldsymbol{D} \triangleq (D_1, \ldots, D_M)$. We set the limit of drawn sequences to $N \geq M$, where we define the coverage depth as $c = \frac{N}{M}$. Then, the random vector $\boldsymbol{D} = (D_1, \ldots, D_M)$ is multinomially distributed as $\boldsymbol{D} \sim \mathsf{Multinom}(M, N, 1/M)$. With this definition of drawing, we model the PCR amplification and sequencing processes that act on the strand in the gene pool.

2. *Transmit:* The drawn strands are transmitted through independent IDS channels, as presented in the previous section, with identical parameters p_I, p_D, and p_S. We denote by $\boldsymbol{z}_j$ the output of the j-th IDS channel, $j \in \{1, \ldots, N\}$, and define $\boldsymbol{Z} \triangleq (\boldsymbol{z}_1, \ldots, \boldsymbol{z}_N)$. Here we are modeling the errors that arise from synthesis and sequencing, where there will exist multiple noisy copies of our target DNA strands.

3. *Permute:* The final output of the channel, denoted as the list $\boldsymbol{Y} \triangleq (\boldsymbol{y}_1, \ldots, \boldsymbol{y}_N)$ is obtained by a permutation of $\boldsymbol{Z}$ chosen uniformly at random. $\boldsymbol{Y}$ are the received sequences after the sequencing process. In turn, this models the loss of ordering the strands go through while in the gene pool.

Now, the receiver has to infer from $\boldsymbol{Y}$ the different clusters, or strands, that exist in the gene pool and has to also re-order them. In part B of the thesis, we present a work that designs an error-correction scheme that can overcome these challenges.

4.3 Memory-k Event Dependent Insertions-Deletions-Substitutions Channel

We assumed previously that the IDS errors that arise in DNA-based storage systems are i.i.d., as in, the errors are independent of the message content, position in the DNA sequence, and previous events. However, by surveying several related works [6, 10, 61, 62], we expect the errors to be correlated. The degree and type of correlation depend on many factors, including which sequencing technology is used.

First, we have asymmetric substitution errors between different bases. In the literature, due to the chemistry of the different bases, we know that some bases are more likely to be substituted by a specific other base. Consequently, we have to take this into consideration in our channel model. Second, in synthesis and sequencing, burst error events are quite common [7, 15]. Third, and most importantly, when it comes to Oxford nanopore sequencing, we will have different error statistics depending on which sequence of bases has passed through the pore. In addition, errors at any time instant can also depend on past errors. Hence, to

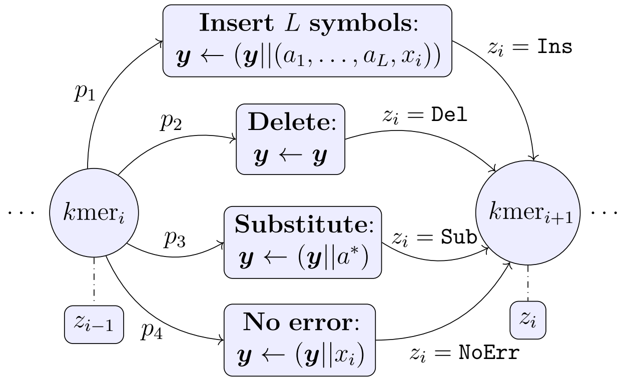

Insert L symbols: $\boldsymbol{y} \leftarrow (\boldsymbol{y}||(a_1,\ldots,a_L,x_i))$
Delete: $\boldsymbol{y} \leftarrow \boldsymbol{y}$
Substitute: $\boldsymbol{y} \leftarrow (\boldsymbol{y}||a^*)$
No error: $\boldsymbol{y} \leftarrow (\boldsymbol{y}||x_i)$
$k\text{mer}_i$
$k\text{mer}_{i+1}$
p_1
p_2
p_3
p_4
$z_i = \texttt{Ins}$
$z_i = \texttt{Del}$
$z_i = \texttt{Sub}$
$z_i = \texttt{NoErr}$
z_{i-1}
z_i

Figure 4.3: Memory-k event dependent IDS channel model.

have a realistic channel model that introduces IDS errors, it should account for the asymmetric substitution rates, sequence content-dependent errors, and previous error-dependent errors.

The authors in [61] were the first to introduce an IDS channel model with correlated errors. The authors proposed a statistical model of the DNA storage channel with nanopore sequencing based on a Markov chain that correlates error and sequence content by representing them as states in the chain. With this, they aimed to model the inherent memory of the DNA storage channel. In particular, it builds on the fact that in the MinION technology, strands traverse a nanopore nucleotide by nucleotide, and an electrical signal is generated for every group of $k \geq 1$ successive nucleotides, called $k$mers. To this end, let $\boldsymbol{x}$ be the DNA strand to be synthesized and $\boldsymbol{z}$ be the sequence of channel events, where $z_i \in \{\texttt{insertion}, \texttt{deletion}, \texttt{substitution}, \texttt{no error}\}$. The key idea in [61] is then to assume that z_i depends on the symbols $x_{i-k+1}, \ldots, x_i$ and the previous event z_{i-1}. This will introduce correlations between the event (error), at the current time instant, and the previous event and the k previous symbols. In turn, the transition probabilities $p(z_i|x_{i-k+1}, \ldots, x_i, z_{i-1})$ can be estimated from experimental data. We refer to this channel model as the memory-k nanopore channel model, as shown in Fig. 4.3.

Let $\boldsymbol{x} = (x_1, \ldots, x_n)$, representing the DNA strand, be the channel input sequence and $\boldsymbol{y} = (y_1, \ldots, y_{n'})$ the channel output sequence, where again the drift at the end is $d_n = n' - n$. Also, let $k\text{mer}_i = (x_{i-k+1}, \ldots, x_i)$ be the $k$mer at time instant $i \geq k$, while for $i < k$, since no complete $k$mer is available, we use $k\text{mer}_i = x_i$. Here, we denote the event associated with $k\text{mer}_{i+1}$ by z_i, where z_i represents an insertion, deletion, substitution, or no error acting on symbol x_i when it is to be transmitted. Hence, $z_i \in \{\texttt{Ins}, \texttt{Del}, \texttt{Sub}, \texttt{NoErr}\}$. The transition probabilities p_1, p_2, p_3, and p_4 in the figure are of the form $p(z_i|k\text{mer}_i, z_{i-1})$, and their value depends on the $k$mer. In the event of a deletion, the symbol x_i in $k\text{mer}_i$ will be deleted, and nothing will be appended to $\boldsymbol{y}$, while when a no error event occurs, $\boldsymbol{y}$ will be appended with x_i. Furthermore, in the event of an insertion, i.e., $z_i = \texttt{Ins}$, the channel will insert $L > 1$ symbols $a_1, \ldots, a_L$, where the length L is random. Each symbol $a_j \in \{0, 1, 2, 3\}$ will be uniformly picked at random. In this case, the channel appends $\boldsymbol{y}$ with $(a_1, \ldots, a_L)$. This also applies to a substitution event where the symbol a^* is randomly picked according to the distribution of the asymmetric substitution matrix and appended to $\boldsymbol{y}$ instead of x_i. In all these scenarios, the channel always moves from state $k\text{mer}_i$ to state $k\text{mer}_{i+1}$. After the last symbol x_n has been processed by the channel, the channel outputs $\boldsymbol{y}$.

Chapter 5

Paper Overview

In this chapter, we give an overview of what is included in our papers and a summary of the contributions.

5.1 Paper I

In the paper "*Concatenated Codes for Multiple Reads of a DNA Sequence*," we introduce concatenated coding schemes for multiple sequence transmission over parallel IDS channels over the DNA alphabet. The proposed schemes employ an inner block code or convolutional code that corrects insertions and deletions, concatenated with an outer LDPC or polar code, which corrects remaining (mostly substitution) errors. Our key contribution is a low-complexity decoding strategy that properly combines information from the multiple reads. Compared to MSA techniques, which make hard decisions on the DNA symbols, the proposed strategy provides soft information to the outer decoder. The proposed schemes achieve significant gains over the single sequence transmission case and outperform MSA if the number of traces is small. Furthermore, for a larger number of traces, they yield comparable performance to MSA at lower complexity. Our main contributions are summarized as follows.

- We propose two novel decoding algorithms for the decoding of the combination of the inner code and the IDS channel with multiple reads: an exact symbolwise MAP decoder based on a multidimensional trellis encompassing the inner code and the multiple IDS channels, and a suboptimal decoder that decodes each received sequence independently and combines the results into a single sequence of approximate APPs that are fed to the outer decoder. The first decoder is optimal but of high complexity, whereas the second decoder is a practical decoder that significantly reduces the complexity while achieving excellent performance.

- We improve on the performance of earlier concatenated schemes by providing optimization techniques tailored to the IDS channel for both the inner and outer codes. In particular, we design an inner time-varying code concatenated with either an LDPC code or a polar code.

- To gain insight into the asymptotic performance of the proposed schemes, we compute AIRs for the case with no iterations between the inner and outer decoder and the case where iterations are allowed. The computed rates depend on the inner code—but not on the outer code—and thus can be used to evaluate and select the inner code.

5.2 Paper II

In the paper "*Finite Blocklength Performance Bound for the DNA Storage Channel*," we evaluate the finite blocklength performance of the coding schemes of Paper I. We provide a finite blocklength performance bound for a DNA storage channel with IDS errors. Particularly, we consider the DT bound based on the *random coding* principle, which gives an upper bound on the frame error probability achievable over the DNA storage channel. The bound is tailored to a concatenated coding scheme that uses an inner synchronization code and depends on the inner code. Hence, it can be used as a handy tool to optimize inner coding schemes for the finite blocklength regime. Further, the bound provides a benchmark to compare coding schemes for DNA storage in the practical short-to-medium blocklength regime. We also consider the optimization of an outer LDPC code for a given inner code using EXIT charts and show that an optimized concatenated coding scheme achieves a normalized rate of 87% to 97% with respect to the DT bound for a frame error probability of 10^{-3} and code rate $1/2$, depending on the sequence length.

5.3 Paper III

In the paper "*Achievable Information Rates and Concatenated Codes for the DNA Nanopore Sequencing Channel*," we deal with a more realistic DNA storage channel model as compared to Papers I and II. We derive the optimum MAP decoder for the memory-k nanopore channel model. The complexity of the decoder increases exponentially with k. Based on this decoder, we derive AIRs. An AIR for the memory-k nanopore channel can be seen as an AIR for the true DNA storage channel of a *mismatched* decoder that assumes that the channel is a memory-k nanopore channel. We show that for increasing k, the AIR improves—, meaning that the decoder is better matched to the true channel—and eventually saturates. This allows us to quantify the trade-off between decoding complexity and performance loss incurred by the suboptimal decoder. The derived MAP decoder can be used to design error-correcting coding schemes tailored to the memory-k nanopore sequencing channel. In particular, we consider the concatenated coding scheme proposed in Paper I and multiple reads of the DNA strand, and we optimize the inner and outer code. We show that the concatenated coding scheme of Paper I yields excellent performance at rates close to the AIRs.

5.4 Paper IV

Finally, in the paper "*Index-Based Concatenated Codes for the Multi-Draw DNA Storage Channel*," we propose an index-based concatenated coding scheme for the multi-draw IDS channel, a more realistic channel model that incorporates the clustering and indexing problem introduced in Chapter 2. In our coding scheme introduced in this paper, the information is first encoded by an outer code to provide overall error protection, primarily coping with unresolved errors and strand erasures of the data. The resulting codeword is split into M data blocks. In each data block, index information is embedded after being encoded by a low-rate index code, whereas the data block itself is encoded by an inner code. Both codes are tailored to deal with IDS errors. The M resulting strands are then transmitted over the multi-draw IDS channel. At the receiver, each noisy strand is decoded separately by a joint index and inner MAP decoder outputting symbolwise APPs. To leverage the multi-copy gain, we combine the APPs according to the subsequent clustering and index decoding. We optionally cluster the received strands based on the obtained APPs, benefiting from the error-correction capabilities of the index and inner codes. For each cluster, an index decision is made by jointly considering the received strands in a cluster. The APPs are ordered according to the index decisions and fed into a soft-input outer decoder.

We analyze our proposed scheme in terms of AIRs. The rates provide insights to the performance of different index-inner coding and decoding techniques and a benchmark for an outer code. For different coding setups in the finite blocklength regime, we compute information-outage probabilities on synthetic data and present frame error rates on experimental data.

Chapter 6

Conclusion

In this thesis, the main goal was to design coding schemes that can combat different types of errors that arise from the DNA storage channel. To accomplish this, it was essential to first gain a fundamental understanding of the channel. By reviewing the literature of relevant experiments that showed the viability of DNA as a storage medium and examining the technologies used, we identified the main components of a DNA-based storage system. We then identified the types of errors that arise from each component where a DNA strand can be affected by IDS errors. Furthermore, strand erasures and loss of order are other challenges that must be addressed. The next step was to build a channel model that incorporates these types of errors, enabling the design of a coding scheme tailored to the DNA storage channel. However, to gain a more fundamental insight into the impact of different types of errors that arise from the channel, we first studied a simplistic channel model with i.i.d. IDS errors where strand erasures and loss of order are perfectly resolved. We then expanded on the simplistic channel model by introducing the multi-draw DNA storage channel, where N uniformly random shuffled draws from $M < N$ target sequences are performed, which in turn introduces strand erasures and loss of order, in addition to the need for clustering. Finally, we studied a more advanced channel model that incorporates correlated IDS errors as opposed to i.i.d. IDS errors as in the first simplistic model.

In our first and second papers, we proposed a coding scheme for the first simplistic channel model, while the third and fourth papers targeted the second and third channel models, respectively. The coding schemes proposed in these papers are all in the form of concatenated codes, where we combine an inner synchronization block code with an outer (non-binary) LDPC code (or an outer polar code). We evaluated the performance of these schemes via frame error rate simulations along with finite and asymptotic performance bounds. The finite length performance bounds included the DT bound and an outage probability bound. The asymptotic bounds used are the mutual information rate R_{MI} and the BCJR-once rate $R_{\text{BCJR-once}}$, both being AIRs. When it came to code design and optimization, we used DE and EXIT charts as tools for the outer LDPC code, while for the inner block code, we used heuristic techniques.

6.1 Future Work

There are several avenues to extend the work in this thesis. Below, we give a list of possible interesting extensions.

- In Paper I, we computed the mutual information rate for the case of a single received sequence and for joint decoding in the case of multiple received sequences. However, for the case of multiple received sequences and with separate decoding, devising a procedure for its computation was left as an open problem. This appears to be challenging but would be important in order to evaluate the ultimate performance limit of separate decoding under iterative inner-outer decoding.
- Having derived the DT bound for the finite-length regime in Paper II, a logical extension to that work is to find a converse bound as well. In addition, we can also perform Markovian shaping of the channel input in order to match it to the DNA storage channel.
- One clear way to expand on the work in this thesis is to combine the multi-draw channel model with the channel model with correlated IDS errors. This will result in a more realistic model that is much closer to the true DNA storage channel.
- In Paper IV, we introduce a clustering algorithm based on the LLRs of the received sequences from the output of the inner decoder. Although this clustering approach has lower complexity compared to other algorithms used in the literature, it still proved to be computationally costly for large blocklengths. Hence, more work is needed to make the clustering practically feasible.
- As always, backing up theoretical work with real-world experiments is important, even if the design is done using realistic datasets. Hence, we believe that an important avenue for future work would be to evaluate the performance of the proposed coding schemes through real DNA experiments where data is synthesized and sequenced using the proposed schemes and decoding methods.
- Moreover, the question of how secure communication, i.e., cryptography, can be implemented through the DNA storage channel is an interesting research question that we did not study in this thesis. Although some work has been done, for example, in [63], the authors study the DNA wiretap channel, more work needs to be focused on the cryptographic aspects of the channel.

Bibliography

[1] D. Panda, K. A. Molla, M. J. Baig, A. Swain, D. Behera, and M. Dash, "DNA as a digital information storage device: hope or hype?," *3 Biotech*, vol. 8, pp. 1–9, May 2018.

[2] V. Zhirnov, R. M. Zadegan, G. S. Sandhu, G. M. Church, and W. L. Hughes, "Nucleic acid memory," *Nature Materials*, vol. 15, pp. 366–370, Apr. 2016.

[3] H. J. Porck and R. Teygeler, "Preservation science survey: An overview of recent developments in research on the conservation of selected analog library and archival materials," tech. rep., Alexandria, VA, USA, Dec. 2000.

[4] G. M. Church, Y. Gao, and S. Kosuri, "Next-generation digital information storage in DNA," *Science*, vol. 337, p. 1628, Aug. 2012.

[5] N. Goldman, P. Bertone, S. Chen, C. Dessimoz, E. M. LeProust, B. Sipos, and E. Birney, "Towards practical, high-capacity, low-maintenance information storage in synthesized DNA," *Nature*, vol. 494, pp. 77–80, Feb. 2013.

[6] R. N. Grass, R. Heckel, M. Puddu, D. Paunescu, and W. J. Stark, "Robust chemical preservation of digital information on DNA in silica with error-correcting codes," *Angew. Chem. Int. Ed.*, vol. 54, pp. 2552–2555, Feb. 2015.

[7] M. Blawat, K. Gaedke, I. Hütter, X.-M. Chen, B. Turczyk, S. Inverso, B. W. Pruitt, and G. M. Church, "Forward error correction for DNA data storage," in *Proc. Int. Conf. Comput. Sci.*, (San Diego, CA, USA), pp. 1011–1022, June 2016.

[8] J. Bornholt, R. Lopez, D. M. Carmean, L. Ceze, G. Seelig, and K. Strauss, "A DNA-based archival storage system," in *Proc. Int. Conf. Architect. Supp. Program. Lang. Operat. Syst.*, (Atlanta, Georgia, USA), pp. 637–649, Mar. 2016.

[9] Y. Erlich and D. Zielinski, "DNA fountain enables a robust and efficient storage architecture," *Science*, vol. 355, pp. 950–954, Mar. 2017.

[10] S. M. H. T. Yazdi, R. Gabrys, and O. Milenkovic, "Portable and error-free DNA-based data storage," *Sci. Rep.*, vol. 7, pp. 1–6, July 2017.

[11] L. Organick, S. D. Ang, Y.-J. Chen, R. Lopez, S. Yekhanin, K. Makarychev, M. Z. Racz, G. Kamath, P. Gopalan, B. Nguyen, C. N. Takahashi, S. Newman, H.-Y. Parker, C. Rashtchian, K. Stewart, G. Gupta, R. Carlson, J. Mulligan,

D. Carmean, G. Seelig, L. Ceze, and K. Strauss, “Random access in large-scale DNA data storage,” *Nature Biotechnol.*, vol. 36, pp. 242–248, Mar. 2018.

[12] Y. Wang, M. Noor-A-Rahim, J. Zhang, E. Gunawan, Y. L. Guan, and C. L. Poh, “High capacity DNA data storage with variable-length oligonucleotides using repeat accumulate code and hybrid mapping,” *J. Biol. Eng.*, vol. 13, pp. 1–11, Nov. 2019.

[13] S. Chandak, K. Tatwawadi, B. Lau, J. Mardia, M. Kubit, J. Neu, P. Griffin, M. Wootters, T. Weissman, and H. Ji, “Improved read/write cost tradeoff in DNA-based data storage using LDPC codes,” in *Proc. Annu. Allerton Conf. Commun., Control, Comp.*, (Monticello, IL, USA), pp. 147–156, Sept. 2019.

[14] S. Chandak, J. Neu, K. Tatwawadi, J. Mardia, B. Lau, M. Kubit, R. Hulett, P. Griffin, M. Wootters, T. Weissman, and H. Ji, “Overcoming high nanopore basecaller error rates for DNA storage via basecaller-decoder integration and convolutional codes,” in *Proc. IEEE Int. Conf. Acoust., Speech, Sig. Process.*, (Barcelona, Spain), pp. 8822–8826, May 2020.

[15] S. M. H. T. Yazdi, Y. Yuan, J. Ma, H. Zhao, and O. Milenkovic, “A rewritable, random-access DNA-based storage system,” *Sci. Rep.*, vol. 5, pp. 1–10, Sept. 2015.

[16] C. Pan, S. K. Tabatabaei, S. M. H. T. Yazdi, A. G. Hernandez, C. M. Schroeder, and O. Milenkovic, “Rewritable two-dimensional DNA-based data storage with machine learning reconstruction,” *Nature Commun.*, vol. 13, pp. 1–12, May 2022.

[17] S. K. Tabatabaei, B. Pham, C. Pan, J. Liu, S. Chandak, S. A. Shorkey, A. G. Hernandez, A. Aksimentiev, M. Chen, C. M. Schroeder, and O. Milenkovic, “Expanding the molecular alphabet of DNA-based data storage systems with neural network nanopore readout processing,” *Nano Lett.*, vol. 22, pp. 1905–1914, Mar. 2022.

[18] P. L. Antkowiak, J. Lietard, M. Z. Darestani, M. M. Somoza, W. J. Stark, R. Heckel, and R. N. Grass, “Low cost DNA data storage using photolithographic synthesis and advanced information reconstruction and error correction,” *Nature Commun.*, vol. 11, pp. 1–10, Oct. 2020.

[19] R. Heckel, G. Mikutis, and R. N. Grass, “A characterization of the DNA data storage channel,” *Sci. Rep.*, vol. 9, pp. 1–12, July 2019.

[20] L. Ceze, J. Nivala, and K. Strauss, “Molecular digital data storage using DNA,” *Nature Reviews Genetics*, vol. 20, pp. 456–466, Aug. 2019.

[21] R. Heckel, I. Shomorony, K. Ramchandran, and D. N. C. Tse, “Fundamental limits of DNA storage systems,” in *Proc. IEEE Int. Symp. Inf. Theory*, (Aachen, Germany), pp. 3130–3134, June 2017.

[22] W. Mao, S. N. Diggavi, and S. Kannan, "Models and information-theoretic bounds for nanopore sequencing," *IEEE Trans. Inf. Theory*, vol. 64, pp. 3216–3236, Apr. 2018.

[23] S. R. Srinivasavaradhan, S. Gopi, H. D. Pfister, and S. Yekhanin, "Trellis BMA: Coded trace reconstruction on IDS channels for DNA storage," in *Proc. IEEE Int. Symp. Inf. Theory*, (Melbourne, Australia), pp. 2453–2458, July 2021.

[24] M. C. Davey and D. J. C. MacKay, "Reliable communication over channels with insertions, deletions, and substitutions," *IEEE Trans. Inf. Theory*, vol. 47, pp. 687–698, Feb. 2001.

[25] H. Li and R. Durbin, "Fast and accurate short read alignment with Burrows-Wheeler transform," *Bioinformatics*, vol. 25, pp. 1754–1760, July 2009.

[26] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon limit error-correcting coding and decoding: Turbo-codes. 1," in *Proc. IEEE Int. Conf. Commun.*, (Geneva, Switzerland), pp. 1064–1070, May 1993.

[27] D. J. C. MacKay and R. M. Neal, "Near Shannon limit performance of low density parity check codes," *Electron. Lett.*, vol. 32, pp. 1645–1646, Aug. 1996.

[28] G. D. Forney, Jr., "The Viterbi algorithm," *Proc. IEEE*, vol. 61, pp. 268–278, Mar. 1973.

[29] L. R. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal decoding of linear codes for minimizing symbol error rate," *IEEE Trans. Inf. Theory*, vol. 20, pp. 284–287, Mar. 1974.

[30] W. E. Ryan and S. Lin, *Channel Codes: Classical and Modern.* Cambridge Univ. Press, 2009.

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

[32] X.-Y. Hu, E. Eleftheriou, and D.-M. Arnold, "Progressive edge-growth Tanner graphs," in *Proc. IEEE Glob. Telecommun. Conf.*, (San Antonio, TX, USA), pp. 995–1001, Nov. 2001.

[33] G. D. Forney, Jr., "Concatenated codes," Tech. Rep. 440, Cambridge, MA, USA, Dec. 1965.

[34] S.-Y. Chung, T. J. Richardson, and R. L. Urbanke, "Analysis of sum-product decoding of low-density parity-check codes using a Gaussian approximation," *IEEE Trans. Inf. Theory*, vol. 47, pp. 657–670, Feb. 2001.

[35] A. Kavčić, X. Ma, and M. Mitzenmacher, "Binary intersymbol interference channels: Gallager codes, density evolution, and code performance bounds," *IEEE Trans. Inf. Theory*, vol. 49, pp. 1636–1652, July 2003.

[36] R. R. Müller and W. H. Gerstacker, "On the capacity loss due to separation of detection and decoding," *IEEE Trans. Inf. Theory*, vol. 50, pp. 1769–1778, Aug. 2004.

[37] J. B. Soriaga, H. D. Pfister, and P. H. Siegel, "Determining and approaching achievable rates of binary intersymbol interference channels using multistage decoding," *IEEE Trans. Inf. Theory*, vol. 53, pp. 1416–1429, Apr. 2007.

[38] H. D. Pfister, J. B. Soriaga, and P. H. Siegel, "On the achievable information rates of finite state ISI channels," in *Proc. IEEE Glob. Telecommun. Conf.*, (San Antonio, TX, USA), pp. 2992–2996, Nov. 2001.

[39] D. M. Arnold, H.-A. Loeliger, P. O. Vontobel, A. Kavčić, and W. Zeng, "Simulation-based computation of information rates for channels with memory," *IEEE Trans. Inf. Theory*, vol. 52, pp. 3498–3508, Aug. 2006.

[40] Y. Polyanskiy, H. V. Poor, and S. Verdú, "Channel coding rate in the finite blocklength regime," *IEEE Trans. Inf. Theory*, vol. 56, pp. 2307–2359, May 2010.

[41] N. Weinberger and N. Merhav, "The DNA storage channel: Capacity and error probability bounds," *IEEE Trans. Inf. Theory*, vol. 68, pp. 5657–5700, Sept. 2022.

[42] D. Buckingham and M. C. Valenti, "The information-outage probability of finite-length codes over AWGN channels," in *Proc. 42nd Annu. Conf. Inf. Sci. Syst.*, (Princeton, USA), Mar. 2008.

[43] R. G. Gallager, "Sequential decoding for binary channel with noise and synchronization errors," tech. rep., Arlington, VA, USA, Sept. 1961.

[44] K. S. Zigangirov, "Sequential decoding for a binary channel with drop-outs and insertions," *Probl. Peredachi Inf.*, vol. 5, no. 2, pp. 23–30, 1969.

[45] L. Calabi and W. E. Hartnett, "Some general results of coding theory with applications to the study of codes for the correction of synchronization errors," *Inf. Control*, vol. 15, pp. 235–249, Sept. 1969.

[46] L. R. Bahl and F. Jelinek, "Decoding for channels with insertions, deletions, and substitutions with applications to speech recognition," *IEEE Trans. Inf. Theory*, vol. 21, pp. 404–411, July 1975.

[47] V. I. Levenshtein, "Binary codes capable of correcting deletions, insertions and reversals," *Soviet Physics Doklady*, vol. 10, pp. 707–710, Feb. 1966.

[48] M. F. Mansour and A. H. Tewfik, "Convolutional decoding in the presence of synchronization errors," *IEEE J. Sel. Areas Commun.*, vol. 28, pp. 218–227, Feb. 2010.

[49] V. Buttigieg and N. Farrugia, "Improved bit error rate performance of convolutional codes with synchronization errors," in *Proc. IEEE Int. Conf. Commun.*, (London, U.K.), pp. 4077–4082, June 2015.

[50] J. A. Briffa, H. G. Schaathun, and S. Wesemeyer, “An improved decoding algorithm for the Davey-MacKay construction,” in *Proc. IEEE Int. Conf. Commun.*, (Cape Town, South Africa), May 2010.

[51] M. Inoue and H. Kaneko, “Adaptive synchronization marker for insertion/deletion/substitution error correction,” in *Proc. IEEE Int. Symp. Inf. Theory*, (Cambridge, MA, USA), pp. 508–512, July 2012.

[52] R. Shibata, G. Hosoya, and H. Yashima, “Design of irregular LDPC codes without markers for insertion/deletion channels,” in *Proc. IEEE Glob. Commun. Conf.*, (Waikoloa, HI, USA), Dec. 2019.

[53] H. D. Pfister and I. Tal, “Polar codes for channels with insertions, deletions, and substitutions,” in *Proc. IEEE Int. Symp. Inf. Theory*, (Melbourne, Australia), pp. 2554–2559, July 2021.

[54] J. A. Briffa and H. G. Schaathun, “Improvement of the Davey-MacKay construction,” in *Proc. Int. Symp. Inf. Theory Appl.*, (Auckland, New Zealand), Dec. 2008.

[55] V. Buttigieg and J. A. Briffa, “Codebook and marker sequence design for synchronization-correcting codes,” in *Proc. IEEE Int. Symp. Inf. Theory*, (Saint Petersburg, Russia), pp. 1579–1583, July/Aug. 2011.

[56] V. Buttigieg and J. A. Briffa, “Improved code construction for synchronization error correction,” in *Proc. 10th Int. ITG Conf. Syst., Commun. Coding*, (Hamburg, Germany), Feb. 2015.

[57] R. R. Varshamov and G. M. Tenengol’ts, “Code correcting single asymmetric errors,” *Avtomat. Telemekh.*, vol. 26, no. 2, pp. 288–292, 1965.

[58] A. A. El Gamal, L. A. Hemaspaandra, I. Shperling, and V. K. Wei, “Using simulated annealing to design good codes,” *IEEE Trans. Inf. Theory*, vol. 33, pp. 116–123, Jan. 1987.

[59] I. Shomorony and R. Heckel, “Information-theoretic foundations of DNA data storage,” *Found. Trends Commun. Inf. Theory*, vol. 19, pp. 1–106, Feb. 2022.

[60] A. Lenz, P. H. Siegel, A. Wachter-Zeh, and E. Yaakobi, “Coding over sets for DNA storage,” *IEEE Trans. Inf. Theory*, vol. 66, pp. 2331–2351, Apr. 2020.

[61] B. Hamoum, E. Dupraz, L. Conde-Canencia, and D. Lavenier, “Channel model with memory for DNA data storage with nanopore sequencing,” in *Proc. Int. Symp. Topics Coding*, (Montréal, QC, Canada), Aug./Sept. 2021.

[62] B. McBain, E. Viterbo, and J. Saunderson, “Finite-state semi-Markov channels for nanopore sequencing,” in *Proc. IEEE Int. Symp. Inf. Theory*, (Espoo, Finland), pp. 216–221, June/July 2022.

[63] P. K. Vippathalla and N. Kashyap, “The secure storage capacity of a DNA wiretap channel model,” *IEEE Trans. Inf. Theory*, vol. 69, pp. 5550–5569, Sept. 2023.

Part B

Papers

Paper I

Concatenated Codes for Multiple Reads of a DNA Sequence

Issam Maarouf, Lorenz Walter, Andreas Lenz, Antonia Wachter-Zeh, Eirik Rosnes, and Alexandre Graell i Amat
IEEE Transactions on Information Theory, vol. 69, no. 2, pp. 910–927, Feb. 2023.

Parts of this paper were presented at the IEEE Information Theory Workshop (ITW), Riva del Garda, Italy, Apr. 2021.

The layout has been revised.

Abstract

Decoding sequences that stem from multiple transmissions of a codeword over an insertion, deletion, and substitution channel is a critical component of efficient deoxyribonucleic acid (DNA) data storage systems. In this paper, we consider a concatenated coding scheme with an outer nonbinary low-density parity-check code or a polar code and either an inner convolutional code or a time-varying block code. We propose two novel decoding algorithms for inference from multiple received sequences, both combining the inner code and channel to a joint hidden Markov model to infer symbolwise a posteriori probabilities (APPs). The first decoder computes the exact APPs by jointly decoding the received sequences, whereas the second decoder approximates the APPs by combining the results of separately decoded received sequences and has a complexity that is linear with the number of sequences. Using the proposed algorithms, we evaluate the performance of decoding multiple received sequences by means of achievable information rates and Monte-Carlo simulations. We show significant performance gains compared to a single received sequence. In addition, we succeed in improving the performance of the aforementioned coding scheme by optimizing both the inner and outer codes.

1 Introduction

Error correction of data storage in deoxyribonucleic acid (DNA) has recently gained a lot of attention from the coding theory community. This attention increased after several successful experiments [1–14] that demonstrated the viability of using synthetic DNA as a reliable medium for data storage. As a result of the pioneering DNA storage experiments, several information-theoretic problems have been identified. The most important problem to our work is reliable communication over channels that introduce insertions, deletions, and substitutions (IDSs) [15] as the processes of DNA synthesis and DNA sequencing introduce errors in the forms of IDSs. Furthermore, in the literature, channels that introduce IDSs have been proposed to model synchronization errors. Thus, coding techniques are an indispensable component to cope with IDSs and improve the reliability of DNA storage systems and channels that are prone to desynchronization.

Work on synchronization errors began decades ago. Several papers in the 1960s-70s have dealt with information-theoretic aspects of IDS errors, some even proposed codes to correct these errors [16–20]. From these works, several constructions of error-correcting codes for the IDS channel have been proposed in the last decade. Among the most important ones, and most relevant to our work, is the one introduced by Davey and MacKay [21]. In that paper, the authors introduce a concatenated coding scheme composed of an inner block code and an outer low-density parity-check (LDPC) code. In addition, they propose a decoding algorithm based on dynamic programming that represents the inner code and channel by a hidden Markov model (HMM). By doing so, the decoding algorithm allows to infer a posteriori probabilities (APPs) to be passed to the outer decoder, which will complete the message recovery process. Inspired by Davey and MacKay's work, the Markov process of convolutional codes was extended to the IDS channel, allowing for decoding algorithms of convolutional codes to be run for the IDS channel [22, 23]. An improvement of the decoding algorithm in [21] was introduced in [24]. Furthermore, marker codes were used as inner codes in [25], which improved the performance of the inner codes of [21]. Additionally, standalone codes (i.e., without an inner code) such as LDPC codes in [26] and polar codes in [27, 28] were studied to tackle synchronization errors.

Most of these studies have focused on error correction for a single block transmission over an IDS channel. However, in DNA-based storage the data is synthesized into many short DNA strands (or sequences), where each strand is replicated thousands of times. As a result, when the data is read via DNA sequencing, multiple copies of each data strand are obtained.

Decoding multiple reads of a DNA sequence is typically performed via a multiple sequence alignment (MSA) [29–32] of the received sequences, followed by a majority decision on the alignment. This method is popular, since it is possible to employ standard decoders on the single resulting sequence to correct remaining errors and has been used in several recent experiments [7, 8, 12, 14].

In this paper, we introduce concatenated coding schemes for multiple sequence transmission over parallel IDS channels over the DNA alphabet. The proposed schemes employ an inner block code or convolutional code (as introduced in [21] and [33], respectively) that corrects insertions and deletions, concatenated with an outer LDPC or polar code, which corrects remaining (mostly substitution) errors. Our key contribution is a low-complexity decoding strategy that properly combines information from the multiple reads. Compared to MSA techniques, which make hard decisions on the DNA symbols, the proposed strategy provides soft information to the outer decoder. The proposed schemes achieve significant gains over the single sequence transmission case and outperform MSA if the number of traces is small. Furthermore, for a larger number of traces, they yield comparable performance to MSA at lower complexity. Our main contributions are summarized as follows.

- We propose two novel decoding algorithms for the decoding of the combination of the inner code and the IDS channel with multiple reads: an exact symbolwise maximum a posteriori (MAP) decoder based on a multidimensional trellis encompassing the inner code and the multiple IDS channels, and a sub-optimal decoder that decodes each received sequence independently and combines the results into a single sequence of approximate APPs that are fed to the outer decoder. The first decoder is optimal but of high complexity, whereas the second decoder is a practical decoder that significantly reduces the complexity while achieving excellent performance. For instance, for a considered scenario, we show that separate decoding with three sequences has comparable performance to joint decoding with two sequences, but with much lower decoding complexity.

- We improve on the performance of earlier concatenated schemes by providing optimization techniques tailored to the IDS channel for both the inner and outer codes. In particular, we design an inner time-varying code (TVC) concatenated with either an LDPC code or a polar code.
- To gain insight into the asymptotic performance of the proposed schemes, we compute achievable information rates (AIRs) for the case with no iterations between the inner and outer decoder and the case where iterations are allowed. The computed AIRs depend on the inner code—but not on the outer code—and thus can be used to evaluate and select the inner code.

Related Work

Work related to ours includes the study of repeated transmission over an erroneous channel, which goes back to Levenshtein [34], who introduced the sequence reconstruction problem. Here, the goal is to analytically quantify the number of sequences, as a function of the sequence length, that are required to guarantee correct reconstruction under an adversarial channel. Recently, the original formulation has been extended to an adversarial DNA channel [35]. Furthermore, similar in spirit to our work, yet different in methods and objectives, is the research on the trace reconstruction problem over deletion channels [36–38], which is the probabilistic variant of the sequence reconstruction problem. These works focus on asymptotic results, while we discuss both asymptotic and finite-length results of received sequences over a channel that additionally allows insertions and substitutions. More recently, the trace reconstruction problem has also been formulated for a fixed number of sequences with a larger focus on algorithmic aspects [39, 40]. However, these works consider only uncoded sequences, while we are interested in coded transmissions.

Inspired by our work [41], independently of this paper, the authors in [42] worked on multiple sequence transmission over parallel IDS channels. In particular in [42], they presented a practical sub-optimal decoder, referred to as trellis bitwise majority alignment, which slightly improves the performance compared to our sub-optimal decoder, while slightly increasing the complexity. The authors in [42] also proposed a less complex decoder than our optimal APP decoder. In [43], a more complex sub-optimal decoder compared to our separate decoder was presented for multiple sequence transmission over the IDS channel, as well as a similar optimal APP decoder as ours. In [44], a concatenated coding scheme with an outer LDPC code and an inner trellis code approaching the Markov capacity of a channel with insertions and deletions for single sequence transmission was proposed.

Other works that discuss AIRs for IDS channels are [44, 45]. These works consider a binary channel and the single sequence case. In [45], bounds on the capacity of binary IDS channels are computed, while in [44] a Markovian input process for their binary insertion and deletion channel is optimized, which results in better AIRs.

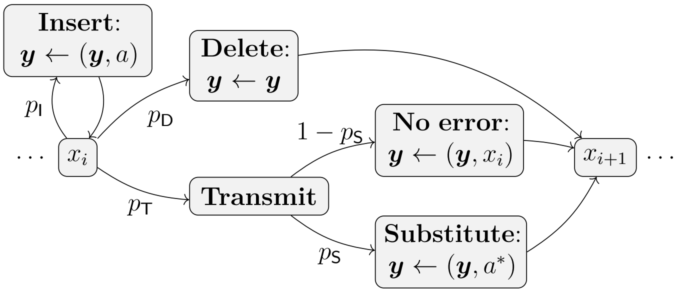

Insert: $\boldsymbol{y} \leftarrow (\boldsymbol{y}, a)$
Delete: $\boldsymbol{y} \leftarrow \boldsymbol{y}$
No error: $\boldsymbol{y} \leftarrow (\boldsymbol{y}, x_i)$
Transmit
Substitute: $\boldsymbol{y} \leftarrow (\boldsymbol{y}, a^*)$
p_I
p_D
$1 - p_\mathsf{S}$
p_T
p_S
$\cdots$ x_i
x_{i+1} $\cdots$

Figure I.1: State-based IDS channel

2 System Model

2.1 Channel Model

We consider the IDS channel model depicted in Fig. I.1 [21–25, 27, 28, 46]. Let $\boldsymbol{x} = (x_1, \ldots, x_N)$, $x_i \in \Sigma_q = \{0, 1, \ldots, q-1\}$, be the information sequence to be transmitted over the channel. The sequence can be viewed as a queue where each symbol x_i is successively transmitted over the channel. We describe in the following how the received sequence $\boldsymbol{y} = (y_1, \ldots, y_{N'})$ is generated state by state. Assume x_i is queued and therefore sought to be transmitted over the channel. The channel enters state x_i and three events may occur: (i) With probability p_I, an insertion event occurs where a uniformly random symbol $a \in \Sigma_q$ is appended to the received sequence. In this case, x_i remains in the queue and the channel returns to state x_i. (ii) Symbol x_i is deleted with probability p_D. Consequently, the queued symbol x_i is not appended to $\boldsymbol{y}$ and the next symbol x_{i+1} is enqueued and the channel enters state x_{i+1}. (iii) x_i is transmitted with probability $p_\mathsf{T} = 1 - p_\mathsf{I} - p_\mathsf{D}$. In this case, the symbol is substituted with a uniformly random symbol $a^* \neq x_i$ with probability p_S and the next transmit symbol x_{i+1} is enqueued and the channel enters state x_{i+1}. When the last transmit symbol x_N leaves the queue, the procedure finishes and the channel outputs $\boldsymbol{y}$. Note that the length of the output sequence N' is random and depends on the probabilities p_D and p_I.

2.2 Coding Scheme

Following [21], we consider an error-correcting code consisting of the serial concatenation of two codes. The role of the inner code is to maintain synchronization and provide reliable soft information to the outer code, while the task of the outer code is to use this soft information to perform an accurate estimate of the transmitted information. In particular, given that the inner code provides reliable synchronization information, the outer code corrects the remaining errors on the synchronized sequence.

We investigate multiple choices of inner codes, which can be classified into two categories: (i) Convolutional codes as introduced in [33], and (ii) TVCs. Note that the inner code construction introduced in [21] can be seen as a non-time-varying block code or a convolutional code with memory zero and thus, for simplicity, we

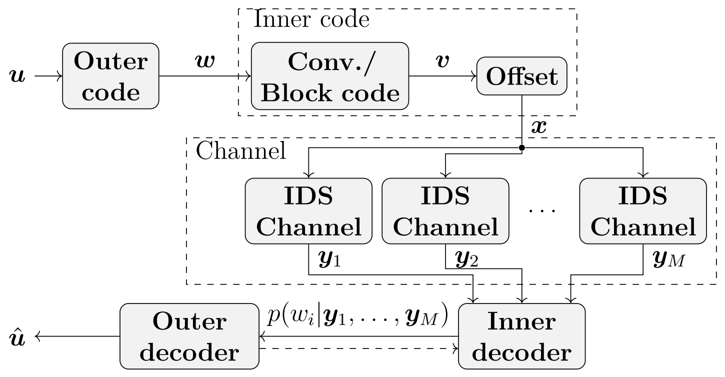


Figure I.2: Communication via multiple transmissions over an IDS channel with a concatenated coding scheme. The IDS channel, depicted in Fig. I.1, is fed M times with the encoded transmit sequence $\boldsymbol{x}$.

restrict some passages in this work to convolutional inner codes only. Moreover, the outer code is either a nonbinary LDPC code or a nonbinary polar code.

The considered system model is depicted in Fig. I.2. The information vector $\boldsymbol{u} = (u_1, \dots, u_K)$,$u_i \in \mathbb{F}_{q_\circ}$, is encoded by an $[N_\circ, K]_{q_\circ}$ outer code to a codeword $\boldsymbol{w} = (w_1, \dots, w_{N_\circ})$,$w_i \in \mathbb{F}_{q_\circ}$.

The sequence $\boldsymbol{x}$ is transmitted M times independently over an IDS channel resulting in the received sequences $\boldsymbol{y}_1, \dots, \boldsymbol{y}_M$ corresponding to M reads of the original strand. The inner decoder uses these received sequences to infer likelihoods for the symbols in $\boldsymbol{w}$. These likelihoods are then fed to the outer decoder, which decides on the decoded sequence $\hat{\boldsymbol{u}}$. Furthermore, we can also iterate between the inner and outer decoder, exchanging extrinsic information between them, which is referred to as *turbo decoding* in the literature.

2.3 Symbolwise MAP Decoding for the IDS Channel (Inner Decoding)

The random memory associated with the insertion and deletion processes makes decoding for IDS channels challenging. This is visualized by the receiver's inability to directly identify the origin of a received symbol y_i. Since insertions or deletions before symbol y_i might have moved the symbol right or left in the received sequence, y_i could be the result of transmitting a symbol $x_{i'}$ with $i' \neq i$. In the following, we describe how it is still possible to infer a posteriori likelihoods from this channel using a hidden Markov representation of the channel [21, 23, 24]. We present and discuss this HMM for $M = 1$ first and extend it to $M > 1$ multiple received sequences afterward. Here we restrict the discussion to convolutional inner codes, as a TVC with $t = 1$ can be viewed as a convolutional code with $m = 0$. The APP of the outer code symbol w_i is

$$p(w_i|\boldsymbol{y}) = \frac{p(\boldsymbol{y}, w_i)}{p(\boldsymbol{y})}.$$

The joint probability $p(\boldsymbol{y}, w_i)$ can be computed by de-marginalizing with respect to the memory states of the convolutional code corresponding to symbol w_i. This is possible due to the Markov property of the convolutional code. However, this Markov property no longer holds for the IDS channel as a result of its memory. To circumvent this issue, a new hidden state variable, the so-called *drift* [21], that incorporates insertions and deletions, is added to the original Markov process, creating an augmented hidden Markov process. The drift d_i, $0 \leq i < N_\mathsf{o} + m$, is defined as the number of insertions minus the number of deletions that occurred before symbol x_{ni+1} is enqueued, while $d_{N_\mathsf{o}+m}$ is defined as the number of insertions minus deletions that occurred after the last symbol $x_{n(N_\mathsf{o}+m)}$ has been transmitted. Thus, by definition, $d_0 = 0$ and $d_{N_\mathsf{o}+m} = N' - N$, both known to the receiver. In the resulting HMM, a transition from time $i-1$ to time i corresponds to a transmission of symbols $\boldsymbol{x}_{(i-1)n+1}^{in}$, where $\boldsymbol{x}_a^b = (x_a, x_{a+1}, \ldots, x_b)$. Further, when transitioning from state d_{i-1} to d_i, the HMM emits $n + d_i - d_{i-1}$ output symbols depending on both the previous and new drift. The key property of the drift is that, by its inclusion as a new state variable inside the Markov process, the Markov property is restored. This is because the drift sequence itself forms a Markov chain and, conditioned on the memory state and drift at time i, the output of the channel after time i becomes independent of the previous memory states and drifts. Introducing the joint state variable $\sigma_i = (s_i, d_i)$, where s_i denotes the state variables of the convolutional code, we obtain with slight abuse of notation

$$p(\boldsymbol{y}, w_i) = \sum_{(\sigma,\sigma'):w_i} p(\boldsymbol{y}, \sigma, \sigma'),$$

where σ and σ' denote realizations of the random variables σ_{i-1} and σ_i, respectively. The summation is over all convolutional code memory states that correspond to information symbol w_i. Using the Markov property, we can expand the joint probability $p(\boldsymbol{y}, \sigma, \sigma')$ into three parts as

$$p(\boldsymbol{y},\sigma,\sigma') = p\left(\boldsymbol{y}_1^{(i-1)n+d}, \sigma\right) p\left(\boldsymbol{y}_{(i-1)n+d+1}^{in+d'}, \sigma' \middle| \sigma\right) p\left(\boldsymbol{y}_{in+d'+1}^{N'} \middle| \sigma'\right).$$

Abbreviating the above terms by $\alpha_{i-1}(\sigma)$, $\gamma_i(\sigma, \sigma')$, and $\beta_i(\sigma')$ in order of appearance, one can deduce the forward and backward recursions

$$\begin{aligned} \alpha_i(\sigma') &= \sum_{\sigma} \alpha_{i-1}(\sigma)\gamma_i(\sigma, \sigma'), \\ \beta_{i-1}(\sigma) &= \sum_{\sigma'} \beta_i(\sigma')\gamma_i(\sigma, \sigma'). \end{aligned} \tag{I.1}$$

Knowing the initial and final drift of the received sequence, the initial and termination conditions of the forward and backward recursions are

$$\alpha_0(\sigma) = \begin{cases} 1 & \text{if } \sigma = (0,0) \\ 0 & \text{otherwise,} \end{cases}$$

$$\beta_{N_\mathsf{o}+m}(\sigma) = \begin{cases} 1 & \text{if } \sigma = (0, N' - N) \\ 0 & \text{otherwise.} \end{cases}$$

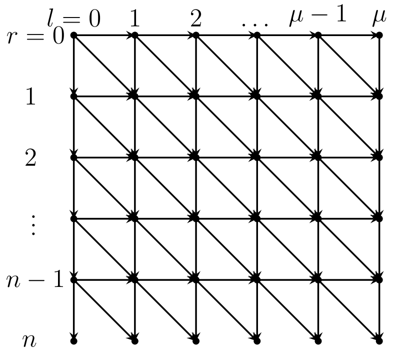


Figure I.3: The lattice used to compute $p(\boldsymbol{y}_{(i-1)n+d+1}^{in+d'}, d' \big| d, s, s')$.

The branch metric can be decomposed as

$$\gamma_i(\sigma, \sigma') = p(w_i)p(\boldsymbol{y}_{(i-1)n+d+1}^{in+d'}, d' \big| d, s, s'),$$

where $p(w_i)$ is the a priori probability of symbol w_i. The expression $p(\boldsymbol{y}_{(i-1)n+d+1}^{in+d'}, d' \big| d, s, s')$ can be efficiently computed using a lattice structure [19, 47] as explained as follows.

Define the lattice $\boldsymbol{F}_{n\times\mu}$ with $n+1$ rows corresponding to the transmitted sequence $\boldsymbol{x}_{(i-1)n+1}^{in} = \dot{\boldsymbol{x}} = (\dot{x}_1, \dots, \dot{x}_n)$ of length n, and $\mu + 1 = n + d' - d + 1$ columns corresponding to the received sequence $\boldsymbol{y}_{(i-1)n+d+1}^{in+d'} = \dot{\boldsymbol{y}} = (\dot{y}_1, \dots, \dot{y}_\mu)$ of length μ. A horizontal transition in the lattice represents an insertion with probability p_I/q, a vertical transition represents a deletion with probability p_D, while a diagonal transition represents a transmission. The last event has probability $p_\mathsf{T}(1 - p_\mathsf{S})$ if the corresponding elements in $\dot{\boldsymbol{y}}$ and $\dot{\boldsymbol{x}}$ match or probability $p_\mathsf{T}p_\mathsf{S}/(q-1)$ otherwise. Let the value of the lattice point in row r and column l be represented by $F_{r,l}$, see Fig. I.3. Then, for $0 < r < n$ and $0 < l \le \mu$, a lattice computation is defined recursively as

$$F_{r,l} = \frac{1}{q}p_\mathsf{I}F_{r,l-1} + p_\mathsf{D}F_{r-1,l} + Q(\dot{y}_l, \dot{x}_r)F_{r-1,l-1},$$

where

$$Q(\dot{y}, \dot{x}) = \begin{cases} p_\mathsf{T}\frac{p_\mathsf{S}}{q-1} & \text{if } \dot{y} \ne \dot{x} \\ p_\mathsf{T}(1 - p_\mathsf{S}) & \text{otherwise.} \end{cases}$$

The lattice computation is initialized as

$$F_{r,l} = \begin{cases} 1 & \text{if } r = 0, l = 0 \\ 0 & \text{otherwise,} \end{cases}$$

and for the last row ($r = n$), since the HMM does not allow insertions at the end of the transmitted sequence, the lattice computation becomes

$$F_{n,l} = p_\mathsf{D}F_{n-1,l} + Q(\dot{y}_l, \dot{x}_n)F_{n-1,l-1}.$$

Finally, $p(\boldsymbol{y}_{(i-1)n+d+1}^{in+d'}, d' \big| d, s, s') = F_{n,\mu}$.

3 Symbolwise MAP Decoding for Multiple Received Sequences

In this work, we propose two novel approaches to take advantage of the redundancy arising from multiple received sequences. The first approach, which we refer to as *joint decoding*, calculates the APPs exactly and works on a multi-sequence state-based trellis (or a joint trellis). The second approach, referred to as *separate decoding*, considers each sequence trellis separately and then combines their respective APPs. Although the first approach offers the best solution for the multiple received sequences problem, it comes with the drawback of very high complexity as will be shown later. As a result, the second approach is proposed as a more practical and less complex solution to the same problem, at the expense of a slight reduction in performance.

Joint Decoding

To calculate the APPs of code symbols w_i given multiple received sequences, we proceed as follows. We introduce a drift state vector $\boldsymbol{d}_i = (d_{i,1}, \ldots, d_{i,M})$ representing the drift at time instant i for each sequence. The resulting HMM has the combined $(M+1)$-dimensional state variables $\sigma_i = (s_i, d_{i,1}, \ldots, d_{i,M})$ with corresponding branch metric

$$\gamma_i(\sigma, \sigma') = p(w_i) \prod_{j=1}^{M} p\left((\boldsymbol{y}_j)_{(i-1)n+d_j+1}^{in+d'_j}, d'_j \middle| d_j, s, s' \right),$$

where $\boldsymbol{y}_j$, $1 \leq j \leq M$, is the j-th received sequence. This branch metric can be efficiently computed using the lattice implementation as $\gamma_i(\sigma, \sigma') = p(w_i) \prod_{j=1}^{M} F^j_{n,\mu_j}$, where F^j_{n,μ_j} is the result of the recursive lattice computation based on the substring $(\boldsymbol{y}_j)_{(i-1)n+d_j+1}^{in+d'_j}$ of the j-th received sequence $\boldsymbol{y}_j$ and $\mu_j = n + d'_j - d_j$ is its length. The APPs can then be computed applying the BCJR algorithm with initial and termination conditions

$$\alpha_0(\sigma) = \begin{cases} 1 & \text{if } \sigma = (0, 0, \ldots, 0) \\ 0 & \text{otherwise,} \end{cases}$$
$$\beta_{N_\mathrm{o}+m}(\sigma) = \begin{cases} 1 & \text{if } \sigma = (0, N'_1 - N, \ldots, N'_M - N) \\ 0 & \text{otherwise,} \end{cases}$$

where N'_j denotes the length of the j-th received sequence.

Separate Decoding

We propose a decoding approach that approximates the APPs by decoding each received sequence separately and combining the resulting single sequence APPs. The proposed separate decoding yields a significant reduction in complexity compared to joint decoding, as discussed in Section 3. More precisely, we approximate the joint APPs $p(w_i|\boldsymbol{y}_1, \dots, \boldsymbol{y}_M)$ given the single sequence APPs $p(w_i|\boldsymbol{y}_j)$, $1 \leq j \leq M$, as

$$p(w_i|\boldsymbol{y}_1, \dots, \boldsymbol{y}_M) \approx\propto \frac{\prod_{j=1}^{M} p(w_i|\boldsymbol{y}_j)}{p(w_i)^{M-1}}.$$

The individual APPs $p(w_i|\boldsymbol{y}_j)$ can be calculated via the BCJR decoder presented in Section 4. As we show in Appendix A, this approximation holds exactly for memoryless channels, which motivates the proposed separate decoding. However, it results here in a loss of performance, which we discuss in Sections 5.1 and 7. Note that the separation of the joint APPs could have been carried out as well at other stages in the decoding flow, e.g., directly between the channel and the inner code decoder or after decoding the outer code. However, we observed that for the considered parameters, the proposed stage yields the best results.

Complexity Analysis

In this subsection, we evaluate the complexity of joint and separate decoding. The number of performed operations by the inner BCJR decoding is mainly proportional to the number of edges in the trellis inferred by the HMM. In line with [21], for all methods we limit the drift to a fixed interval, $d_{i,j} \in [d_{\min}, d_{\max}]$, and the maximum number of insertions per symbol to $I_{\max}$. Let us denote by $\Delta = d_{\max} - d_{\min} + 1$ the total number of drift states. Moreover, we can quantify the total number of possible drift transisitions as $\delta = n(I_{\max} + 1) + 1$. Let ν be the number of binary memory elements of the convolutional encoder. The complexity to decode a single received sequence by the inner BCJR deocoder is

$$C_{\text{single}} = \frac{N}{n} 2^{\nu+k} \Delta\delta.$$

When considering the multiple sequence case, with M being the total number of sequences, the complexity of the presented methods can be computed as follows. For the separate decoding, the complexity is simply M times that of the single sequence decoding, i.e.,

$$C_{\text{sep}} = M \cdot C_{\text{single}} = \frac{N}{n} 2^{\nu+k} M \Delta\delta.$$

Since the joint decoding introduces a joint drift state vector, the decoding trellis grows exponentially in the number of sequences M. Thus, the complexity of joint decoding is

$$C_{\text{joint}} = \frac{N}{n} 2^{\nu+k} (\Delta\delta)^M.$$

Note that we do not take into account the effect of the trellis termination on the number of edges in the aforementioned decoding methods. Moreover, typical values for the product $\Delta\delta$ can be as high as ≈ 1000 in the DNA storage application. Therefore, the inner code decoding is the main contributor to the overall complexity and thus we have neglected the complexity of the outer code decoding.

For a fixed length N, increasing the number of sequences M has different impact in the scale of complexity. The complexity of the separate decoding method scales linearly with M while the complexity of the joint decoding approach scales exponentially with M. Therefore, it directly becomes evident that the joint decoding method is only practical for a small number of sequences. Hence, the reason we proposed the separate decoding algorithm. With separate decoding, one can increase the number of sequences M in order to match the performance of optimal joint decoding with a significantly lower complexity. As illustrated in later sections of the paper, the number of sequences needed to match the performance of joint decoding with $M = 2$ is low.

4 Concatenated Code Design

In this section, we discuss the optimization of the coding scheme. On a high level, we proceed as follows. We first design a novel inner TVC that is optimized in terms of the Levenshtein distance between two codewords in a block and also avoids overlaps between the codebooks of two adjacent blocks, improving its time variance. We evaluate the performance of this inner code by computing AIRs in Section 5 and compare with codes from the literature. For a selected inner code, we then proceed with optimizing an outer LDPC code using protographs and density evolution (DE). We also optimize the frozen symbols of an outer nonbinary polar code using Monte-Carlo methods.

Inner Block Code

Our goal is to improve on the performance of the inner code construction introduced in [21], which we refer to as the Davey-MacKay (DM) construction. The authors of [21] proposed to construct a binary inner block code of size 2^k and length n by selecting the 2^k vectors of lowest Hamming weight from all 2^n vectors of length n. In other words, the constructed code consists of the 2^k sparsest length-n binary vectors. We refer to the combination of the block code arising from the DM construction and a random offset sequence as a *watermark* code.

The use of a random offset sequence is essential as it helps the inner code in maintaining synchronization with the transmitted sequence. More precisely, the random sequence helps in tracking the codeword boundaries between consecutive transmissions of inner code codewords. This in turn helps the inner code to avoid confusing adjacent codewords, especially in the presence of a large number of insertions and deletions. The authors in [21] state that the more random the watermark code is, the better the synchronization, which motivated their sparse code construction, as it alters little the random sequence. However, for low IDS probabilities, where loss of synchronization is less probable, the dominating factor of the inner code performance is its *edit distance* profile. The edit distance between two codewords, denoted by d_{e}, is defined as the minimum number of IDS errors required to change one codeword into another. Hence, an inner code with good minimum d_{e}, $d_{\mathsf{e}}^{\min}$, will perform well in that region. We observe this phenomenon in the AIR results in Section 5 (for an in-depth discussion on the synchronization ability and performance of inner codes, we refer the reader to that section). In the special case when no substitutions are allowed, the edit distance is typically referred to as the Levenshtein distance, which we denote by d_{L}.

It was shown in [48] that the DM construction is far from optimal due to its poor Levenshtein distance profile. The authors in [49] considered the so-called *weighted* Levenshtein distance and improved its profile for the watermark code, hence the code performance, by first building a set of sequences that when added to the codewords, improve the overall distance profile of the code. Secondly, for every transmission, a sequence is randomly picked from this set. However, the performance is still relatively poor due to a low $d_{\mathsf{e}}^{\min}$. In [47, 50], it was proposed to use codewords from Varshamov-Tenegol'ts (VT) codes [51], known for their good IDS correction capabilities, to construct an inner code. In particular, in [47], the authors chose a set of codewords from a VT code using a simulated annealing algorithm [52] that targets the minimization of the so-called change probability of the set. One more issue to combat comes from the requirement of using a random offset sequence which does not destroy the edit distance properties of the inner code. This issue was previously addressed by the same authors in [50] where they proposed to randomly choose an offset sequence only from the set of sequences that when added to the codewords of the inner code do not alter its $d_{\mathsf{e}}^{\min}$.[1] However, such inner codes perform better when the codeword boundaries between consecutive transmissions are known, which is typically not the case. As a result, in the later work [47], TVCs, which are composed of several alternating codebooks found by minimizing the change probability through a simulated annealing search, were proposed. One issue that may arise is that using the simulated annealing algorithm does not guarantee all codebooks to have the best $d_{\mathsf{e}}^{\min}$.

Here, we consider nonbinary TVCs, where the codebooks are found by searching for cliques in an undirected graph as described below. Searching for codebooks in this way guarantees that they all have the same $d_{\mathsf{e}}^{\min}$. First, an undirected graph with a vertex for each vector in Σ_q^n is constructed. Then, we search for a maximum clique in that graph. In general, finding a maximum clique in a graph is an NP-hard problem. However, for small graphs, there exist several efficient algorithms that can be used [53–55]. Here, we use the branch-and-bound algorithm proposed in [54]. Formally, let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be an undirected graph, where $\mathcal{V}$ denotes the set of vertices and $\mathcal{E}$ the set of edges. Each vertex $v = v(\boldsymbol{c}) \in \mathcal{V}$ represents a nonbinary vector $\boldsymbol{c} \in \Sigma_q^n$, and there is an edge between vertices $v(\boldsymbol{c}_i)$ and $v(\boldsymbol{c}_j)$ if and only if $d_{\mathsf{e}}(\boldsymbol{c}_i, \boldsymbol{c}_j) \geq d_{\mathsf{e}}^{\min}$, where $d_{\mathsf{e}}^{\min}$ is a prescribed minimum edit distance.[2] Then, an $[n, k]_q$ (nonlinear) block code with edit distance at least equal to $d_{\mathsf{e}}^{\min}$ corresponds to a clique of size 2^k in $\mathcal{G}$.

The branch-and-bound algorithm searches for a set of cliques (or codebooks) of a given size, and from this set we select t codebooks that overlap as little as possible using a heuristic approach. The t selected codebooks can then either be repeated periodically or randomly in order to construct a TVC. In this work, we only consider the special case of no substitutions when constructing a TVC.

[1]Note that in [50] the edit distance is referred to as the Levenshtein distance.

[2]Here, $d_{\mathsf{e}}(\cdot, \cdot)$ denotes the edit distance between its first and second argument. Moreover, when computing the distance, according to the IDS channel model, we do not allow for insertions at the end.

Table I.1: Designed TVC With 4 Codebooks

Codebook 1	Codebook 2	Codebook 3	Codebook 4
$(0,0,0,0)$	$(0,0,0,1)$	$(0,0,3,0)$	$(0,0,0,3)$
$(0,0,2,2)$	$(0,0,3,3)$	$(0,1,1,1)$	$(0,0,2,2)$
$(0,3,2,3)$	$(0,2,1,2)$	$(0,2,3,2)$	$(0,3,2,3)$
$(1,0,1,0)$	$(1,0,2,0)$	$(0,3,1,3)$	$(1,0,1,1)$
$(1,1,1,1)$	$(1,1,2,2)$	$(1,0,0,1)$	$(1,1,0,0)$
$(1,1,3,3)$	$(1,1,3,1)$	$(1,2,0,2)$	$(1,1,3,3)$
$(1,2,3,2)$	$(1,3,0,3)$	$(1,3,2,3)$	$(1,2,3,2)$
$(2,0,2,1)$	$(2,0,0,2)$	$(2,2,0,0)$	$(2,1,2,1)$
$(2,1,2,0)$	$(2,2,0,3)$	$(2,2,1,3)$	$(2,2,0,0)$
$(2,2,2,2)$	$(2,2,1,1)$	$(2,2,2,2)$	$(2,2,2,3)$
$(2,2,3,3)$	$(2,3,1,3)$	$(2,3,0,3)$	$(2,3,0,3)$
$(3,0,3,1)$	$(3,0,1,0)$	$(3,0,0,2)$	$(3,0,0,1)$
$(3,1,3,0)$	$(3,2,2,2)$	$(3,2,1,2)$	$(3,1,3,0)$
$(3,2,0,0)$	$(3,2,3,0)$	$(3,3,1,1)$	$(3,2,0,2)$
$(3,3,2,2)$	$(3,3,1,1)$	$(3,3,2,0)$	$(3,2,3,1)$
$(3,3,3,3)$	$(3,3,3,2)$	$(3,3,3,3)$	$(3,3,3,3)$

Hence, we consider the Levenshtein distance. In Table I.1, we list $t = 4$ quaternary codebooks, each of size 16, length $n = 4$, and with minimum Levenshtein distance $d_{\mathsf{L}}^{\min} = 4$. Note that for these code parameters, finding 4 disjoint codebooks is difficult. Hence, some of the codebooks overlap and the number of distinct codewords is 56. The resulting TVC is a $[4,4,4]_4$ TVC.

In Section 7, we compare the performance of our concatenated coding schemes using a quaternary watermark code and the quaternary TVC code in Table I.1 as inner codes. We further consider the use of a convolutional inner code. Here, we do not extend the inner code optimization to the convolutional code, but rather pick the same code as in [47], which is the one corresponding to the generator polynomial $g = [5,7]_{\text{oct}}$. By grouping trellis sections together in the decoding stage, we can interpret this convolutional code also as a nonbinary inner code, thus forming APPs for higher order field sizes.

Outer Code: Low-Density Parity-Check Code

We consider protograph LDPC codes as they facilitate achieving lower error floors compared to unstructured codes. Formally, a protograph is a small multi-edge-type graph with n_{p} variable-node (VN) types and r_{p} check-node (CN) types. A protograph can be represented by a base matrix

$$\boldsymbol{B} = \begin{pmatrix} b_{0,0} & b_{0,1} & \dots & b_{0,n_{\mathsf{p}}-1} \\ b_{1,0} & b_{1,1} & \dots & b_{1,n_{\mathsf{p}}-1} \\ \vdots & \vdots & \dots & \vdots \\ b_{r_{\mathsf{p}}-1,0} & b_{r_{\mathsf{p}}-1,1} & \dots & b_{r_{\mathsf{p}}-1,n_{\mathsf{p}}-1} \end{pmatrix},$$

where entry $b_{i,j}$ is an integer representing the number of edge connections from a type-i VN to a type-j CN. A parity-check matrix $\boldsymbol{H}$ of an LDPC code can then be constructed by lifting the base matrix $\boldsymbol{B}$ by replacing each nonzero (zero) $b_{i,j}$ with a $Q_{\mathsf{p}} \times Q_{\mathsf{p}}$ circulant (zero) matrix with row and column weight equal to $b_{i,j}$. The circulant matrices are picked in order to maximize the girth of the corresponding Tanner graph by using the progressive edge-growth algorithm [56]. The resulting lifted parity-check matrix, of dimensions $Q_{\mathsf{p}} r_{\mathsf{p}} \times Q_{\mathsf{p}} n_{\mathsf{p}}$, defines an LDPC code of length $Q_{\mathsf{p}} n_{\mathsf{p}}$ and dimension at least $Q_{\mathsf{p}}(n_{\mathsf{p}} - r_{\mathsf{p}})$. To construct a nonbinary code from the lifted matrix, we randomly assign nonzero entries from $\mathbb{F}_{q_{\mathsf{o}}}$ to the edges of the corresponding Tanner graph.

We optimize the protograph using DE. Particularly, we consider the DE algorithm proposed in [57] for the optimization of binary LDPC codes for intersymbol interference channels, extended to nonbinary protographs. The algorithm is based on estimating the probability density functions of the messages from the inner code to the outer LDPC code via Monte-Carlo simulations of the inner decoder and channel detector. When performing the DE, we assume uniformly random protograph edge weights. The optimized protograph is then chosen as the one yielding the best iterative decoding threshold, p_{th}. The iterative decoding threshold predicts the asymptotic performance of the code, in the sense that the bit error probability under iterative decoding approaches zero for $p_{\mathsf{I}} = p_{\mathsf{D}} < p_{\mathsf{th}}$, for a given p_{S}, as the block length goes to infinity. The optimization problem is hence

$$\begin{aligned} \arg\max_{\boldsymbol{B}} \quad & p_{\mathsf{th}} \\ \text{s.t.} \quad & b_{i,j} \leq b_{\max}, \quad \forall\,(i,j), \\ & n_{\mathsf{p}} \leq n_{\mathsf{p}}^{\max}, \\ & \frac{n_{\mathsf{p}} - r_{\mathsf{p}}}{n_{\mathsf{p}}} = R_{\mathsf{o}}, \end{aligned}$$

where $b_{\max}$ is the maximum allowed circulant weight, $n_{\mathsf{p}}^{\max}$ is the maximum allowed number of different VN types, and R_{o} is the design rate of the LDPC code.

One issue with optimizing over protographs is that neither the dimensions of the protograph nor the number of edge connections have a natural limit. To make the search feasible, we restrict the search to $n_{\mathsf{p}}^{\max} = 4$ and $b_{\max} = 2$.

Outer Code: Polar Code

As another choice for the nonbinary outer code, we consider the polar code construction from [58] over the field $\mathbb{F}_{q_{\mathsf{o}}}$. The construction's main feature is the extension of Arıkan's binary kernel [59] to a nonbinary kernel

$$\boldsymbol{K} = \begin{pmatrix} 1 & 0 \\ \alpha & \beta \end{pmatrix},$$

where $\alpha, \beta \in \mathbb{F}_{q_{\mathsf{o}}}$. The encoding can be written as $\boldsymbol{w} = \boldsymbol{u}'\boldsymbol{K}^{\otimes \log_{q_{\mathsf{o}}} N_{\mathsf{o}}}$, where $(\cdot)^{\otimes}$ denotes the Kronecker power and $\boldsymbol{u}' \in \mathbb{F}_{q_{\mathsf{o}}}^{N_{\mathsf{o}}}$. One can determine a mapping from the actual information vector $\boldsymbol{u} \in \mathbb{F}_{q_{\mathsf{o}}}^{K}$ to $\boldsymbol{u}'$ by fixing $N_{\mathsf{o}} - K$ positions, denoted as *frozen positions*, to a specific symbol and padding the vector $\boldsymbol{u}$ into the remaining positions. For decoding, we use successive cancellation list (SCL) decoding over $\mathbb{F}_{q_\mathsf{o}}$ as described in [58, 60]. To boost the decoding performance of the SCL decoder, a cyclic redundancy check (CRC) of bit length ℓ_{CRC} is applied. To ensure comparability for fixed rates, the number of frozen positions is hence reduced to $N_\mathsf{o} - K - {\ell_{\mathrm{CRC}}}/{\log(q_\mathsf{o})}$.[3]

In this work, we optimize two components of the nonbinary polar code for our channel using Monte-Carlo simulations: the kernel selection and the frozen positions. For the kernel, we optimize the single-level polarization effect as described in [58], which means that the optimization step is only done via considering a single 2×2 kernel. We revise the method shortly here while directly applying our channel model. We treat the inner code including the IDS channels as an auxiliary channel. For this channel, we generate random input sequences $\boldsymbol{w}$ uniformly at random and transmit them via the auxiliary channel to gain APP samples $p(w_i|\boldsymbol{y})$, where $\boldsymbol{y} = (\boldsymbol{y}_1, \ldots, \boldsymbol{y}_M)$.

First, fix $\alpha, \beta \in \mathbb{F}_{q_\mathsf{o}}$ and do the following.

1. Choose an input vector $\boldsymbol{u'} = (u_1, u_2) = (0, u_2)$, where $u_2 \in \mathbb{F}_{q_\mathsf{o}}$ is chosen uniformly at random.

2. Calculate $\boldsymbol{w} = (w_1, w_2) = (u_1 + \alpha u_2, \beta u_2)$.

3. Pick uniformly at random two APP samples $p(w_1|\boldsymbol{y})$ and $p(w_2|\boldsymbol{y})$ under the constraint that these correspond to the values in the vector $\boldsymbol{w}$.

4. Calculate $p(u_2|u_1 = 0, \boldsymbol{y})$ using the APP samples.

Consequently, we have that $p(u_2|u_1 = 0, \boldsymbol{y})$ is a random variable depending only on the auxiliary channel. Hence, we can find the best ratio ${\alpha}/{\beta}$ via Monte-Carlo simulations by choosing

$$\frac{\alpha}{\beta} = \operatorname*{arg\,min}_{\frac{\alpha}{\beta} \in \mathbb{F}_{q_\mathsf{o}}} \mathbb{E}\left(1 - p(u_2|u_1 = 0, \boldsymbol{y})\right),$$

where the expectation is taken with respect to the random variable on the auxiliary channel with fixed inner code and channel parameters.

This optimization is dependent on the choice of the inner code and may vary for different codes. In Fig. I.4, the failure rate $\mathbb{E}\left(1 - p(u_2|u_1 = 0, \boldsymbol{y})\right)$ is depicted as a function of the ratio ${\alpha}/{\beta}$ for various insertion and deletion probabilities $p_\mathsf{I} = p_\mathsf{D} = p$, with $p_\mathsf{S} = 0$ and $M = 1$. We denote the field elements $\alpha, \beta \in \mathbb{F}_{q_\mathsf{o}}$ as integers such that their binary transformation corresponds to the coefficients of the respective polynomial representation. Since the goal is to find the best kernel selection, i.e., minimize the failure rate over all possible values p for a given inner code, good choices of the kernel parameters for the depicted setup would be $\alpha = 3, 5, 10, 11, 12$ while fixing $\beta = 1$.

Moreover, we determine the frozen positions via a genie-aided rate-one polar code with a fixed inner coding scheme by using Monte-Carlo simulations.

[3]To simplify notation, in the rest of the paper, log denotes logarithm to the base 2.

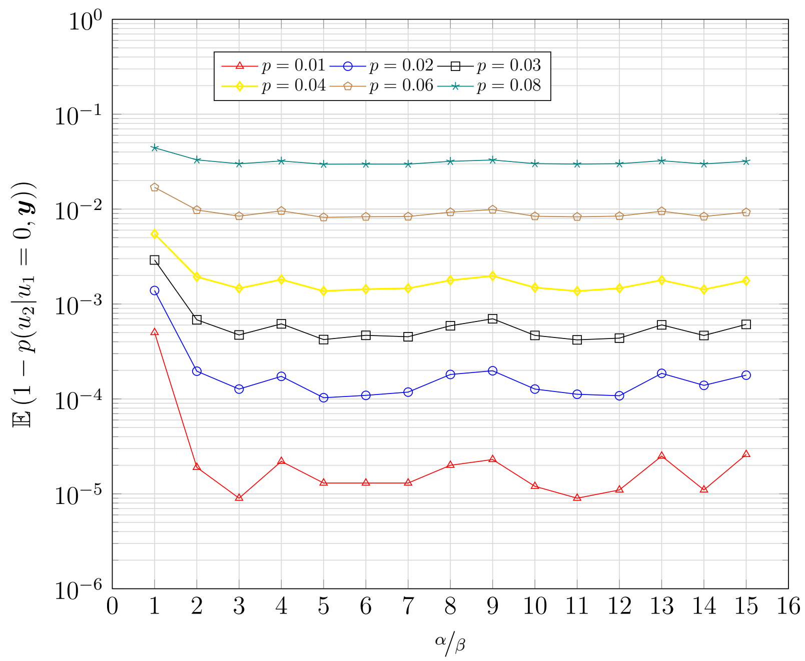


Figure I.4: Monte-Carlo simulations of $\mathbb{E}\left(1 - p(u_2|u_1 = 0, \boldsymbol{y})\right)$ for $M = 1$, $p_\mathsf{S} = 0$, and different $p_\mathsf{I} = p_\mathsf{D} = p$ on different ratios α/β with fixed $\beta = 1$, $q_\mathsf{o} = 16$, and primitive polynomial $x^4 + x + 1$. The inner code is a convolutional code with generator polynomial $g = [5, 7]_{\text{oct}}$ with a random offset sequence.

5 Achievable Information Rates

We now turn to presenting AIRs over multiple IDS channels to benchmark our coding schemes. We follow two standard approaches for the estimation of AIRs over hidden Markov channels, i.e., channels whose output is the output of a HMM. The first approach is to compute the so-called *BCJR-once* rate [57, 61, 62], which is defined as the symbolwise mutual information between the input process and its corresponding log-likelihood ratios, produced by a symbolwise MAP detector (BCJR algorithm). The second approach is based on [63, 64] and uses concentration properties of Markov chains to estimate the mutual information between channel input and output.

5.1 BCJR-Once Rate

The BCJR-once rate [57, 61, 62], which we denote by $R_{\text{BCJR-once}}$, serves as an information rate that can be achieved using an inner decoder that passes once symbolwise APPs to an appropriate outer decoder that is unaware of possible correlations between the symbolwise estimates. The outer and inner decoders hereby perform no iterations. The BCJR-once rate can be derived by computing an AIR of a mismatched decoder as follows. While computing achievable rates for mismatched decoders is well understood [65], we shortly review the most important derivations for completeness and convenience of the readers. To this end, denote by $q(\boldsymbol{w}|\boldsymbol{y})$, where $\boldsymbol{y} = (\boldsymbol{y}_1, \ldots, \boldsymbol{y}_M)$, an arbitrary decoding metric that is a valid distribution, i.e., $\sum_{\boldsymbol{w}} q(\boldsymbol{w}|\boldsymbol{y}) = 1$ for all $\boldsymbol{y}$ and satisfies $q(\boldsymbol{w}|\boldsymbol{y}) = 0$ if $p(\boldsymbol{w}|\boldsymbol{y}) = 0$. In the following, let $H(\cdot)$ denote the entropy function. We obtain for the mutual information between the message $\boldsymbol{w}$ and output $\boldsymbol{y}$,[4]

$$\begin{aligned} I(\boldsymbol{w};\boldsymbol{y}) &= H(\boldsymbol{w}) - H(\boldsymbol{w}|\boldsymbol{y}) \\ &= H(\boldsymbol{w}) + \sum_{\boldsymbol{w},\boldsymbol{y}} p(\boldsymbol{w},\boldsymbol{y}) \log p(\boldsymbol{w}|\boldsymbol{y}) \\ &= H(\boldsymbol{w}) + \sum_{\boldsymbol{w},\boldsymbol{y}} p(\boldsymbol{w},\boldsymbol{y}) \left(\log q(\boldsymbol{w}|\boldsymbol{y}) + \log \frac{p(\boldsymbol{w}|\boldsymbol{y})}{q(\boldsymbol{w}|\boldsymbol{y})} \right) \\ &\geq H(\boldsymbol{w}) + \sum_{\boldsymbol{w},\boldsymbol{y}} p(\boldsymbol{w},\boldsymbol{y}) \log q(\boldsymbol{w}|\boldsymbol{y}), \end{aligned} \tag{I.2}$$

where the last inequality is due to identifying the sum over the second summand as a Kullback-Leibler divergence, which is nonnegative. Next, we concretize the mismatched metrics for the BCJR-once rate and separate decoding, including a description of how we numerically estimate the above terms. Using the mismatched decoding setup, we can obtain the BCJR-once rate by computing the AIR of a mismatched decoder with decoding metric

$$q_{\text{BCJR}}(\boldsymbol{w}|\boldsymbol{y}) = \prod_{i=1}^{N_\text{o}+m} q(w_i|\boldsymbol{y}).$$

Note that here also the symbolwise a posteriori likelihoods $q(w_i|\boldsymbol{y})$ are considered mismatched, due to the fact that their trellis-based computation is not exact as we truncate some of the edges and states for complexity reasons. For the case of decoding multiple received sequences with separate decoding, we use the mismatched metric

$$q_{\text{BCJR-sep}}(\boldsymbol{w}|\boldsymbol{y}_1, \ldots, \boldsymbol{y}_M) \propto \prod_{i=1}^{N_\text{o}+m} \prod_{j=1}^{M} q(w_i|\boldsymbol{y}_j),$$

where the proportionality constant is chosen such that the decoding metric is a valid distribution, i.e., $\sum_{\boldsymbol{w}} q_{\text{BCJR-sep}}(\boldsymbol{w}|\boldsymbol{y}_1, \ldots, \boldsymbol{y}_M) = 1$. Defining the associated mismatched log-likelihood ratios

$$L_i^{\text{BCJR-sep}}(a) = \sum_{j=1}^{M} \ln \frac{q(w_i = a|\boldsymbol{y}_j)}{q(w_i = 0|\boldsymbol{y}_j)}$$

for all $a \in \mathbb{F}_{q_\text{o}}$, we can combine the mismatched metric with the mismatched log-likelihood ratios to obtain

$$q_{\text{BCJR-sep}}(\boldsymbol{w}|\boldsymbol{y}_1, \ldots, \boldsymbol{y}_M) = \prod_{i=1}^{N_\text{o}+m} \frac{\mathrm{e}^{L_i^{\text{BCJR-sep}}(w_i)}}{\sum_{a \in \mathbb{F}_{q_\text{o}}} \mathrm{e}^{L_i^{\text{BCJR-sep}}(a)}},$$

[4]With some abuse of notation, for simplicity, we do not distinguish notationwise between random variables and their realizations. Hence, e.g., $\boldsymbol{w}$ in $H(\boldsymbol{w})$ denotes a random vector, while $p(\boldsymbol{w})$ denotes the probability of a given realization.

Table I.2: Inner Code Scheme Selection

Scheme	Inner code	Gen. polynomial	Alt. pattern	Rate
CC-1	$(2,2,2)_4$ Conv. code with RS	g_1*	-	0.98
CC-2	$(4,4,2)_4$ Conv. code with RS	g_2*	-	0.98
WM	$[4,4,1]_4$ Watermark code	-	-	1.0
TVC-1	$[4,4,4]_4$ TVC	-	Random*	1.0
TVC-2	$[4,4,4]_4$ TVC with RS	-	CB1 to CB4*	1.0

*The alternating pattern of the TVC-1 scheme is done by choosing randomly the 4 codebooks, denoted by CB1-CB4, from Table I.1 and avoiding consecutive codebooks. For the TVC-2 scheme, it is simply done by repeating CB1 to CB4 in a round Robin fashion. The generator polynomials are $g_1 = [5,7,12,16]_{\text{oct}}$ and $g_2 = [24,34,70,120,160,240,340]_{\text{oct}}$, where g_1 and g_2 correspond to combining 2 and 4 consecutive trellis sections of $g = [5,7]_{\text{oct}}$, respectively. RS is shorthand for random sequence.

where the denominator has been chosen such that we obtain a valid distribution. Plugging the result into (I.2) yields

$$\begin{aligned} I(\boldsymbol{w};\boldsymbol{y}) &\geq H(\boldsymbol{w}) + \sum_{\boldsymbol{w},\boldsymbol{y}} p(\boldsymbol{w},\boldsymbol{y}) \log q_{\text{BCJR-sep}}(\boldsymbol{w}|\boldsymbol{y}) \\ &= H(\boldsymbol{w}) + \sum_{i=1}^{N_{\mathsf{o}}+m} \sum_{w_i,\boldsymbol{y}} p(w_i,\boldsymbol{y}) \log \frac{\mathrm{e}^{L_i^{\text{BCJR-sep}}(w_i)}}{\sum_{a\in\mathbb{F}_{q_{\mathsf{o}}}} \mathrm{e}^{L_i^{\text{BCJR-sep}}(a)}}. \end{aligned}$$

For independent and uniform inputs, we obtain $H(\boldsymbol{w}) = (N_{\mathsf{o}}+m)\log q_{\mathsf{o}}$. Under the assumption that the expectation above obeys asymptotic ergodicity, we conclude that, for large N_{o}, we can estimate the BCJR-once rate by sampling a long input sequence $\boldsymbol{w}$ and corresponding output sequence $\boldsymbol{y}$ and compute the log-likelihood ratios $L_i^{\text{BCJR-sep}}(a)$, allowing to conclude with the estimate

$$R_{\text{BCJR-once}} \approx R_{\mathsf{i}} \log q_{\mathsf{o}} + \frac{R_{\mathsf{i}}}{N_{\mathsf{o}}+m} \sum_{i=1}^{N_{\mathsf{o}}+m} \log \frac{\mathrm{e}^{L_i^{\text{BCJR-sep}}(w_i)}}{\sum_{a\in\mathbb{F}_{q_{\mathsf{o}}}} \mathrm{e}^{L_i^{\text{BCJR-sep}}(a)}}.$$

Note that in the above expression, we have taken into account that the BCJR-once rate is measured in terms of information bits per channel use and we thus normalize the above sum with respect to the number of channel uses $(N_{\mathsf{o}}+m)/R_{\mathsf{i}}$. This expression has previously been derived in, e.g., [65, 66] and is also valid for mismatched log-likelihood ratios, as we have shown in our derivation. It is important to mention that to compute this expression, it is necessary to use both the mismatched log-likelihood ratios $L_i^{\text{BCJR-sep}}(a)$ and the original input sequence $\boldsymbol{w}$. Other approaches to compute the BCJR-once rate [67, 68] rely on the correctness of the computed log-likelihood ratios. In that case, it is possible to estimate the rate using a formula that only depends on the log-likelihood ratios and *not* the input sequence $\boldsymbol{w}$, as pointed out in [66]. However, in our case we compute mismatched log-likelihood ratios, and thus such an approach might give incorrect results.

Fig. I.5 compares the BCJR-once rates of some inner codes we will use in this paper along with an inner code designed in [42] for the single sequence scenario.

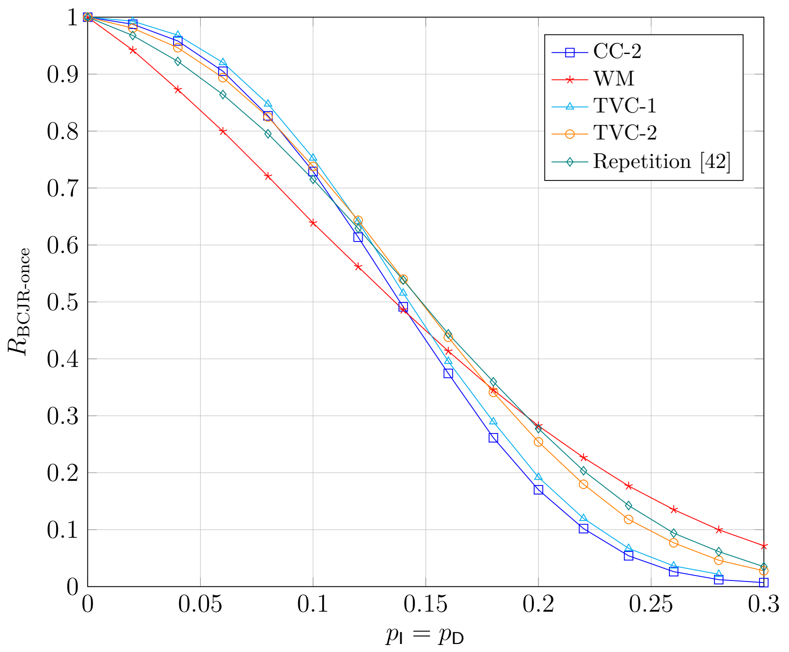


Figure I.5: BCJR-once rates $R_{\text{BCJR-once}}$ versus $p_{\mathsf{I}} = p_{\mathsf{D}}$ with $p_{\mathsf{S}} = 0$ and $M = 1$ using different inner codes.

The parameters of the inner codes are given in Table I.2. These plots have been simulated with 10^5 channel uses and have been smoothed over several iterations. Interestingly, we can observe that it seems that codes that perform well for small insertion and deletion error probabilities perform poor for large error rates and vice versa. While we do not conjecture this to be a general result, this can be explained for our codes at hand as follows. For small error probabilities, the distance spectrum within a single code block, i.e., the set of possible length-n words that can be generated within one trellis section, is the dominant property influencing the performance, as synchronization between blocks is not a problem. However, on the other hand, for large error rates, codes that can retain synchronization between blocks provide the highest BCJR-once rates. In particular, codes for which the codewords in one block have a small $d_{\mathsf{L}}^{\min}$, such as the watermark code, synchronize well, as the structure of a block is less variable over codewords within one block, helping the receiver to know the rough structure of this block.

The above exposition also indicates in which cases the employment of a random offset sequence can improve the performance of a code. In particular, adding a random sequence has three effects on the inner code. First, while the Hamming distance spectrum is invariant to offsets, the Levenshtein distance spectrum can both improve or worsen when adding an offset sequence. Second, a convolutional code, as a block code, is block-cyclic, meaning that shifting an inner codeword by n symbols to the left or right will again be a valid codeword. It is not surprising that such a property is problematic for synchronization and thus a random offset

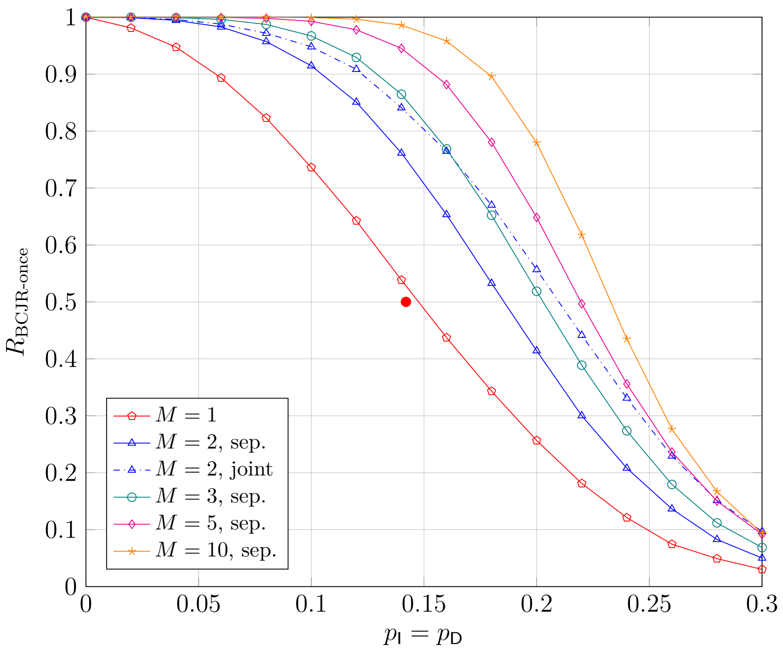


Figure I.6: BCJR-once rates $R_{\text{BCJR-once}}$ versus $p_{\mathsf{I}} = p_{\mathsf{D}}$ with $p_{\mathsf{S}} = 0$ using the TVC-2 inner coding scheme with different number of transmissions M. The red solid circle indicates the iterative decoding threshold of our concatenated coding scheme with the optimized outer protograph LDPC code from Section 6.2 for $M = 1$.

sequence, which destroys the cyclicity, improves the performance at higher error rates. We can identify this behavior for the TVC-2 inner coding scheme, whose performance worsens compared to TVC-1 for small error probabilities, as the carefully designed Levenshtein distance spectrum is destroyed when adding a random sequence. However, for large error probabilities, the random sequence helps as it produces a larger spectrum of possible codewords, increasing the distinguishability between close blocks. We refer the reader to Section 4 for an in-depth discussion of inner code design.

Fig. I.6 shows the BCJR-once rates for the TVC-2 inner coding scheme with different number of transmissions M. It is observed that multiple transmissions can significantly improve the BCJR-once rates. In particular, even going from one to two sequences, we already see a notable difference in terms of information rates. In addition, as expected, we observe a reduced achievable rate with separate decoding compared to joint decoding. However, this loss can be compensated by decoding more received sequences. For instance, for $p_{\mathsf{I}} = p_{\mathsf{D}} = 0.16$, the BCJR-once rate of separate decoding for $M = 3$ is close to that of joint decoding for $M = 2$, but with a much lower decoding complexity. In Fig. I.6, we also show the iterative decoding threshold of our concatenated coding scheme (red solid circle) using the TVC-2 inner coding scheme concatenated with an optimized outer protograph LDPC code for $M = 1$. We observe that (in the asymptotic limit of infinitely-large block

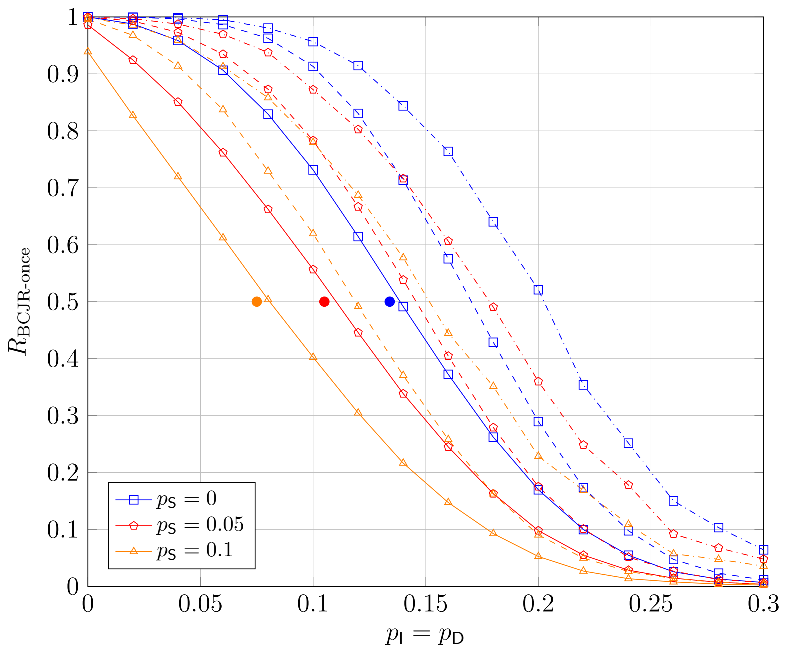


Figure I.7: BCJR-once rates $R_{\text{BCJR-once}}$ versus $p_{\mathsf{I}} = p_{\mathsf{D}}$ with $p_{\mathsf{S}} = 0, 0.05, 0.1$ using the CC-2 inner coding scheme with $M = 1$ and $M = 2$. Solid lines are for $M = 1$, dashed lines are for $M = 2$ and separate decoding, and dash dotted lines are for $M = 2$ and joint decoding. The solid circles indicate the iterative decoding thresholds of our concatenated coding scheme with the optimized outer protograph LDPC codes from Section 6.5 for $M = 1$.

length) the proposed coding scheme performs very close to the corresponding AIR. The optimized base matrix found by computer search along with the corresponding iterative decoding threshold are given in Section 6.2.

In Fig. I.7, we show BCJR-once rates with an inner convolutional code (CC-2) for both single and double sequence transmission (separate and joint decoding) when $p_{\mathsf{S}} = 0, 0.05, 0.1$. As can be observed from the figure, the achievable rate loss of separate decoding with increasing p_{S} is about the same for $M = 1$ and $M = 2$. Moreover, the gap between separate and joint decoding (for $M = 2$) stays approximately the same for different p_{S}, which shows that the proposed separate decoding approach is robust to substitution errors. In the figure we also show the iterative decoding thresholds of our concatenated coding scheme (solid circles) using the CC-2 inner coding scheme with optimized outer protograph LDPC codes, optimized individually for each p_{S}, for $M = 1$ (see Section 6.5 for the optimized base matrices).

5.2 Mutual Information Rate

A method to compute the mutual information for a given coding scheme was introduced in [63, 64], where the mutual information between an input process

$\boldsymbol{w} = (w_1, w_2, \dots)$ and an output process $\boldsymbol{y} = (y_1, y_2, \dots)$, $I(\boldsymbol{w}; \boldsymbol{y})$, is computed via trellis-based simulations. Given that a source/channel decoding trellis exists for a coding scheme, the mutual information point

$$I(w; y) \triangleq \lim_{N_\mathsf{o} \to \infty} \frac{1}{N_\mathsf{o} + m} I(\boldsymbol{w}; \boldsymbol{y}), \tag{I.3}$$

which is an AIR, can be computed using the forward recursion of the BCJR algorithm on the given trellis. Since

$$I(\boldsymbol{w}; \boldsymbol{y}) = H(\boldsymbol{w}) + H(\boldsymbol{y}) - H(\boldsymbol{w}, \boldsymbol{y})$$

and $-\log p(\boldsymbol{w})$, $-\log p(\boldsymbol{y})$, and $-\log p(\boldsymbol{w}, \boldsymbol{y})$ converge with probability 1 to $H(\boldsymbol{w})$, $H(\boldsymbol{y})$, and $H(\boldsymbol{w}, \boldsymbol{y})$, respectively, for long sequences, $I(w; y)$ can be estimated by

$$\hat{I}(w; y) = -\frac{1}{N_\mathsf{o} + m} \log p(\boldsymbol{w}) - \frac{1}{N_\mathsf{o} + m} \log p(\boldsymbol{y}) + \frac{1}{N_\mathsf{o} + m} \log p(\boldsymbol{w}, \boldsymbol{y})$$

when N_o is very large.

This approach can be used for our coding schemes as well. For the case of single sequence transmission, given an input sequence $\boldsymbol{w}$ to the inner code and a corresponding output sequence $\boldsymbol{y}$ from the channel, $\log p(\boldsymbol{y})$, $\log p(\boldsymbol{w}, \boldsymbol{y})$, and $\log p(\boldsymbol{w})$ can be computed as follows. First, compute $\log p(\boldsymbol{y})$ from

$$p(\boldsymbol{y}) = \sum_\sigma p\Big(\boldsymbol{y}_1^{(N_\mathsf{o}+m)n+d}, \sigma\Big) \overset{(a)}{=} \sum_\sigma \alpha_{N_\mathsf{o}+m}(\sigma),$$

where (a) follows since $\alpha_i(\sigma) = p\Big(\boldsymbol{y}_1^{in+d}, \sigma\Big)$. Hence, it can be computed using the recursion in (I.1). Second, $\log p(\boldsymbol{w})$ and $\log p(\boldsymbol{w}, \boldsymbol{y})$ can be computed using a recursion in a similar manner. In particular, the recursion for computing $p(\boldsymbol{w}, \boldsymbol{y})$ is

$$\alpha_i^{(\mathsf{w},\mathsf{y})}(\sigma) = \sum_{\hat{\sigma}} \alpha_{i-1}^{(\mathsf{w},\mathsf{y})}(\hat{\sigma}) \gamma_i(\hat{\sigma}, \sigma),$$

where the summation is over all states $\hat{\sigma}$ with an outgoing edge to σ labeled with the input sequence symbol w_i at time i. In other words, the recursion for $p(\boldsymbol{w}, \boldsymbol{y})$ does not marginalize the input sequence $\boldsymbol{w}$ like in $p(\boldsymbol{y})$. Then,

$$p(\boldsymbol{w}, \boldsymbol{y}) = \sum_\sigma \alpha_{N_\mathsf{o}+m}^{(\mathsf{w},\mathsf{y})}(\sigma).$$

The recursion for computing $p(\boldsymbol{w})$ is

$$\alpha_i^{(\mathsf{w})}(\sigma) = \sum_{\hat{\sigma}} \alpha_{i-1}^{(\mathsf{w})}(\hat{\sigma}) p(w_i, \sigma | \hat{\sigma}),$$

where again the summation is over all states $\hat{\sigma}$ with an outgoing edge to σ labeled with the input sequence symbol w_i at time i. Then,

$$p(\boldsymbol{w}) = \sum_\sigma \alpha_{N_\mathsf{o}+m}^{(\mathsf{w})}(\sigma).$$

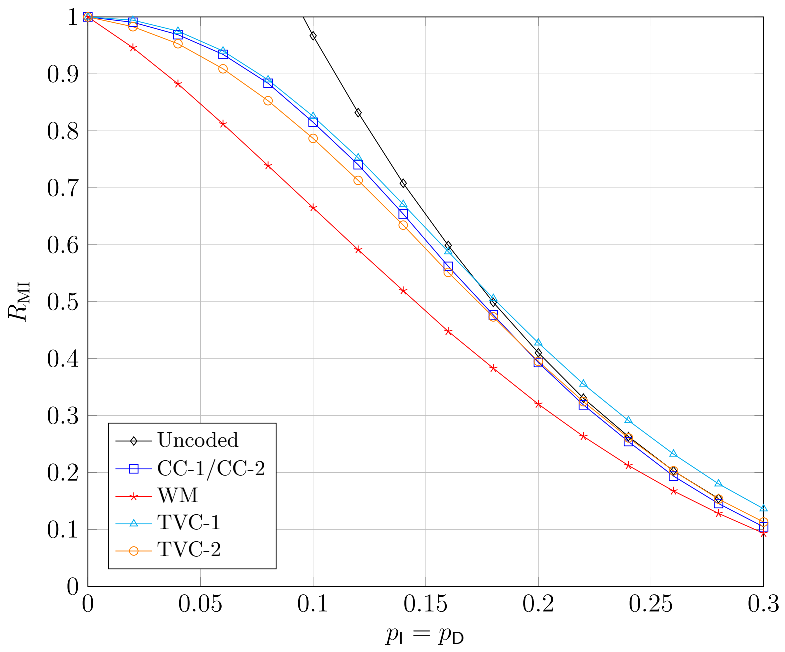


Figure I.8: Mutual information rates R_{MI} versus $p_{\mathsf{I}} = p_{\mathsf{D}}$ with $p_{\mathsf{S}} = 0$ for single sequence transmission for several inner coding schemes.

However, since we consider an input sequence of independent and uniformly distributed symbols, $H(\boldsymbol{w})$ is equal to $(N_{\mathsf{o}} + m) \log q_{\mathsf{o}}$, and hence we do not need to run the recursion.

Similarly, this approach can be used to compute the analog mutual information point of (II.7) for the case of multiple received sequences. Given an input sequence $\boldsymbol{w}$ transmitted over M identical and independent IDS channels, we will receive the M output sequences $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_M$. Then, the analog mutual information point of (II.7), denoted by $I(w; y_1, \ldots, y_M)$, which is an AIR, can be estimated by

$$\begin{aligned}\hat{I}(w; y_1, \ldots, y_M) = &- \frac{1}{N_{\mathsf{o}} + m} \log p(\boldsymbol{w}) - \frac{1}{N_{\mathsf{o}} + m} \log p(\boldsymbol{y}_1, \ldots, \boldsymbol{y}_M) \\ &+ \frac{1}{N_{\mathsf{o}} + m} \log p(\boldsymbol{w}, \boldsymbol{y}_1, \ldots, \boldsymbol{y}_M)\end{aligned}$$

when N_{o} is very large. By considering the joint sequences $(\boldsymbol{y}_1)_1^{in+d_1}, \ldots, (\boldsymbol{y}_M)_1^{in+d_M}$, $\log p(\boldsymbol{w})$, $\log p(\boldsymbol{y}_1, \ldots, \boldsymbol{y}_M)$, and $\log p(\boldsymbol{w}, \boldsymbol{y}_1, \ldots, \boldsymbol{y}_M)$ can be computed in an analog way as for the single sequence case.

Fig. I.8 shows the mutual information rate $R_{\text{MI}} \approx \hat{I}(w; y)$ for the different inner codes that we consider in this paper (see Table I.2). These plots have been simulated using 10^6 channel uses. It is evident from the curves that the TVC-1 scheme performs the best for high insertion and deletion probabilities. The TVC-1 scheme even outperforms the uncoded scheme, which might seem surprising at first glance; however, this can happen when memory is introduced to the

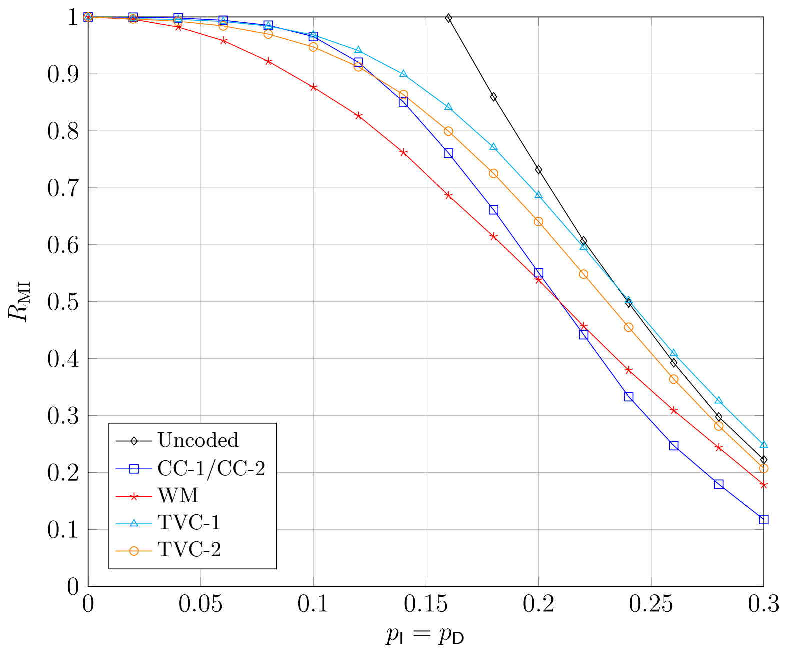


Figure I.9: Mutual information rates R_{MI} versus $p_{\mathsf{I}} = p_{\mathsf{D}}$ with $p_{\mathsf{S}} = 0$ for multiple sequence transmission with $M = 2$ for several inner coding schemes.

source/channel, which is the case for TVC-1. The mutual information rate gives us insights on the expected performance of the tabulated inner codes concatenated with an outer code tailored toward them, where extrinsic information has been passed between the inner and outer decoders. In other words, the achievable rates R_{MI} give insights on the inner-outer iterative decoding performance of our coding scheme. This is the reason why we see a difference between mutual information and BCJR-once rates ($R_{\mathrm{MI}} \geq R_{\text{BCJR-once}}$), where we also observe a change in which inner code achieves the better rate. This is expected since the effects of synchronization loss can be mitigated by performing inner-outer decoding. We observe and confirm this disposition in our results section. Finally, the achievable rate $R_{\mathrm{MI}} \approx \hat{I}(w; y_1, \ldots, y_M)$ for the case of multiple sequence transmission with $M = 2$ is shown in Fig. I.9.

It should be noted that the rate R_{MI} for the uncoded case in Figs. I.8 and I.9 is computed using the lattice implementation introduced in Section 4. Since in our decoding algorithm we have to limit the maximum and minimum drift values, we end up with inaccurate computations of $\log p(\boldsymbol{y})$ and $\log p(\boldsymbol{w}, \boldsymbol{y})$. By using a lattice implementation, we can exactly compute these quantities (details omitted for brevity). The discrepancy between the two methods is most visible for the uncoded case, whereas it is negligible for the other inner codes.

6 Simulation Results

In this section, we provide frame error rate (FER) performance results for several set-ups of our coding schemes. First, we show a comparison of the different inner coding schemes introduced in the earlier sections. In particular, we show that our results are in accordance with the BCJR-once and mutual information rates. Second, we simulate the FER performance of our designed nonbinary polar and LDPC codes with the TVC-2 inner coding scheme. Third, we simulate our coding schemes under short sequence transmission conditions. Next, we combine the best inner and outer code selection and simulate for the case of multiple received sequences. In addition, we compare the performance of our coding schemes in the multiple sequence transmission scenario with an existing MSA algorithm. Finally, to show robustness to substitution errors, we present FER results for two different values of $p_{\mathsf{S}} > 0$ using the CC-2 inner coding scheme concatenated with a designed nonbinary LDPC code.

Our simulations are done over an alphabet of size $q = 4$, which equates to the four bases $\{\mathsf{A}, \mathsf{C}, \mathsf{G}, \mathsf{T}\}$ of the DNA and we set $p_{\mathsf{S}} = 0$ (except for Fig. I.15) and $p_{\mathsf{I}} = p_{\mathsf{D}}$. An outer LDPC code is decoded via belief propagation with a maximum number of 100 iterations. When applicable, the maximum number of iterations between the inner and outer decoders is set to 100. The polar code is decoded via CRC-aided SCL decoding. Regarding the inner codes, we set $I_{\max} = 2$ and the limit of the drift random variable in decoding is set dynamically as follows. We set the drift limit to five times the standard deviation of the final drift at position N, i.e., $d_{\max} = -d_{\min} = 5\sqrt{N\frac{p_{\mathsf{D}}}{1-p_{\mathsf{D}}}}$. However, if the received sequence has a drift outside this limit, we increase the limit to be ten times the standard deviation. This can be motivated by the fact that in DNA storage, the length of each strand is known, so the drift limit in decoding can be set accordingly. Moreover, for multiple transmissions, we use $M = 2$, 3, 5, and 10. For short sequence transmission, our overall code length is $N = 128$ DNA symbols, while we use $N = 960$ DNA symbols otherwise. We picked the short and long sequence lengths to be within the range of the corresponding DNA sequencing technologies. Illumina sequencing can produce sequence lengths ranging between $100 - 300$ DNA symbols, while Oxford nanopore sequencing produces sequences of lengths $1000 - 2000$ DNA symbols. All FER results are with an overall code rate of $R = 1/2$ (in bits per DNA symbol), with an outer code rate of $R_{\mathsf{o}} = 1/2$ and an inner code rate of $R_{\mathsf{i}} = 1$.

6.1 Inner Code Optimization/Comparison

Comparing the different inner codes in our coding schemes can be done in several ways. We have already presented one way of comparison by providing the BCJR-once and mutual information rates. However, these rates correspond to the asymptotics of these codes. To present a more clear comparison, we plot in Fig. I.10 the FER performance of the different inner coding schemes concatenated with a $[240, 120]_{2^4}$ WiMax-like nonbinary outer LDPC code. Confirming the BCJR-once rate results in Fig. I.5, for a coding rate of $1/2$, we observe that the TVC-2 inner coding scheme performs the best with no iterations between the inner and outer decoders. Furthermore, when considering iterative inner-outer decoding, the

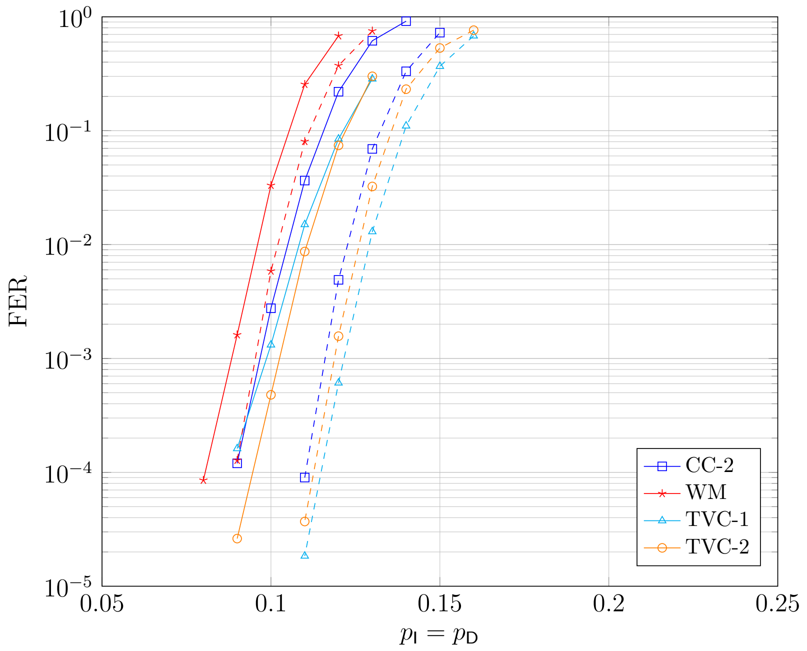


Figure I.10: FER performance vs. $p_\mathsf{I} = p_\mathsf{D}$ on different inner codes concatenated with a $[240, 120]_{2^4}$ WiMax-like outer LDPC code with overall block length of $N = 960$ DNA symbols and with $p_\mathsf{S} = 0$. Dashed lines are with inner-outer iterations while solid lines are without.

Table I.3: Iterative Decoding Thresholds of LDPC Codes With $p_\mathsf{S} = 0$

Code	p_th	Rate
Designed	0.142	1/2
WiMax-like	0.139	1/2

TVC-1 scheme performs the best. This in turn confirms the mutual information rate results in Fig. I.8. It is evident that our designed inner code improves the performance of the overall concatenated coding scheme.

6.2 Outer Code Optimization

In the following, we show FER performance results for the TVC-2 inner coding scheme concatenated with our designed outer codes. The optimization of the outer code was presented in Sections 4 and 4, where we describe how to design, respectively, an outer LDPC or polar code tailored to an inner coding scheme combined with the IDS channel. The results are shown in Fig. I.11. Furthermore, Table I.3 shows the iterative decoding threshold p_th of the WiMax-like LDPC code and our designed protograph LDPC code. The base matrix corresponding to our

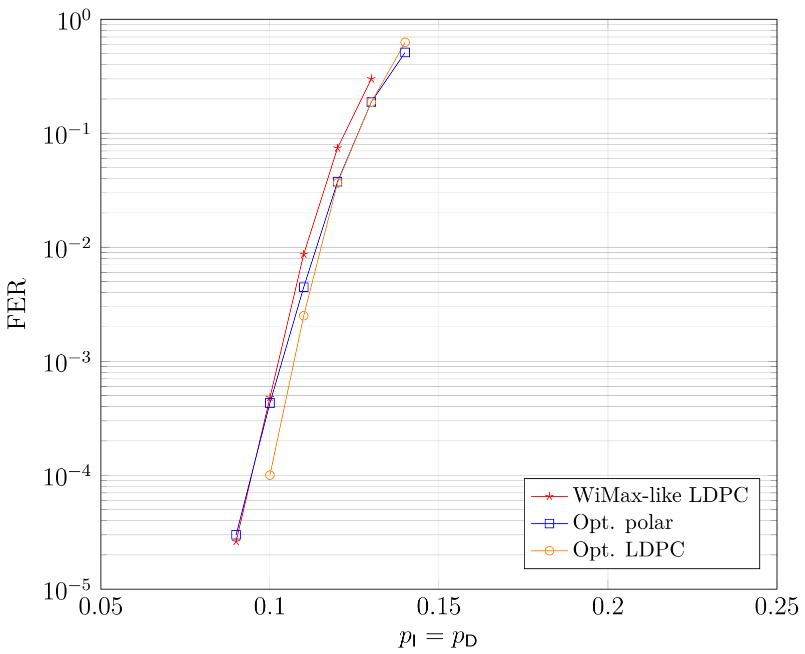


Figure I.11: FER performance vs. $p_\mathsf{I} = p_\mathsf{D}$ on different outer codes concatenated with the TVC-2 inner coding scheme, with $p_\mathsf{S} = 0$ and with no iterations between the inner and outer decoders. The polar code has parameters $N_\mathsf{o} = 256$, $R_\mathsf{o} = {}^1\!/_2$, $q_\mathsf{o} = 16$, ${}^\alpha\!/_\beta = 6$, list size 32, and $\ell_\text{CRC} = 8$.

designed protograph LDPC code is

$$\boldsymbol{B} = \begin{pmatrix} 1 & 2 & 1 & 1 \\ 1 & 1 & 2 & 1 \end{pmatrix}. \tag{I.4}$$

It should be noted that we simulate the ensemble average of the lifted protograph (see Section 4) by assigning new random weights for the edges of the Tanner graph for every new block transmission. Also, the LDPC code has been optimized for the case of no iterations between the inner and outer decoders and for the TVC-2 inner coding scheme. Evidently, we succeed in improving the performance of our coding schemes with both of the designed outer codes; the protograph LDPC code performing the best. The designed protograph LDPC code is a $[240, 120]_{2^4}$ code with girth 10, which gives a block length of $N = 960$ DNA symbols, while the designed polar code is a $[256, 128]_{2^4}$ code, resulting in $N = 1024$ DNA symbols. Again, as for the LDPC code, the polar code has been optimized for the TVC-2 inner coding scheme and with no iterations between the inner and outer decoders. These results validate our optimization and design techniques as we have managed to improve the performance compared to the standard case. For a better visual representation of how well the optimization performs, we plotted in Fig. I.6 (the red solid circle) the iterative decoding threshold for our concatenated coding scheme (TVC-2 inner coding scheme and optimized outer LDPC code constructed from the base matrix $\boldsymbol{B}$ in (I.4)) for $M = 1$. The gap to the AIR curve is very small, which

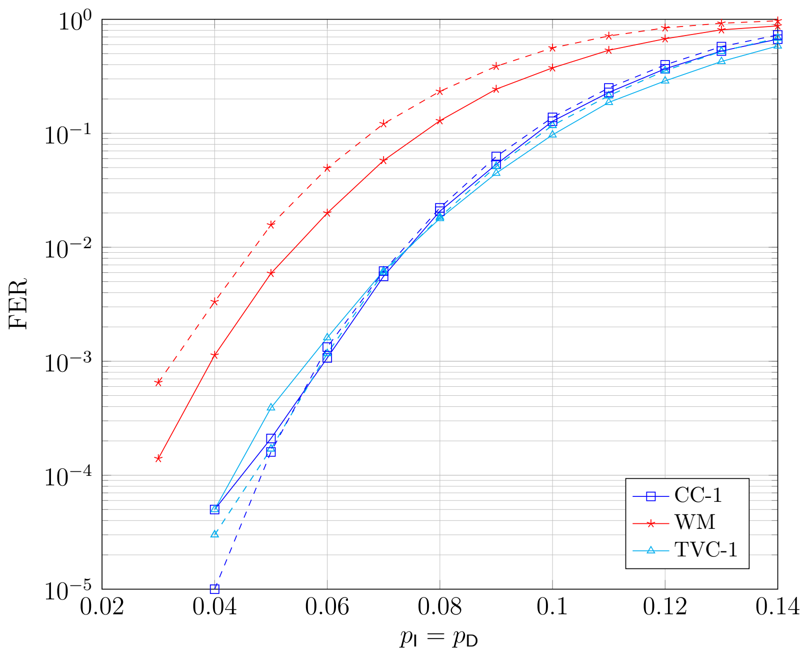


Figure I.12: FER performance vs. $p_\mathsf{I} = p_\mathsf{D}$ on short sequences of $N = 128$ DNA symbols for different inner coding schemes, with $p_\mathsf{S} = 0$ and with no iterations between the inner and outer decoders. Solid lines are for an outer polar code with parameters $N_\mathsf{o} = 64$, $R_\mathsf{o} = {}^1\!/\!_2, q_\mathsf{o} = 4$, ${}^\alpha\!/\!_\beta = 3$, list size 32, and $\ell_{\mathrm{CRC}} = 8$, while dashed lines are for an outer $[64, 32]_{2^2}$ optimized LDPC code.

validates our optimization. On a side note, optimizing the inner code is of greater importance as the inner code is responsible for maintaining synchronization, which is the main obstacle faced when dealing with IDS channels.

6.3 Short Sequence Transmission

The FER performance for the short block length regime is presented in Fig. I.12. The comparison is done on different inner codes using a $[64, 32]_{2^2}$ outer polar code or a $[64, 32]_{2^2}$ LDPC code that both have been optimized separately for each inner coding scheme. In fact, the protograph in (I.4) is optimal for all four WM, CC-1, TVC-1, and TVC-2 inner coding schemes, which could be attributed to the rather small search space. The corresponding designed nonbinary LDPC code has girth 8. Similar to the long sequence case, the watermark code is performing worse than other inner code choices, which has been as well predicted by the AIR results for our chosen rate. Surprisingly, other combinations of inner and outer codes perform very similar in the short sequence case. Due to clarity of presentation, we have excluded in the plot the results for the TVC-2 inner coding scheme, which performed the best for long sequences. However, the performance of this coding scheme is slightly worse than the CC-1 and TVC-1 results. Moreover, the TVC-1 inner coding scheme performs better for higher insertion and deletion probabilities than all inner codes with a random offset sequence. The difference of the results compared to the AIR and long sequence results may stem from the fact that for short sequences the synchronization process is done over a shorter trellis. Therefore, the knowledge of the fixed start and end point has more influence on the APPs in the middle of the sequence, which makes synchronization without a random offset sequence also more feasible for higher insertion and deletion probabilities. However, the CC-1 inner coding scheme seems to perform best for decreasing failure probabilities for short sequences. Considering the outer code choice, for the TVC-1 and CC-1 inner coding schemes, the LDPC code and the polar code yield similar performance, with the polar code slightly outperforming the optimized nonbinary LDPC code for high deletion and insertion probabilities and vice versa for low error probabilities. However, the polar code outperforms the LDPC code by a more significant gap for the WM inner coding scheme. This may be explained by the limited search space for the LDPC protograph; a higher dimension protograph might allow to close the gap.

6.4 Multiple Sequence Transmission

Next, we present FER results for the case of multiple sequence transmission with both long and short sequences. For transmission block length $N = 960$ DNA symbols, we have shown that our designed protograph LDPC code combined with the TVC-2 inner coding scheme performs the best when we do not iterate between the inner and outer decoders. Fig. I.13 shows the FER results for this scheme with multiple sequences, where we observe a significant gain in performance when increasing the number of sequences. There is also a notable gain of joint decoding compared to separate decoding as illustrated for $M = 2$ sequences. Moreover, separate decoding with $M = 3$ performs similar to optimal joint decoding for $M = 2$, but with much lower complexity. This confirms the earlier observation with the BCJR-once rates (see Fig. I.6), and shows the viability of our proposed sub-optimal decoding algorithm. For short sequence transmission we perform our analysis with an outer polar code using the CC-1 inner coding scheme. In the following we include a comparison of that coding scheme with an existing MSA method from the literature for the multiple sequence scenario.[5]

MSA tools are often used in the biological sector for reconstruction of an original sequence given multiple corrupted sequences. Here, we consider the *T-Coffee* MSA method presented in [31] using the open-source library *SeqAn* [32]. Given the M sequences $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_M$ from the channel output, the MSA algorithm computes a consensus sequence $\widetilde{\boldsymbol{y}} = (\widetilde{y}_1, \ldots, \widetilde{y}_{\widetilde{N}})$, where $\widetilde{y}_i \in \Sigma_q$ and $\widetilde{N}$ is not necessarily equal to N. We use a simple Levenshtein distance orientated scoring scheme to compute the alignment, where a match, a mismatch, and a gap have scores 0, -2, and -1, respectively. Given the computed alignment, we apply a majority decision to output the consensus sequence $\widetilde{\boldsymbol{y}}$. Subsequently, the sequence $\widetilde{\boldsymbol{y}}$ is given to the inner decoder to compute the symbol APPs $p(w_i|\widetilde{\boldsymbol{y}})$ for each outer codeword

[5]Note that we have *not* optimized neither the LDPC code nor the polar code for the multiple sequence scenario, but rather use the optimized codes for the case of a single sequence transmission.

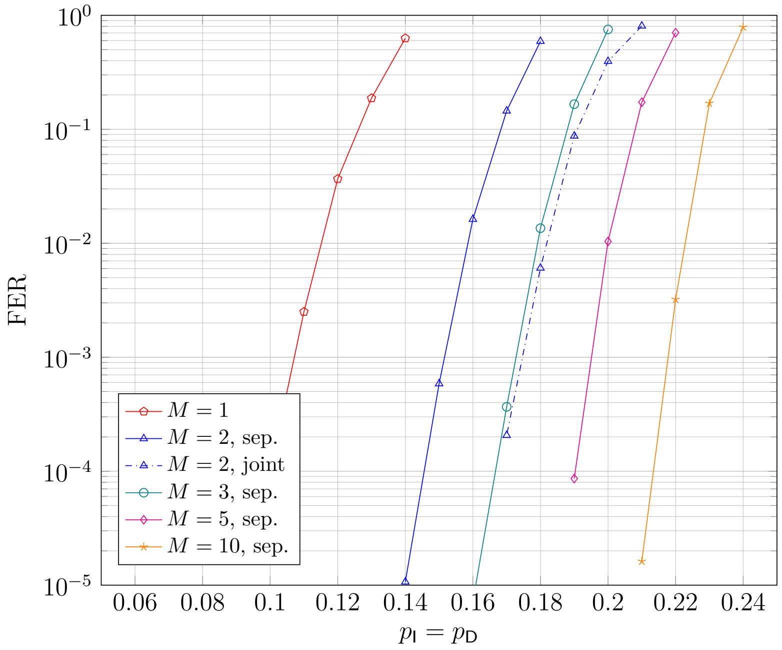


Figure I.13: FER performance vs. $p_\mathsf{I} = p_\mathsf{D}$ for the TVC-2 inner coding scheme concatenated with our optimized $[240, 120]_{2^4}$ outer LDPC code with block length $N = 960$ DNA symbols for multiple sequences, with $p_\mathsf{S} = 0$ and with no iterations between the inner and outer decoders. Solid lines represent separate decoding while dash dotted are for joint decoding.

symbol as described in Section 4. We can directly determine the total number of computed operations by the sum of that of decoding a single received sequence using the inner code and that of the MSA algorithm itself. Therefore, combining the complexity of the T-Coffee algorithm according to [31] and that of decoding a single received sequence using the inner code, the complexity of the MSA method is

$$C_{\text{MSA}} = C_{\text{single}} + \mathcal{O}(M^2N^2) + \mathcal{O}(M^3N).$$

Fig. I.14 shows the FER of a $[64, 32]_{2^2}$ outer polar code concatenated with the CC-1 inner coding scheme for a short block length of $N = 128$ DNA symbols and multiple received sequences. Specifically, we compare the separate and joint decoding approaches discussed in Section 3 and the aforementioned MSA method. First observe that due to complexity reasons joint decoding for $M > 2$ becomes infeasible. Moreover, the MSA method is impractical for the case of $M = 2$ due to the applied majority decision. For the case of $M = 2$, we see a significant gain of joint decoding compared to separate decoding. This is no surprise since joint decoding exploits the full knowledge of the two received sequences concurrently, however, at the expense of a large trellis increasing exponentially in M. On the other hand, the separate decoding approach ignores during the inner decoding the

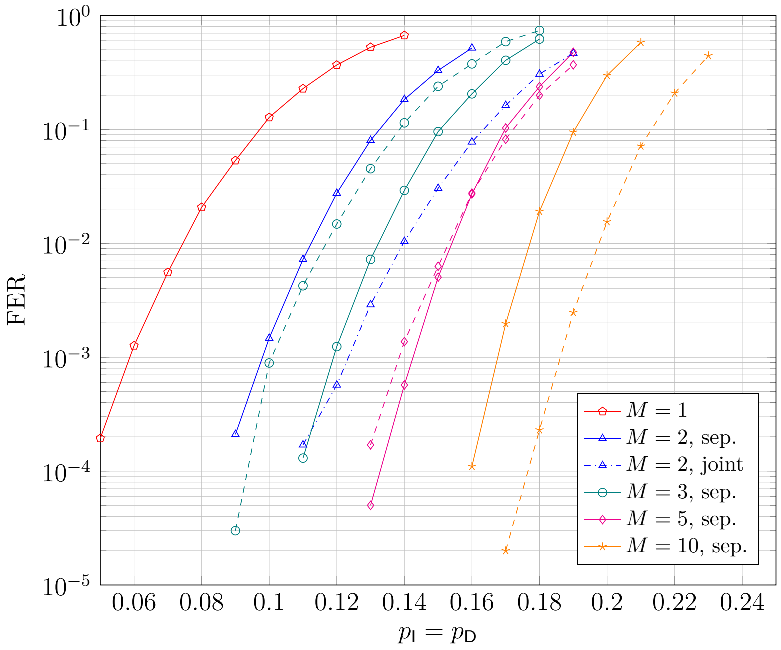


Figure I.14: FER performance vs. $p_\mathsf{I} = p_\mathsf{D}$ on short sequences of $N = 128$ DNA symbols, with $p_\mathsf{S} = 0$ and with no iterations between the inner and outer decoders. The outer code is a polar code with parameters $N_\mathsf{o} = 64$, $R_\mathsf{o} = 1/2$, $q_\mathsf{o} = 4$, $\alpha/\beta = 3$, list size 32, and $\ell_{\text{CRC}} = 8$, while we use the CC-1 inner coding scheme. Comparison on different multiple sequence decoding approaches. Solid lines represent separate decoding, dashed are for MSA decoding, and dash dotted are for joint decoding.

fact that the received sequences stem from the same original transmitted sequence. Comparing the MSA method and separate decoding for the case of $M > 2$, we see that the performance of separate decoding is better than that of the MSA method for $M = 3$, similar for $M = 5$, and worse for $M = 10$. This can be explained similarly as before, since the MSA method exploits coherences between the received sequences, albeit without any structural assumption, e.g., knowledge of the codebooks. Nevertheless, the gain of the MSA method comes at the price of a higher complexity, where the dominating factor is $\mathcal{O}(M^2N^2) + \mathcal{O}(M^3N)$ compared to separate decoding with $\mathcal{O}(MN^{3/2})$ complexity, as the total number of drift states Δ is of order $\mathcal{O}(\sqrt{N})$.

For a better grasp on the difference in complexity between joint and separate decoding, we now provide some numerical examples of the total number of candidates for the HMM state variable σ_i, denoted by σ_{total}, for the case of an inner convolutional code. For a sequence of length $N = 960$ and $p_\mathsf{I} = p_\mathsf{D} = 0.15$, $\sigma_{\text{total}} = 2^\nu \times 131$ for separate decoding, while $\sigma_{\text{total}} = 2^\nu \times 131^M$ for joint decoding. For a sequence of length $N = 128$, $\sigma_{\text{total}} = 2^\nu \times 48$ for separate decoding, while $\sigma_{\text{total}} = 2^\nu \times 48^M$ for joint decoding.

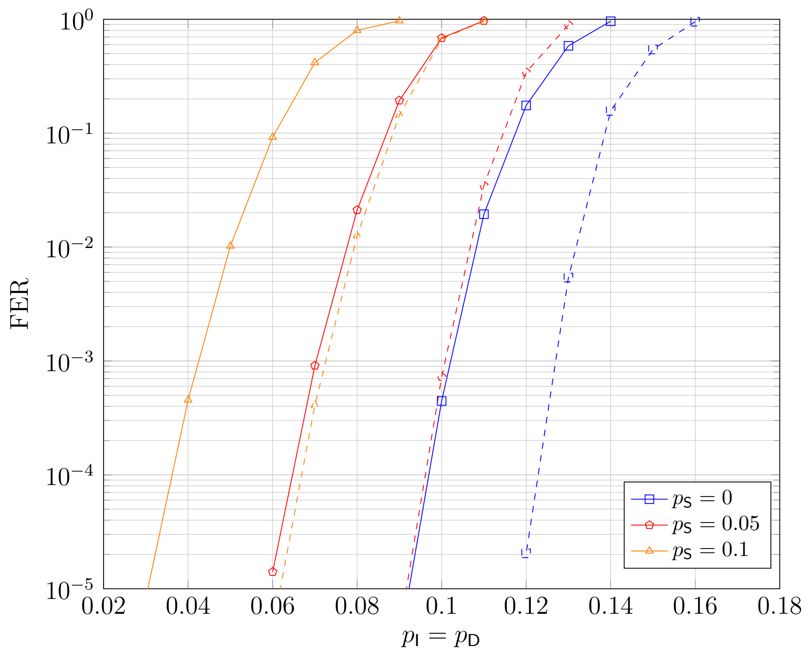


Figure I.15: FER performance vs. $p_{\mathsf{I}} = p_{\mathsf{D}}$ for the CC-2 inner coding scheme concatenated with an optimized $[240, 120]_{2^4}$ outer LDPC code with block length $N = 960$ DNA symbols for several substitution error probabilities and with no iterations between the inner and outer decoders. Solid lines correspond to $M = 1$ and dashed lines to $M = 2$ with separate decoding.

6.5 Frame Error Rate Results With Substitution Errors

To show the robustness of our coding schemes and decoding algorithms to substitution errors, we show in Fig. I.15 FER results with $p_{\mathsf{S}} > 0$. We use the CC-2 inner coding scheme concatenated with an optimized (for single sequence transmission and with no iterations between the inner and outer decoders) nonbinary protograph LDPC code using the techniques mentioned earlier. The optimal base matrix for the case of $p_{\mathsf{S}} = 0$ is $\boldsymbol{B} = \left(\begin{smallmatrix} 1 & 2 & 1 & 1 \\ 1 & 1 & 2 & 1 \end{smallmatrix}\right)$ as mentioned earlier, while $\boldsymbol{B} = \left(\begin{smallmatrix} 1 & 2 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{smallmatrix}\right)$ gives the best performance for $p_{\mathsf{S}} = 0.05$ and 0.1. The FER curves in Fig. I.15 are in agreement with the BCJR-once rates in Fig. I.7; the loss in performance with increasing p_{S} is approximately the same for $M = 1$ and $M = 2$ with separate decoding.

7 Conclusion

In this paper, we proposed concatenated coding schemes for the problem of transmitting one DNA sequence over multiple parallel IDS channels. First, we proposed two novel approaches, or decoding algorithms, for multiple sequence transmission. The first algorithm, being a benchmark for the second one, is an optimal symbolwise MAP decoder but it suffers from high complexity, while the second algorithm is a sub-optimal but more practical decoder with much reduced complexity. We showed with these algorithms that we can achieve significant gains as compared to the single sequence transmission case. Second, we proposed optimization techniques for both the inner and outer codes tailored to the IDS channel. We designed an inner TVC and outer nonbinary LDPC and polar codes that improve the overall performance of the scheme. In addition, we studied the asymptotic performance of different inner codes through AIRs and showed that the FER results are in accordance with them. Lastly, we would like to point out that the code rate for our concatenated coding scheme was chosen for convenience, while for a real-life scheme it should be selected based on the target IDS channel error rates which depend on the sequencing technology. Although it may be considered to be a low rate, the scheme can be straightforwardly adapted to higher rates. In this work, we showed that for an IDS channel with a very high error rate for insertions and deletions, our coding scheme with rate $1/2$ performs very well.

A Symbolwise APPs for Memoryless Channels

We show that for independent channel input symbols, i.e., $p(\boldsymbol{x}) = \prod_{i=1}^{N} p(x_i)$ and a memoryless channel that produces M output sequences it holds that

$$p(x_i|\boldsymbol{y}) \propto p(x_i)^{1-M} \prod_{j=1}^{M} p(x_i|\boldsymbol{y}_j),$$

where the proportionality is with respect to a constant that does not depend on x_i. We start with expanding the APP to

$$\begin{aligned} p(x_i|\boldsymbol{y}) &\overset{(a)}{=} \sum_{\boldsymbol{x}:x_i} p(\boldsymbol{x}|\boldsymbol{y}) = \sum_{\boldsymbol{x}:x_i} \frac{p(\boldsymbol{y}|\boldsymbol{x})p(\boldsymbol{x})}{p(\boldsymbol{y})} = \sum_{\boldsymbol{x}:x_i} \frac{p(\boldsymbol{x})}{p(\boldsymbol{y})} \prod_{j=1}^{M} p(\boldsymbol{y}_j|\boldsymbol{x}) \\ &\overset{(b)}{=} \sum_{\boldsymbol{x}:x_i} \frac{1}{p(\boldsymbol{y})} \prod_{k=1}^{N} p(x_k) \prod_{j=1}^{M} p(y_{j,k}|x_k) \\ &\overset{(c)}{=} p(x_i) \prod_{j=1}^{M} p(y_{j,i}|x_i) \sum_{\boldsymbol{x}:x_i} \frac{1}{p(\boldsymbol{y})} \prod_{k\neq i} p(x_k) \prod_{l=1}^{M} p(y_{l,k}|x_k) \\ &\propto p(x_i) \prod_{j=1}^{M} p(y_{j,i}|x_i), \end{aligned}$$

where in (a) we used the notation of the sum over $\boldsymbol{x} : x_i$, which ranges over all vectors $\boldsymbol{x}$ whose i-th symbol is equal to x_i. Equality (b) is due to the fact that the channel is memoryless. Finally, in (c) we factored out the terms corresponding to x_i, which is possible as x_i is constant within the sum. We can do the analogous steps for $p(x_i|\boldsymbol{y}_j)$ to deduce that

$$p(x_i|\boldsymbol{y}_j) \propto p(x_i)p(y_{j,i}|x_i).$$

Combining these two equivalences, we arrive at the desired proportionality. Notice that this relation translates to the APPs $p(w_i|\boldsymbol{y})$ when no inner code is used, i.e., when $\boldsymbol{w} = \boldsymbol{x}$.

References

[1] G. M. Church, Y. Gao, and S. Kosuri, "Next-generation digital information storage in DNA," *Science*, vol. 337, p. 1628, Aug. 2012.

[2] N. Goldman, P. Bertone, S. Chen, C. Dessimoz, E. M. LeProust, B. Sipos, and E. Birney, "Towards practical, high-capacity, low-maintenance information storage in synthesized DNA," *Nature*, vol. 494, pp. 77–80, Feb. 2013.

[3] R. N. Grass, R. Heckel, M. Puddu, D. Paunescu, and W. J. Stark, "Robust chemical preservation of digital information on DNA in silica with error-correcting codes," *Angew. Chem. Int. Ed.*, vol. 54, pp. 2552–2555, Feb. 2015.

[4] M. Blawat, K. Gaedke, I. Hütter, X.-M. Chen, B. Turczyk, S. Inverso, B. W. Pruitt, and G. M. Church, "Forward error correction for DNA data storage," in *Proc. Int. Conf. Comput. Sci.*, (San Diego, CA, USA), pp. 1011–1022, June 2016.

[5] J. Bornholt, R. Lopez, D. M. Carmean, L. Ceze, G. Seelig, and K. Strauss, "A DNA-based archival storage system," in *Proc. Int. Conf. Architect. Supp. Program. Lang. Operat. Syst.*, (Atlanta, Georgia, USA), pp. 637–649, Mar. 2016.

[6] Y. Erlich and D. Zielinski, "DNA fountain enables a robust and efficient storage architecture," *Science*, vol. 355, pp. 950–954, Mar. 2017.

[7] S. M. H. T. Yazdi, R. Gabrys, and O. Milenkovic, "Portable and error-free DNA-based data storage," *Sci. Rep.*, vol. 7, pp. 1–6, July 2017.

[8] L. Organick, S. D. Ang, Y.-J. Chen, R. Lopez, S. Yekhanin, K. Makarychev, M. Z. Racz, G. Kamath, P. Gopalan, B. Nguyen, C. N. Takahashi, S. Newman, H.-Y. Parker, C. Rashtchian, K. Stewart, G. Gupta, R. Carlson, J. Mulligan, D. Carmean, G. Seelig, L. Ceze, and K. Strauss, "Random access in large-scale DNA data storage," *Nature Biotechnol.*, vol. 36, pp. 242–248, Mar. 2018.

[9] Y. Wang, M. Noor-A-Rahim, J. Zhang, E. Gunawan, Y. L. Guan, and C. L. Poh, "High capacity DNA data storage with variable-length oligonucleotides using repeat accumulate code and hybrid mapping," *J. Biol. Eng.*, vol. 13, pp. 1–11, Nov. 2019.

[10] S. Chandak, J. Neu, K. Tatwawadi, J. Mardia, B. Lau, M. Kubit, R. Hulett, P. Griffin, M. Wootters, T. Weissman, and H. Ji, "Overcoming high nanopore basecaller error rates for DNA storage via basecaller-decoder integration and convolutional codes," in *Proc. IEEE Int. Conf. Acoust., Speech, Sig. Process.*, (Barcelona, Spain), pp. 8822–8826, May 2020.

[11] S. M. H. T. Yazdi, Y. Yuan, J. Ma, H. Zhao, and O. Milenkovic, "A rewritable, random-access DNA-based storage system," *Sci. Rep.*, vol. 5, pp. 1–10, Sept. 2015.

[12] C. Pan, S. K. Tabatabaei, S. M. H. T. Yazdi, A. G. Hernandez, C. M. Schroeder, and O. Milenkovic, “Rewritable two-dimensional DNA-based data storage with machine learning reconstruction,” *Nature Commun.*, vol. 13, pp. 1–12, May 2022.

[13] S. K. Tabatabaei, B. Pham, C. Pan, J. Liu, S. Chandak, S. A. Shorkey, A. G. Hernandez, A. Aksimentiev, M. Chen, C. M. Schroeder, and O. Milenkovic, “Expanding the molecular alphabet of DNA-based data storage systems with neural network nanopore readout processing,” *Nano Lett.*, vol. 22, pp. 1905–1914, Mar. 2022.

[14] P. L. Antkowiak, J. Lietard, M. Z. Darestani, M. M. Somoza, W. J. Stark, R. Heckel, and R. N. Grass, “Low cost DNA data storage using photolithographic synthesis and advanced information reconstruction and error correction,” *Nature Commun.*, vol. 11, Oct. 2020.

[15] R. Heckel, G. Mikutis, and R. N. Grass, “A characterization of the DNA data storage channel,” *Sci. Rep.*, vol. 9, pp. 1–12, July 2019.

[16] R. G. Gallager, “Sequential decoding for binary channel with noise and synchronization errors,” tech. rep., Arlington, VA, USA, Sept. 1961.

[17] K. S. Zigangirov, “Sequential decoding for a binary channel with drop-outs and insertions,” *Probl. Peredachi Inf.*, vol. 5, no. 2, pp. 23–30, 1969.

[18] L. Calabi and W. E. Hartnett, “Some general results of coding theory with applications to the study of codes for the correction of synchronization errors,” *Inf. Control*, vol. 15, pp. 235–249, Sept. 1969.

[19] L. R. Bahl and F. Jelinek, “Decoding for channels with insertions, deletions, and substitutions with applications to speech recognition,” *IEEE Trans. Inf. Theory*, vol. 21, pp. 404–411, July 1975.

[20] V. I. Levenshtein, “Binary codes capable of correcting deletions, insertions and reversals,” *Soviet Physics Doklady*, vol. 10, pp. 707–710, Feb. 1966.

[21] M. C. Davey and D. J. C. MacKay, “Reliable communication over channels with insertions, deletions, and substitutions,” *IEEE Trans. Inf. Theory*, vol. 47, pp. 687–698, Feb. 2001.

[22] M. F. Mansour and A. H. Tewfik, “Convolutional decoding in the presence of synchronization errors,” *IEEE J. Sel. Areas Commun.*, vol. 28, pp. 218–227, Feb. 2010.

[23] V. Buttigieg and N. Farrugia, “Improved bit error rate performance of convolutional codes with synchronization errors,” in *Proc. IEEE Int. Conf. Commun.*, (London, U.K.), pp. 4077–4082, June 2015.

[24] J. A. Briffa, H. G. Schaathun, and S. Wesemeyer, “An improved decoding algorithm for the Davey-MacKay construction,” in *Proc. IEEE Int. Conf. Commun.*, (Cape Town, South Africa), May 2010.

[25] M. Inoue and H. Kaneko, “Adaptive synchronization marker for insertion/deletion/substitution error correction,” in *Proc. IEEE Int. Symp. Inf. Theory*, (Cambridge, MA, USA), pp. 508–512, July 2012.

[26] R. Shibata, G. Hosoya, and H. Yashima, “Design of irregular LDPC codes without markers for insertion/deletion channels,” in *Proc. IEEE Glob. Commun. Conf.*, (Waikoloa, HI, USA), Dec. 2019.

[27] H. Koremura and H. Kaneko, “Insertion/deletion/substitution error correction by a modified successive cancellation decoding of polar code,” *IEICE Trans. Fundam. Electron., Commun. Comput. Sci.*, vol. E103.A, pp. 695–703, Apr. 2020.

[28] H. D. Pfister and I. Tal, “Polar codes for channels with insertions, deletions, and substitutions,” in *Proc. IEEE Int. Symp. Inf. Theory*, (Melbourne, Australia), pp. 2554–2559, July 2021.

[29] R. C. Edgar, “MUSCLE: multiple sequence alignment with high accuracy and high throughput,” *Nucleic Acids Res.*, vol. 32, pp. 1792–1797, Mar. 2004.

[30] J. Kim and J. Ma, “PSAR-Align: Improving multiple sequence alignment using probabilistic sampling,” *Bioinformatics*, vol. 30, pp. 1010–1012, Apr. 2014.

[31] C. Notredame, D. G. Higgins, and J. Heringa, “T-Coffee: A novel method for fast and accurate multiple sequence alignment,” *J. Mol. Biol.*, vol. 302, pp. 205–217, Sept. 2000.

[32] K. Reinert, T. H. Dadi, M. Ehrhardt, H. Hauswedell, S. Mehringer, R. Rahn, J. Kim, C. Pockrandt, J. Winkler, E. Siragusa, G. Urgese, and D. Weese, “The SeqAn C++ template library for efficient sequence analysis: A resource for programmers,” *J. Biotechnol.*, vol. 261, pp. 157–168, Nov. 2017.

[33] H. Mercier and V. K. Bhargava, “Convolutional codes for channels with deletion errors,” in *Proc. Canadian Workshop Inf. Theory*, (Ottawa, ON, Canada), pp. 136–139, May 2009.

[34] V. I. Levenshtein, “Efficient reconstruction of sequences from their subsequences or supersequences,” *J. Combin. Theory, Series A*, vol. 93, pp. 310–332, Feb. 2001.

[35] M. A. Sini and E. Yaakobi, “Reconstruction of sequences in DNA storage,” in *Proc. IEEE Int. Symp. Inf. Theory*, (Paris, France), pp. 290–294, July 2019.

[36] J. Brakensiek, R. Li, and B. Spang, “Coded trace reconstruction in a constant number of traces,” in *Proc. IEEE Annu. Symp. Found. Comput. Sci.*, (Durham, NC, USA), pp. 482–493, Nov. 2020.

[37] M. Cheraghchi, R. Gabrys, O. Milenkovic, and J. Ribeiro, “Coded trace reconstruction,” *IEEE Trans. Inf. Theory*, vol. 66, pp. 6084–6103, Oct. 2020.

[38] M. Abroshan, R. Venkataramanan, L. Dolecek, and A. Guillén i Fàbregas, “Coding for deletion channels with multiple traces,” in *Proc. IEEE Int. Symp. Inf. Theory*, (Paris, France), pp. 1372–1376, July 2019.

[39] S. R. Srinivasavaradhan, M. Du, S. Diggavi, and C. Fragouli, “Symbolwise MAP for multiple deletion channels,” in *Proc. IEEE Int. Symp. Inf. Theory*, (Paris, France), pp. 181–185, July 2019.

[40] O. Sabary, A. Yucovich, G. Shapira, and E. Yaakobi, “Reconstruction algorithms for DNA-storage systems.” Sept. 2020.

[41] A. Lenz, I. Maarouf, L. Welter, A. Wachter-Zeh, E. Rosnes, and A. Graell i Amat, “Concatenated codes for recovery from multiple reads of DNA sequences,” in *Proc. IEEE Inf. Theory Workshop*, (Riva del Garda, Italy), Apr. 2020.

[42] S. R. Srinivasavaradhan, S. Gopi, H. D. Pfister, and S. Yekhanin, “Trellis BMA: Coded trace reconstruction on IDS channels for DNA storage,” in *Proc. IEEE Int. Symp. Inf. Theory*, (Melbourne, Australia), pp. 2453–2458, July 2021.

[43] R. Sakogawa and H. Kaneko, “Symbolwise MAP estimation for multiple-trace insertion/deletion/substitution channels,” in *Proc. IEEE Int. Symp. Inf. Theory*, (Los Angles, CA, USA), pp. 781–785, June 2020.

[44] R. Shibata, G. Hosoya, and H. Yashima, “Concatenated LDPC/trellis codes: Surpassing the symmetric information rate of channels with synchronization errors,” *IEICE Trans. Fundam. Electron., Commun. Comput. Sci.*, vol. E103.A, pp. 1283–1291, Nov. 2020.

[45] D. Fertonani, T. M. Duman, and M. F. Erden, “Bounds on the capacity of channels with insertions, deletions and substitutions,” *IEEE Trans. Commun.*, vol. 59, pp. 2–6, Jan. 2011.

[46] R. L. Dobrushin, “Shannon’s theorems for channels with synchronization errors,” *Probl. Peredachi Inf.*, vol. 3, no. 4, pp. 18–36, 1967.

[47] V. Buttigieg and J. A. Briffa, “Improved code construction for synchronization error correction,” in *Proc. 10th Int. ITG Conf. Syst., Commun. Coding*, (Hamburg, Germany), Feb. 2015.

[48] J. A. Briffa and H. G. Schaathun, “Improvement of the Davey-MacKay construction,” in *Proc. Int. Symp. Inf. Theory Appl.*, (Auckland, New Zealand), Dec. 2008.

[49] P.-M. Nguyen, M. A. Armand, and T. Wu, “On the watermark string in the Davey-MacKay construction,” *IEEE Commun. Lett.*, vol. 17, pp. 1830–1833, Sept. 2013.

[50] V. Buttigieg and J. A. Briffa, “Codebook and marker sequence design for synchronization-correcting codes,” in *Proc. IEEE Int. Symp. Inf. Theory*, (Saint Petersburg, Russia), pp. 1579–1583, July/Aug. 2011.

[51] R. R. Varshamov and G. M. Tenengol'ts, "Code correcting single asymmetric errors," *Avtomat. Telemekh.*, vol. 26, no. 2, pp. 288–292, 1965.

[52] A. A. El Gamal, L. A. Hemaspaandra, I. Shperling, and V. K. Wei, "Using simulated annealing to design good codes," *IEEE Trans. Inf. Theory*, vol. 33, pp. 116–123, Jan. 1987.

[53] P. R. J. Östergård, "A fast algorithm for the maximum clique problem," *Discrete Appl. Math.*, vol. 120, pp. 197–207, Aug. 2002.

[54] E. C. Sewell, "A branch and bound algorithm for the stability number of a sparse graph," *INFORMS J. Comput.*, vol. 10, pp. 438–447, Nov. 1998.

[55] R. Carraghan and P. M. Pardalos, "An exact algorithm for the maximum clique problem," *Operations Res. Lett.*, vol. 9, pp. 375–382, Nov. 1990.

[56] X.-Y. Hu, E. Eleftheriou, and D.-M. Arnold, "Progressive edge-growth Tanner graphs," in *Proc. IEEE Glob. Telecommun. Conf.*, (San Antonio, TX, USA), pp. 995–1001, Nov. 2001.

[57] A. Kavčić, X. Ma, and M. Mitzenmacher, "Binary intersymbol interference channels: Gallager codes, density evolution, and code performance bounds," *IEEE Trans. Inf. Theory*, vol. 49, pp. 1636–1652, July 2003.

[58] P. Yuan and F. Steiner, "Construction and decoding algorithms for polar codes based on 2×2 non-binary kernels," in *Proc. Int. Symp. Turbo Codes Iterative Inf. Process.*, (Hong Kong, China), Dec. 2018.

[59] E. Arıkan, "Channel polarization: A method for constructing capacity-achieving codes for symmetric binary-input memoryless channels," *IEEE Trans. Inf. Theory*, vol. 55, pp. 3051–3073, July 2009.

[60] I. Tal and A. Vardy, "List decoding of polar codes," *IEEE Trans. Inf. Theory*, vol. 61, pp. 2213–2226, May 2015.

[61] R. R. Muller and W. H. Gerstacker, "On the capacity loss due to separation of detection and decoding," *IEEE Trans. Inf. Theory*, vol. 50, pp. 1769–1778, Aug. 2004.

[62] J. B. Soriaga, H. D. Pfister, and P. H. Siegel, "Determining and approaching achievable rates of binary intersymbol interference channels using multistage decoding," *IEEE Trans. Inf. Theory*, vol. 53, pp. 1416–1429, Apr. 2007.

[63] H. D. Pfister, J. B. Soriaga, and P. H. Siegel, "On the achievable information rates of finite state ISI channels," in *Proc. IEEE Glob. Telecommun. Conf.*, (San Antonio, TX, USA), pp. 2992–2996, Nov. 2001.

[64] D. M. Arnold, H.-A. Loeliger, P. O. Vontobel, A. Kavčić, and W. Zeng, "Simulation-based computation of information rates for channels with memory," *IEEE Trans. Inf. Theory*, vol. 52, pp. 3498–3508, Aug. 2006.

[65] L. Szczecinski and A. Alvarado, *Bit-Interleaved Coded Modulation: Fundamentals, Analysis and Design.* Chichester, U.K.: John Wiley and Sons, 2015.

[66] J. Hagenauer, "The exit chart - introduction to extrinsic information transfer in iterative processing," in *Proc. Eur. Sig. Process. Conf.*, (Vienna, Austria), pp. 1541–1548, Sept. 2004.

[67] I. Land, P. A. Hoeher, and S. Gligorević, "Computation of symbol-wise mutual information in transmission systems with logAPP decoders and application to EXIT charts," in *Proc. Int. ITG Conf. Source Channel Coding*, (Erlangen, Germany), pp. 195–202, Jan. 2004.

[68] J. Kliewer, S. X. Ng, and L. Hanzo, "Efficient computation of EXIT functions for nonbinary iterative decoding," *IEEE Trans. Commun.*, vol. 54, pp. 2133–2136, Dec. 2006.

Paper II

Finite Blocklength Performance Bound for the DNA Storage Channel

Issam Maarouf, Gianluigi Liva, Eirik Rosnes, and Alexandre Graell i Amat
International Symposium on Topics in Coding (ISTC), Brest, France, Sep. 2023.

The layout has been revised.

Abstract

We present a finite blocklength performance bound for a DNA storage channel with insertions, deletions, and substitutions. The considered bound—the dependency testing (DT) bound, introduced by Polyanskiy *et al.* in 2010—, provides an upper bound on the achievable frame error probability and can be used to benchmark coding schemes in the practical short-to-medium blocklength regime. In particular, we consider a concatenated coding scheme where an inner synchronization code deals with insertions and deletions and the outer code corrects remaining (mostly substitution) errors. The bound depends on the inner synchronization code. Thus, it allows to guide its choice. We then consider low-density parity-check codes for the outer code, which we optimize based on extrinsic information transfer charts. Our optimized coding schemes achieve a normalized rate of 88% to 96% with respect to the DT bound for code lengths up to 2000 DNA symbols for a frame error probability of 10^{-3} and code rate $1/2$.

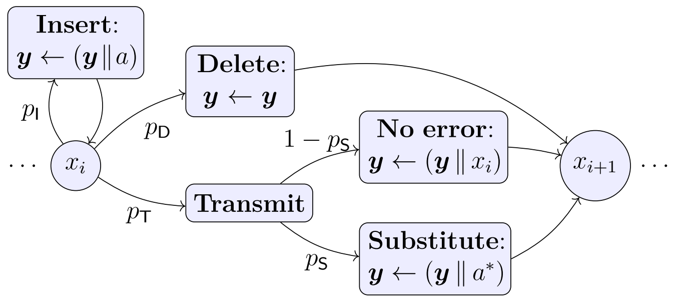
1 Introduction

Using deoxyribonucleic acid (DNA) as a medium to store data is seen as the next frontier of data storage, providing unprecedented durability and density. Several experiments have already demonstrated the viability of DNA-based data storage, see, e.g., [1, 2].

The DNA storage channel is impaired by insertions, deletions, and substitutions (IDSs) arising from the synthesis and sequencing of DNA sequences[3]. Hence, reliable storage of data in DNA requires the use of error-correcting codes. Designing a code that handles IDS errors jointly is, however, a daunting task. Davey and MacKay [4] proposed a clever solution to this problem by introducing a serially-concatenated coding scheme (for the binary IDS channel) in which the inner code, called *synchronization* code, deals with insertions and deletions, and the outer code (a low-density parity-check (LDPC) code in [4]) corrects remaining errors, mostly in the form of substitutions.

The literature on coding for DNA storage is abundant. Most works consider a very small number of deletions and/or insertions—i.e., an adversarial channel—and a single DNA strand. In DNA-based storage, however, errors occur probabilistically and can be substantial, and the synthesis and sequencing processes result in multiple (noisy) copies of the same DNA strand. The authors in [5] were the first to introduce decoding algorithms for coding schemes exploiting multiple reads of the DNA sequence. The work [5] was followed by [6].

The works [5] and [6] also provided achievable information rates, which give insight into the performance of coding schemes with very large blocklengths. However, current DNA storage technology only supports the synthesis and sequencing of short-to-medium-length DNA strands, in the range of 100-2000 DNA symbols. Therefore, performance bounds for the finite blocklength regime would be more



Insert: $\boldsymbol{y} \leftarrow (\boldsymbol{y} \,\|\, a)$
Delete: $\boldsymbol{y} \leftarrow \boldsymbol{y}$
No error: $\boldsymbol{y} \leftarrow (\boldsymbol{y} \,\|\, x_i)$
Transmit
Substitute: $\boldsymbol{y} \leftarrow (\boldsymbol{y} \,\|\, a^*)$
p_I
p_D
p_T
$1 - p_\mathsf{S}$
p_S
$\cdots$ x_i
x_{i+1} $\cdots$

Figure II.1: State-based representation of the DNA storage channel with IDS errors.

informative for the DNA channel. To the best of our knowledge, no finite blocklength performance bounds for the DNA storage channel (and IDS channels in general) exist in the literature.

In this paper, we provide a finite blocklength performance bound for a DNA storage channel with IDS errors. Particularly, we consider the dependency testing (DT) bound [7] based on the *random coding* principle, which gives an upper bound on the frame error probability achievable over the DNA storage channel. The bound is tailored to a concatenated coding scheme that uses an inner synchronization code and depends on the inner code. Hence, it can be used as a handy tool to optimize the inner synchronization code for the finite blocklength regime. Further, the bound provides a benchmark to compare coding schemes for DNA storage in the practical short-to-medium blocklength regime. We also consider the optimization of an outer LDPC code for a given inner code using extrinsic information transfer (EXIT) charts, and show that an optimized concatenated coding scheme achieves a normalized rate of 87% to 97% with respect to the DT bound for a frame error probability of 10^{-3} and code rate $1/2$, depending on the sequence length. These values are similar to those of state-of-the-art coding schemes for simpler memoryless channels (such as the Gaussian channel and the binary symmetric channel), highlighting that the scheme in [5] achieves excellent performance for the DNA storage channel in the short-to-medium blocklength regime.

2 System Model

2.1 Channel Model

We consider the widely-used simplified channel model depicted in Fig. II.1 [4, 8] for the DNA storage channel, where IDS errors are independent and identically distributed. Let $\boldsymbol{x} = (x_1, \ldots, x_N)$, $x_i \in \Sigma_q = \{0, 1, \ldots, q-1\}$,[6] be the information DNA sequence of length N to be transmitted over the channel. The sequence can be viewed as a queue of symbols, where each symbol x_i is successively transmitted over the channel. The received sequence $\boldsymbol{y} = (y_1, \ldots, y_{N'})$, where N' may be different to N due to insertions and deletions, is generated state by state and is obtained as follows. Assume x_i is next in queue to be transmitted over the channel.

[6]For the DNA storage channel, $q = 4$. However, we use q for the sake of generality.

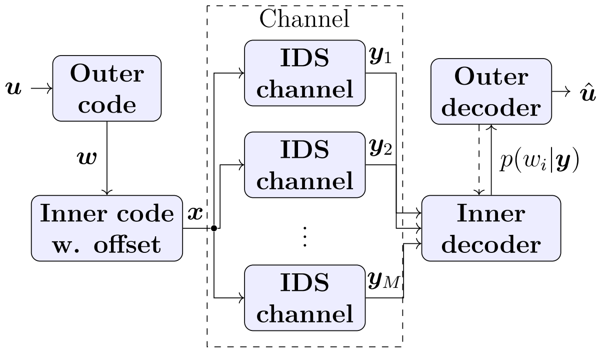


Figure II.2: Block diagram of the encoder and decoder for the DNA storage channel. The DNA storage channel is modeled as multiple reads of the DNA strand transmitted over parallel IDS channels: the channel depicted in Fig. II.1 is fed M times with the DNA sequence $\boldsymbol{x}$. Here, $\boldsymbol{y} = (\boldsymbol{y}_1, \ldots, \boldsymbol{y}_M)$.

The channel enters state x_i where three events may occur: i) an insertion event, with probability p_{I}, where a random symbol $a \in \Sigma_q$ is appended to $\boldsymbol{y}$ instead of x_i. In this case, x_i remains in the queue and the channel returns to state x_i; ii) a deletion event, with probability p_{D}, where symbol x_i is deleted from the queue. In this case, nothing is appended to $\boldsymbol{y}$, the next symbol x_{i+1} is enqueued, and the channel enters state x_{i+1}; iii) a transmission event, with probability $p_{\mathsf{T}} = 1 - p_{\mathsf{I}} - p_{\mathsf{D}}$, where x_i is transmitted. In this case, the symbol x_i is either received with no error with probability $1 - p_{\mathsf{S}}$ or in error with probability p_{S}, in which case x_i is substituted by a random symbol $a \neq x_i$. In either case, the next symbol x_{i+1} is enqueued, and the channel enters state x_{i+1}. The process finishes when the last symbol x_N leaves the queue. The channel output is $\boldsymbol{y}$.

The difference $N - N'$ is referred to as the *drift* [4] at the end of the transmitted sequence. We can also define a drift for each symbol x_i to be transmitted, or each time instant i. Formally, the *symbol-level* drift d_i^{sym}, $0 \leq i < N$, is defined as the difference between the number of insertions and the number of deletions that occurred before symbol x_{i+1} is enqueued, while d_N^{sym} is defined as the number of insertions minus deletions that occurred after the last symbol x_N has been transmitted.

Finally, we model the multiple reads of a DNA sequence resulting from the synthesis and sequencing processes as transmitting the DNA sequence $\boldsymbol{x}$ over M parallel and independent IDS channels, see Fig. II.2, resulting in the received sequences $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_M$.

2.2 Coding Scheme

We consider a concatenated coding scheme with an inner synchronization code depicted in Fig. II.2. First, the information sequence $\boldsymbol{u} = (u_1, \ldots, u_K)$, $u_i \in \mathbb{F}_{q_{\mathsf{o}}}$, is encoded by an $[N_{\mathsf{o}}, K]_{q_{\mathsf{o}}}$ outer code to produce a codeword $\boldsymbol{w} = (w_1, \ldots, w_{N_{\mathsf{o}}})$, $w_i \in \mathbb{F}_{q_{\mathsf{o}}}$, where $\mathbb{F}_{q_{\mathsf{o}}}$ is a binary field extension with $q_{\mathsf{o}} = 2^k_{\text{CC}}$. The codeword $\boldsymbol{w}$ is then encoded by an inner synchronization code. Here, we consider block and convolutional codes for the inner code. We denote the block code by $[n, k_{\mathrm{CC}}, t]_q$, where n and k_{CC} are the length and dimension of the code, respectively, and t represents the number of different codebooks that are used (see [5] for details). Furthermore, the convolutional code is denoted by $(n, k_{\mathrm{CC}}, m)_q$, where m is the number of memory elements. For simplicity, in the rest of the paper we will consider an inner convolutional code for notations and equations. We denote the codeword of the inner code by $\boldsymbol{v} = (v_1, \ldots, v_N)$, $v_i \in \Sigma_q$, which is of length $N = (N_{\mathsf{o}} + m)n$ due to termination of the convolutional code. Finally, a pseudo-random offset sequence is optionally added to $\boldsymbol{v}$ before transmission for synchronization purposes [4, 9], resulting in the sequence $\boldsymbol{x} = (x_1, \ldots, x_N)$. (A detailed explanation of the role of the random sequence in maintaining synchronization and aiding the decoding of the inner code is given in [5].) The DNA sequence $\boldsymbol{x}$ is finally stored in the DNA medium.

The coding scheme rate is measured in bits per DNA symbol (i.e., per nucleotide) and is given by $R = R_{\mathsf{o}} R_{\mathsf{i}} = {}^{Kk}/_{N}$, where $R_{\mathsf{o}} = {}^{K}/_{N_{\mathsf{o}}}$ and $R_{\mathsf{i}} = {}^{N_{\mathsf{o}}k}/_{N}$ are the rates of the outer and inner code, respectively. As we will be only concerned with the drift at time instances that are multiples of n, we define the shorthand $d_i \triangleq d_{in}^{\mathrm{sym}}$. Note that $d_0 = 0$ and $d_{N_{\mathsf{o}}+m} = N' - N$, both known to the receiver.

To recover the information sequence $\boldsymbol{u}$, the inner decoder uses the (noisy) multiple reads $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_M$ of the DNA sequence $\boldsymbol{x}$ to compute (approximate) a posteriori probabilities (APPs) for the symbols in $\boldsymbol{w}$. These APPs are then fed to the outer decoder, which decides on the decoded sequence $\hat{\boldsymbol{u}}$. Furthermore, we can also iterate between the inner and outer decoder, exchanging extrinsic information between them, which is referred to as *turbo decoding* in the literature.

3 Bound on the Finite Blocklength Performance

In this section, we provide an upper bound to the frame error probability, denoted by $P_{\mathsf{f}}(e)$, achievable over the DNA storage channel in the finite blocklength regime. In particular, we consider the DT bound [7]. The bound we provide is tailored to concatenated coding schemes with an inner synchronization code and depends on the inner code. Hence, it can be used to guide its choice and serves as a benchmark to compare coding schemes.

The DT bound for the combination of the inner code and the DNA storage channel is given by

$$P_{\mathsf{f}}(e) \leq \mathbb{E}\left[2^{-\left(i(\mathbf{w};\mathbf{y}) - \log_2 \frac{q_{\mathsf{o}}^{N_{\mathsf{o}}} - 1}{2}\right)^+}\right], \tag{II.5}$$

where $(x)^+ \triangleq \max(x, 0)$, $\mathbb{E}[\cdot]$ denotes expectation, $\mathbf{y} = (\mathbf{y}_1, \ldots, \mathbf{y}_M)$ for the multiple sequences case, and

$$i(\mathbf{w}; \mathbf{y}) \triangleq \log_2 \frac{p(\mathbf{y}|\mathbf{w})}{p(\mathbf{y})}$$

is the so-called *information density* with expected value equal to the mutual information between w and y.[7] The distribution of the information density $i(\mathsf{w};\mathsf{y})$ is not known in closed form for the DNA storage channel. However, the right-hand-side of (II.5) can be accurately estimated using the Monte-Carlo approach proposed in [10, 11], which exploits concentration properties of Markov chains to estimate the mutual information between an input process $\boldsymbol{w}$ and an output process $\boldsymbol{y}$ via trellis-based simulations. We can then approximate (II.5) by

$$P_{\mathsf{f}}(e) \lesssim \frac{1}{V} \sum_{(\boldsymbol{w},\boldsymbol{y})} 2^{-(i(\boldsymbol{w};\boldsymbol{y})-(N_{\mathsf{o}}\log_2 q_{\mathsf{o}}-1))^+}, \tag{II.6}$$

where V is the number of pairs $(\boldsymbol{w},\boldsymbol{y})$ considered in the computation.

In the following, we show how to efficiently compute $i(\boldsymbol{w};\boldsymbol{y})$ for fixed $\boldsymbol{w}$ and $\boldsymbol{y}$. We stress that the values of $i(\boldsymbol{w};\boldsymbol{y})$ and, hence, their distribution depends on the choice of the inner code. The information density can be written as

$$i(\boldsymbol{w};\boldsymbol{y}) = -\log_2 p(\boldsymbol{w}) - \log_2 p(\boldsymbol{y}) + \log_2 p(\boldsymbol{w},\boldsymbol{y}), \tag{II.7}$$

where the probabilities $p(\boldsymbol{w})$, $p(\boldsymbol{y})$, and $p(\boldsymbol{w},\boldsymbol{y})$ can be computed using the forward recursion of the symbol-wise maximum a posteriori decoding algorithm on the trellis describing the combination of the inner code and the DNA storage channel [5] (hereafter in this paragraph referred to as simply the inner code for the sake of simplicity). For simplicity, we consider the case of a single sequence, i.e., $M = 1$. However, the approach below can be generalized to $M > 1$ straightforwardly. For $M = 1$, the APP of the outer code symbol w_i can be computed as $p(w_i|\boldsymbol{y}) = \frac{p(w_i,\boldsymbol{y})}{p(\boldsymbol{y})}$. The joint probability $p(w_i,\boldsymbol{y})$ can be computed by marginalizing the trellis states of the inner code that correspond to symbol w_i. Introducing the joint state variable $\sigma_i = (s_i, d_i)$, where s_i denotes the memory state variables of the convolutional code, we obtain

$$p(w_i,\boldsymbol{y}) = \sum_{(\sigma,\sigma'):w_i} p(\boldsymbol{y},\sigma,\sigma'),$$

where σ and σ' denote realizations of the random variables σ_{i-1} and σ_i, respectively. The summation is over all the inner code memory states that correspond to information symbol w_i. Introducing a drift random variable retains the Markov property of the hidden Markov model (HMM) that was lost due to the insertions and deletions [4]. In this new HMM, a transition from time $i-1$ to time i corresponds to a transmission of a vector of symbols $\boldsymbol{x}_{(i-1)n+1}^{in}$, where $\boldsymbol{x}_a^b = (x_a, x_{a+1}, \dots, x_b)$. Further, when transitioning from state d_{i-1} to d_i, the HMM emits $n + d_i - d_{i-1}$ output symbols depending on both the previous and the new drift. As a result, using the Markov property of the underlying trellis, we can factor the joint probability $p(\boldsymbol{y},\sigma,\sigma')$ into three terms as

$$p(\boldsymbol{y},\sigma,\sigma') = p\Big(\boldsymbol{y}_1^{(i-1)n+d},\sigma\Big)p\Big(\boldsymbol{y}_{(i-1)n+d+1}^{in+d'},\sigma'\Big|\sigma\Big)p\Big(\boldsymbol{y}_{in+d'+1}^{N'}\Big|\sigma'\Big).$$

[7]In order to distinguish between random variables and their realizations, w and y denote the random variables corresponding to $\boldsymbol{w}$ and $\boldsymbol{y}$, respectively.

Abbreviating the above terms by $\alpha_{i-1}(\sigma)$, $\gamma_i(\sigma,\sigma')$, and $\beta_i(\sigma')$ in order of appearance, one can deduce the forward and backward recursions

$$\alpha_i(\sigma') = \sum_{\sigma} \alpha_{i-1}(\sigma)\gamma_i(\sigma,\sigma')\,, \tag{II.8}$$

$$\beta_{i-1}(\sigma) = \sum_{\sigma'} \beta_i(\sigma')\gamma_i(\sigma,\sigma')\,, \tag{II.9}$$

where $\gamma_i(\sigma,\sigma') = p(w_i)p(\boldsymbol{y}_{(i-1)n+d+1}^{in+d'}, d' \big| d, s, s')$ can be efficiently computed using a lattice implementation [12].

Now, $\log_2 p(\boldsymbol{y})$ and $\log_2 p(\boldsymbol{w},\boldsymbol{y})$ in (II.7) can be computed based on the forward recursion in (II.8). In particular,

$$p(\boldsymbol{y}) = \sum_{\sigma} p\Big(\boldsymbol{y}_1^{(N_\mathsf{o}+m)n+d}, \sigma\Big) \overset{(a)}{=} \sum_{\sigma} \alpha_{N_\mathsf{o}+m}(\sigma)\,,$$

where (a) follows since $\alpha_i(\sigma) = p\Big(\boldsymbol{y}_1^{in+d}, \sigma\Big)$. The quantity $\log_2 p(\boldsymbol{w},\boldsymbol{y})$ can be computed in a similar manner by restricting the summation in (II.8) to be over all states σ with an outgoing edge to σ' labeled with the input sequence symbol w_i at time i. Since we consider an input sequence of independent and uniformly distributed symbols, the first term $\log_2 p(\boldsymbol{w})$ in (II.7) is equal to $N_\mathsf{o} \log_2 q_\mathsf{o}$. Note that the backward recursion in (II.9) is not required for the computation of the information density, but only for the calculation of the APP $p(w_i|\boldsymbol{y})$ in decoding.

To obtain an estimate of the right-hand-side of (II.5), we randomly generate $\boldsymbol{w}$ and encode it using the inner code to obtain $\boldsymbol{x}$. Then, we pass $\boldsymbol{x}$ through the DNA storage channel to obtain $\boldsymbol{y}$. For each tuple $(\boldsymbol{w},\boldsymbol{y})$, we evaluate $i(\boldsymbol{w};\boldsymbol{y})$ using the defined recursions and the corresponding summand in (II.6). We repeat this procedure V times, each time creating a new random $\boldsymbol{w}$, and average over the outcomes according to (II.6).

4 Concatenated Coding Scheme Design

4.1 Inner Code

We consider four different inner codes: the watermark code introduced in [4], a convolutional code [13], and two time-varying codes (TVCs) recently introduced in [5]. The watermark code is an $[n, k_{\mathrm{CC}}, 1]_q$ block code to which a random sequence is added, which can also be thought of as a TVC with $t = 1$. We will use the TVCs from [5, Tab. I] with $t = 4$ and a minimum Levenshtein distance of 4. The inner coding schemes that we consider are summarized in Table II.1.

4.2 Outer Code

We use protograph-based LDPC codes for the outer code. Formally, the protograph of an LDPC code is a multi-edge-type graph with n_p variable-node (VN) types

Table II.1: Inner Synchronization Code Scheme Selection

Scheme	Inner code	Gen. polynomial	Alt. pattern	Rate
CC	$(1,1,2)_4$ Conv. code with RS	$[5,7]_{\text{oct}}$	-	0.98
WM	$[4,4,1]_4$ Watermark code	-	-	1.0
TVC-1	$[4,4,4]_4$ TVC	-	Random*	1.0
TVC-2	$[4,4,4]_4$ TVC with RS	-	CB1 to CB4*	1.0

*The alternating pattern of the TVC-1 scheme is done by choosing randomly the 4 codebooks, denoted by CB1-CB4, from [5, Tab. I] and avoiding consecutive codebooks. For the TVC-2 scheme, it is simply done by repeating CB1 to CB4 in a round Robin fashion. RS is shorthand for random sequence.

and r_{p} check-node (CN) types. A protograph can be represented by a base matrix

$$\boldsymbol{B} = \begin{pmatrix} b_{0,0} & b_{0,1} & \cdots & b_{0,n_{\mathsf{p}}-1} \\ b_{1,0} & b_{1,1} & \cdots & b_{1,n_{\mathsf{p}}-1} \\ \vdots & \vdots & \cdots & \vdots \\ b_{r_{\mathsf{p}}-1,0} & b_{r_{\mathsf{p}}-1,1} & \cdots & b_{r_{\mathsf{p}}-1,n_{\mathsf{p}}-1} \end{pmatrix},$$

where $b_{i,j}$ is an integer representing the number of edge connections between a type-i VN and a type-j CN. A parity-check matrix $\boldsymbol{H}$ of an LDPC code can be constructed from a protograph by lifting the base matrix $\boldsymbol{B}$. Lifting is the procedure of replacing each nonzero (zero) $b_{i,j}$ with a $Q_{\mathsf{p}} \times Q_{\mathsf{p}}$ permutation (zero) matrix with row and column weight equal to $b_{i,j}$. The LDPC code resulting from the lifting procedure has length $Q_{\mathsf{p}}n_{\mathsf{p}}$ and dimension at least $Q_{\mathsf{p}}(n_{\mathsf{p}} - r_{\mathsf{p}})$. To construct a nonbinary code from the lifted matrix, we randomly assign nonzero entries from $\mathbb{F}_{q_{\mathsf{o}}}$ to the edges of the corresponding Tanner graph.

In this work, we optimize the protograph $\boldsymbol{B}$ using EXIT charts, extended to the DNA storage channel. Particularly, we optimize the protograph for the case of iterations between the decoder of the LDPC code and the decoder of the combination of the inner code and the DNA storage channel. We limit our search to protographs of dimensions 3×6 (larger protographs may lead to better performance). The choice of the protograph is done by considering both the iterative decoding threshold from the EXIT chart, for $p_{\mathsf{I}} = p_{\mathsf{D}}$ and $p_{\mathsf{S}} = 0$, and the frame error rate (FER) performance of the corresponding code ensemble (i.e., by using random permutation matrices for the protograph liftings). More precisely, we sort the protographs from highest to lowest decoding threshold, and then we pick the first protograph (starting from the top of the list) that shows no sign of an error floor above a FER of 10^{-3}. The best protographs from this list are

$$\boldsymbol{B}_1 = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 3 \\ 0 & 1 & 1 & 2 & 1 & 0 \\ 1 & 1 & 1 & 0 & 1 & 1 \end{pmatrix}, \boldsymbol{B}_2 = \begin{pmatrix} 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 0 & 0 \end{pmatrix} \tag{II.10}$$

for the CC and WM, and TVC-1 and TVC-2 inner coding schemes, respectively. We remark that the search provided protographs with a better threshold, but they

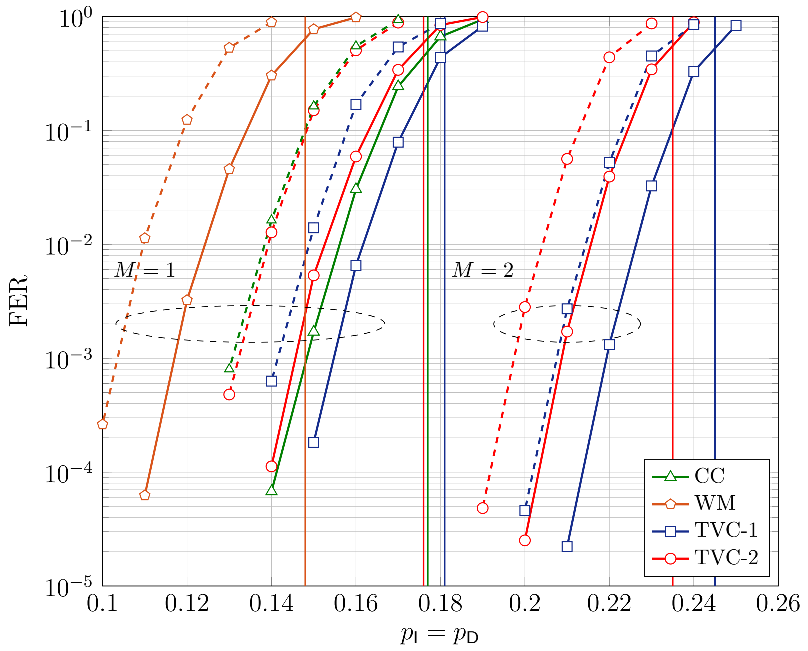


Figure II.3: DT bounds (solid lines with markers) for different inner synchronization codes, $N = 960$ DNA symbols, and $M = 1$ and $M = 2$. The simulated FER performance (dashed lines with markers) are for a concatenated code with an optimized outer LDPC code of rate $R = 1/2$.

all showed a higher error floor than $\boldsymbol{B}_1$ and $\boldsymbol{B}_2$. All protographs were optimized for the case of $M = 1$ and over $\mathbb{F}_{16}$, except for the CC inner coding scheme for which $\mathbb{F}_2$ was used.

5 Numerical Results

In this section, we evaluate the DT bound (II.6) with the inner synchronization codes listed in Table II.1.

5.1 Simulation Parameters

We perform our simulations over the DNA alphabet $\{\mathsf{A}, \mathsf{C}, \mathsf{G}, \mathsf{T}\}$, which corresponds to $q = 4$. We consider the DNA storage channel in Figs. II.1 and II.2 with $p_{\mathsf{S}} = 0$ and $p_{\mathsf{I}} = p_{\mathsf{D}}$ so that the drift random variable has zero mean (however, we remark that similar results are observed for other values and $p_{\mathsf{S}} \neq 0$). To limit the complexity of the decoder of the combination of the inner code and the DNA storage channel, we set the maximum number of consecutive insertions considered by the decoder to 2. Furthermore, we set the limit of the drift random variable to five times the standard deviation of the final drift at position N, i.e., to $5\sqrt{N\frac{p_{\mathsf{D}}}{1-p_{\mathsf{D}}}}$. Note, however, that the simulated channel may introduce more than

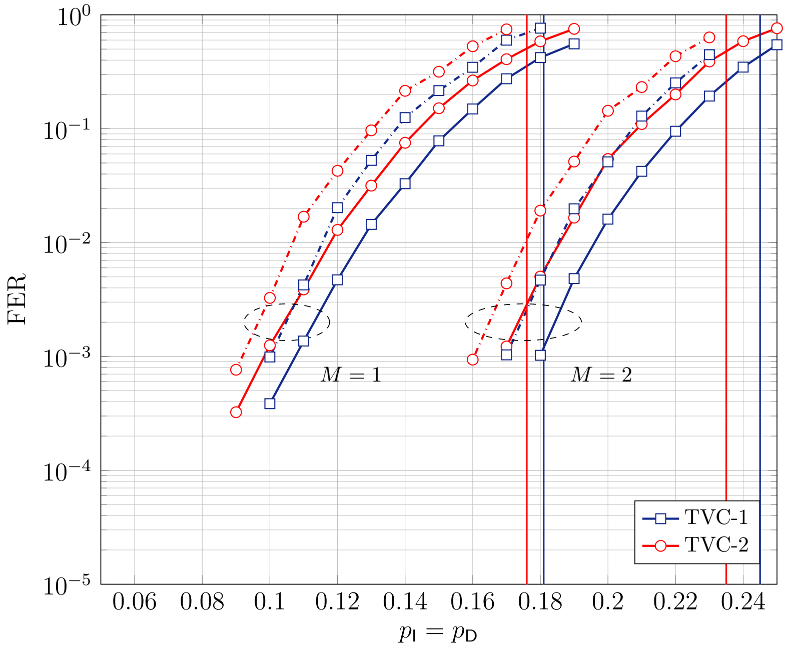


Figure II.4: DT bounds (solid lines with markers) for the TVC-1 and TVC-2 inner coding schemes, $N = 128$ DNA symbols, and $M = 1$ and $M = 2$. The simulated FER performance (dashed lines with markers) are for a concatenated code with an optimized outer LDPC code of rate $R = 1/2$.

two consecutive insertions and lead to a larger drift. The outer LDPC code is decoded with belief propagation with a maximum number of 100 iterations, and the maximum number of turbo iterations is set to 100.

We compute the DT bound for two code lengths, $N = 960$ and $N = 128$ DNA symbols, corresponding to a short and a medium-length sequence, respectively, and for $M = 1$ and $M = 2$ reads. The choice of these lengths is motivated by the current DNA sequencing technologies. All inner codes are of rate (or close to) $R_\mathsf{i} = 1$ (in bits per DNA symbol) and all outer codes are of rate $R_\mathsf{o} = 1/2$.

5.2 Discussion

In Figs. II.3 and II.4, we plot the DT bound (solid lines with markers) for the DNA storage channel with the inner synchronization codes in Table II.1 for $N = 960$ and $N = 128$, respectively. The bound for $M = 2$ is obtained by considering the *joint decoding* algorithm proposed in [5]. Furthermore, in the figures we plot the asymptotic achievable information rates (vertical lines) computed in [5] for each inner coding scheme.

The TVC-1 scheme yields the best bound for both code lengths and values of M, and the watermark code gives the worst bound. Interestingly, the hierarchy of the bounds coincides with the hierarchy of the asymptotic achievable information rates.

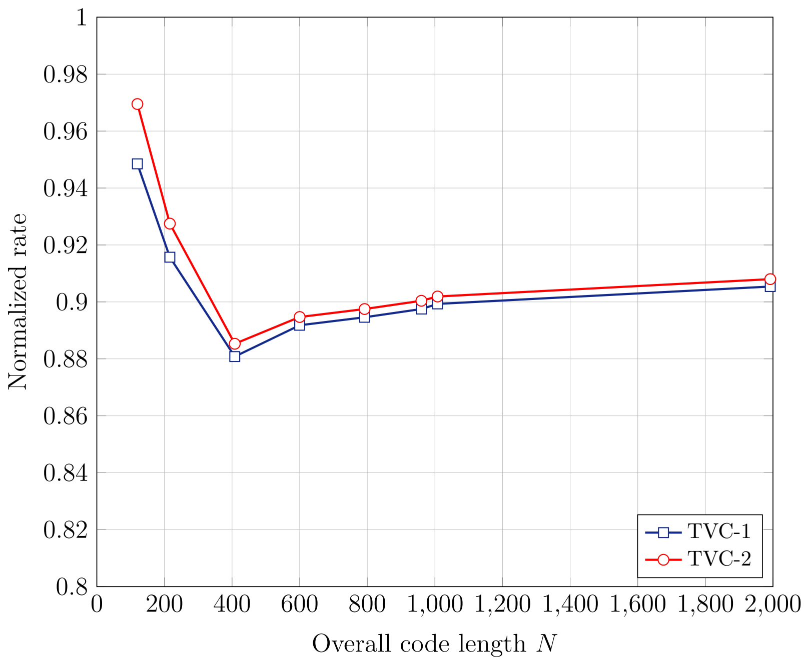


Figure II.5: Normalized rate for a concatenated coding scheme with an optimized outer LDPC code constructed from the protograph $\boldsymbol{B}_2$ in (II.10) and the TVC-1 and TVC-2 inner coding schemes as a function of the code length N. The overall code rate is $R = 1/2$ and the target FER is 10^{-3}.

In the figures, we also plot the FER performance (dashed lines with markers) for a concatenated code with an outer LDPC code built from the optimized protographs in (II.10) and the inner coding schemes from Table II.1. In contrast to the optimization, circulant matrices for the protograph liftings, built using the progressive edge-growth algorithm [14], are used. The slope of the FER curves is similar to the slope of the corresponding DT bounds, and a similar gap to the bounds is observed for the simulated FER curves. Notably, the proposed concatenated schemes perform close to the DT bounds.

To gain more insight on the performance of the proposed concatenated schemes to the DT bound, in Fig. II.5, we plot the normalized rate [7] as a function of the code length N for the concatenated code with the TVC-1 and TVC-2 inner coding schemes. The normalized rate is computed as the fraction between the rate of the concatenated code and the maximum rate provided by the DT bound so that decoding with a probability of error below a given value is possible. In other words, we want a normalized rate close to one and a normalized rate of one means that the code achieves the DT bound. In the plot, we consider a FER of 10^{-3}.

For both TVC-1 and TVC-2, the normalized rate is within 87% to 97% for a code length up to $N = 2000$ DNA symbols. These values are similar to those for state-of-the-art codes over memoryless channels [7, Fig. 15], indicating that the proposed concatenated codes yield excellent performance on the DNA storage channel.

6 Conclusion

We provided an upper bound to the performance of random coding schemes on a DNA storage channel with insertions, deletions, and substitutions in the practical short-to-medium blocklength regime. The bound, which is based on the dependency testing bound yields an achievability result and is particularly useful to capture the performance of concatenated coding schemes with an inner synchronization code as it depends on the inner code. Hence, it is a handy tool to guide the choice of the inner synchronization code and provides a reference to benchmark the performance of coding schemes.

References

[1] S. M. H. T. Yazdi, R. Gabrys, and O. Milenkovic, "Portable and error-free DNA-based data storage," *Sci. Rep.*, vol. 7, no. 5011, Jul. 2017.

[2] L. Organick *et al.*, "Random access in large-scale DNA data storage," *Nature Biotechnol.*, vol. 36, no. 3, pp. 242–248, Mar. 2018.

[3] G. M. Church, Y. Gao, and S. Kosuri, "Next-generation digital information storage in DNA," *Science*, vol. 337, no. 6102, p. 1628, Aug. 2012.

[4] M. C. Davey and D. J. C. MacKay, "Reliable communication over channels with insertions, deletions, and substitutions," *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 687–698, Feb. 2001.

[5] I. Maarouf, A. Lenz, L. Welter, A. Wachter-Zeh, E. Rosnes, and A. Graell i Amat, "Concatenated codes for multiple reads of a DNA sequence," *IEEE Trans. Inf. Theory*, 2022.

[6] S. R. Srinivasavaradhan, S. Gopi, H. D. Pfister, and S. Yekhanin, "Trellis BMA: Coded trace reconstruction on IDS channels for DNA storage," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Melbourne, VIC, Australia, Jul. 2021, pp. 2453–2458.

[7] Y. Polyanskiy, H. V. Poor, and S. Verdú, "Channel coding rate in the finite blocklength regime," *IEEE Trans. Inf. Theory*, vol. 56, no. 5, pp. 2307–2359, May 2010.

[8] J. A. Briffa, H. G. Schaathun, and S. Wesemeyer, "An improved decoding algorithm for the Davey-MacKay construction," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Cape Town, South Africa, May 2010.

[9] V. Buttigieg and J. A. Briffa, "Codebook and marker sequence design for synchronization-correcting codes," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Saint Petersburg, Russia, Jul./Aug. 2011, pp. 1579–1583.

[10] D. M. Arnold, H.-A. Loeliger, P. O. Vontobel, A. Kavčić, and W. Zeng, "Simulation-based computation of information rates for channels with memory," *IEEE Trans. Inf. Theory*, vol. 52, no. 8, pp. 3498–3508, Aug. 2006.

[11] H. D. Pfister, J. B. Soriaga, and P. H. Siegel, "On the achievable information rates of finite state ISI channels," in *Proc. IEEE Glob. Telecommun. Conf. (GLOBECOM)*, San Antonio, TX, USA, Nov. 2001, pp. 2992–2996.

[12] L. R. Bahl and F. Jelinek, "Decoding for channels with insertions, deletions, and substitutions with applications to speech recognition," *IEEE Trans. Inf. Theory*, vol. 21, no. 4, pp. 404–411, Jul. 1975.

[13] M. F. Mansour and A. H. Tewfik, "Convolutional decoding in the presence of synchronization errors," *IEEE J. Sel. Areas Commun.*, vol. 28, no. 2, pp. 218–227, Feb. 2010.

[14] X.-Y. Hu, E. Eleftheriou, and D.-M. Arnold, "Progressive edge-growth Tanner graphs," in *Proc. IEEE Glob. Telecommun. Conf. (GLOBECOM)*, San Antonio, TX, USA, Nov. 2001, pp. 995–1001.

Paper III

Achievable Information Rates and Concatenated Codes for the DNA Nanopore Sequencing Channel

Issam Maarouf, Eirik Rosnes, and Alexandre Graell i Amat

IEEE Information Theory Workshop (ITW), Saint-Malo, France, Apr. 2023.

The layout has been revised.

Abstract

The errors occurring in DNA-based storage are correlated in nature, which is a direct consequence of the synthesis and sequencing processes. In this paper, we consider the memory-k nanopore channel model recently introduced by Hamoum *et al.*, which models the inherent memory of the channel. We derive the maximum a posteriori (MAP) decoder for this channel model. The derived MAP decoder allows us to compute achievable information rates for the true DNA storage channel assuming a mismatched decoder matched to the memory-k nanopore channel model, and quantify the loss in performance assuming a small memory length—and hence limited decoding complexity. Furthermore, the derived MAP decoder can be used to design error-correcting codes tailored to the DNA storage channel. We show that a concatenated coding scheme with an outer low-density parity-check code and an inner convolutional code yields excellent performance.

1 Introduction

Storing data in deoxyribonucleic acid (DNA) promises unprecedented density and durability and is seen as the new frontier of data storage. Recent experiments have already demonstrated the viability of DNA-based data storage [1–3].

The DNA storage channel suffers from multiple impairments and constraints due to the synthesis and sequencing of DNA and the limitations of current technologies. In particular, errors in the form of insertions, deletions, and substitutions (IDS) occur. This has spurred a great deal of research on devising coding schemes for the DNA storage channel.

While the literature on error-correcting coding for the DNA storage channel is abundant, most works have considered a simplified and unrealistic channel model with a small and fixed number of insertions and/or deletions, i.e., an *adversarial* channel. Fewer works have considered a more realistic probabilistic channel model in which errors occur with a given probability, e.g., [4–16]. These works usually assume independent and identically distributed (i.i.d.) errors. In DNA storage, however, IDS errors are not i.i.d., but the channel has memory [17, 18]. Potentially, the memory is as large as the whole DNA sequence.

In [17], the authors proposed a statistical model of the DNA storage channel with nanopore sequencing based on a Markov chain that models the inherent memory of the channel. In particular, it builds on the fact that in the MinION technology strands traverse a nanopore nucleotide by nucleotide, and an electrical signal is generated for every group of $k \geq 1$ successive nucleotides, called $k$mers. Let $\boldsymbol{x}$ be the DNA strand to be synthesized and $\boldsymbol{z}$ be the sequence of channel events, where $z_i \in \{\texttt{insertion}, \texttt{deletion}, \texttt{substitution}, \texttt{no error}\}$. The key idea in [17] is then to assume that z_i depends on the symbols $x_{i-k+1}, \ldots, x_i$ and the previous event z_{i-1} and estimate the probabilities $p(z_i|x_{i-k+1}, \ldots, x_i, z_{i-1})$ from experimental data. We refer to this channel model as the memory-k nanopore channel model.

The starting point of this paper is the work [17]. Our main contributions are as follows.

- We derive the optimum maximum a posteriori (MAP) decoder for the memory-k nanopore channel model. The complexity of the decoder increases exponentially with k. Based on this decoder, we derive achievable information rates (AIRs).
- The AIR for the memory-k nanopore channel can be seen as an AIR for the true DNA storage channel of a *mismatched* decoder (i.e., a decoder that is not matched to the true channel) that assumes that the channel is a memory-k nanopore channel. We show that for increasing k, the AIR improves—meaning that the decoder is better matched to the true channel—and eventually saturates. This allows us to quantify the trade-off between decoding complexity and performance loss incurred by the suboptimal decoder.
- The derived MAP decoder can be used to design error-correcting coding schemes tailored to the memory-k nanopore sequencing channel. In particular, we consider the concatenated coding scheme proposed in [4] and multiple reads of the DNA strand, and we optimize the inner and outer code. We show that the concatenated coding scheme of [4] yields excellent performance at rates close to the AIRs.
- We validate the AIRs and simulation results of the memory-k nanopore channel model using the dataset of DNA reads in [6] obtained using Oxford Nanopore Technologies (ONT) sequencing with the MinION technology.

2 The Nanopore Sequencing Channel Model

We consider the memory-k nanopore channel model introduced in [17]. The key property of this channel model is that it removes the assumption of i.i.d. IDS errors. In particular, the occurrence of an error event, or a correct transmission, depends on the previous event (an insertion, deletion, substitution, or no error event), and previous channel input symbols. Let $\boldsymbol{x} = (x_1, \ldots, x_N)$, $x_i \in \Sigma_4 = \{0, 1, 2, 3\}$, be the channel input sequence and $\boldsymbol{y} = (y_1, \ldots, y_{N'})$ the channel output sequence, and define $x_{N+1} = 0$. Note that N' is random and depends on the number of insertion and deletion events, i.e., $N' \neq N$ in general. Also, let $k\mathrm{mer}_i = (x_{i-k+1}, \ldots, x_i)$ be the $k$mer at time instant $i \geq k$, while for $i < k$, since no complete $k$mer is available, we use $k\mathrm{mer}_i = x_i$. The channel model is depicted in Fig. III.1. Here, we denote the event associated with $k\mathrm{mer}_{i+1}$ by z_i, where z_i represents an insertion, deletion, substitution, or no error acting on symbol x_i when it is to be transmitted. Hence, $z_i \in \{\texttt{Ins}, \texttt{Del}, \texttt{Sub}, \texttt{NoErr}\}$. The transition probabilities p_1, p_2, p_3, and p_4 in the figure are of the form $p(z_i|k\mathrm{mer}_i, z_{i-1})$, and their value depends on the $k$mer. In the event of a deletion, the symbol x_i in $k\mathrm{mer}_i$ will be deleted and nothing will be appended to $\boldsymbol{y}$, while when a no error event occurs, $\boldsymbol{y}$ will be

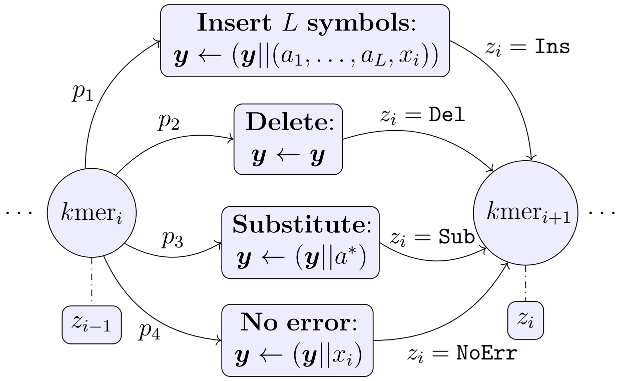

Insert L symbols: $\boldsymbol{y} \leftarrow (\boldsymbol{y}||(a_1,\ldots,a_L,x_i))$
$z_i = \texttt{Ins}$
p_1
p_2
Delete: $\boldsymbol{y} \leftarrow \boldsymbol{y}$
$z_i = \texttt{Del}$
$\cdots$ $k\text{mer}_i$
$k\text{mer}_{i+1}$ $\cdots$
p_3
Substitute: $\boldsymbol{y} \leftarrow (\boldsymbol{y}||a^*)$
$z_i = \texttt{Sub}$
z_{i-1}
p_4
No error: $\boldsymbol{y} \leftarrow (\boldsymbol{y}||x_i)$
z_i
$z_i = \texttt{NoErr}$

Figure III.1: The memory-k nanopore channel in [61].

appended with x_i. Furthermore, in the event of an insertion, i.e., $z_i = \texttt{Ins}$, the channel will insert $L > 1$ symbols $a_1, \ldots, a_L$, where the length L is random. Each symbol a_j will be uniformly picked from Σ_4. In this case, the channel appends $\boldsymbol{y}$ with $(a_1, \ldots, a_L)$. This also applies to a substitution event where the symbol a^* is uniformly picked from $\Sigma_4 \setminus \{x_i\}$ and appended to $\boldsymbol{y}$ instead of x_i. Moreover, we define $p(L|k\text{mer}_i, z_i = \texttt{Ins})$ as the probability of inserting L symbols given $k\text{mer}_i$ and an insertion event, and $p(a^*|k\text{mer}_i, z_i = \texttt{Sub})$ as the probability of x_i being substituted by a^* given $k\text{mer}_i$ and a substitution event. In all these scenarios, the channel always moves from state $k\text{mer}_i$ to state $k\text{mer}_{i+1}$. After the last symbol x_N has been processed by the channel, the channel outputs $\boldsymbol{y}$.

2.1 Channel Transition Probability Estimation

The transition probabilities of the channel model can be estimated using a DNA storage dataset $\mathcal{D}$ containing input (before synthesis) and the corresponding multiple output (after sequencing) sequences.

Let $\boldsymbol{x}^{(l)}$ denote the l-th input sequence and $\boldsymbol{y}_1^{(l)}, \ldots, \boldsymbol{y}_{M_l}^{(l)}$ the corresponding $M_l \geq 1$ output sequences from the dataset $\mathcal{D}$. The method we use to estimate the transition probabilities is as follows. For each pair $(\boldsymbol{x}^{(l)}, \boldsymbol{y}_j^{(l)})$, we compute the edit distance between the two vectors using a lattice (see [4, Sec. III],[19]). By backtracking from the end of the lattice, a corresponding sequence of events $\boldsymbol{z}_j^{(l)}$ can be identified. In case of ties, one event is chosen uniformly at random. Finally, we estimate the probabilities $p(z_i = z|k\text{mer}_i = k\text{mer}, z_{i-1} = z')$ by a simple counting argument. In particular, let

$$\mathcal{S}_{z,z',k\text{mer}}^{(l,j)} = \left\{k \leq u < N : (x_{u-k+1}^{(l)}, \ldots, x_u^{(l)}) = k\text{mer}, (z_{j,u-1}^{(l)}, z_{j,u}^{(l)}) = (z', z)\right\}.$$

Then, for $k \leq i < N$, $p(z_i = z|k\text{mer}_i = k\text{mer}, z_{i-1} = z')$ is estimated by

$$\frac{\sum_{l=1}^{|\mathcal{D}|} \sum_{j=1}^{M_l} |\mathcal{S}_{z,z',k\text{mer}}^{(l,j)}|}{\sum_{z \in \{\texttt{Ins},\texttt{Del},\texttt{Sub},\texttt{NoErr}\}} \sum_{l=1}^{|\mathcal{D}|} \sum_{j=1}^{M_l} |\mathcal{S}_{z,z',k\text{mer}}^{(l,j)}|}.$$

As a consequence, the channel parameters do not vary with i for $k \leq i < N$. For $1 < i < k$, we estimate the channel transition probabilities in a similar manner as

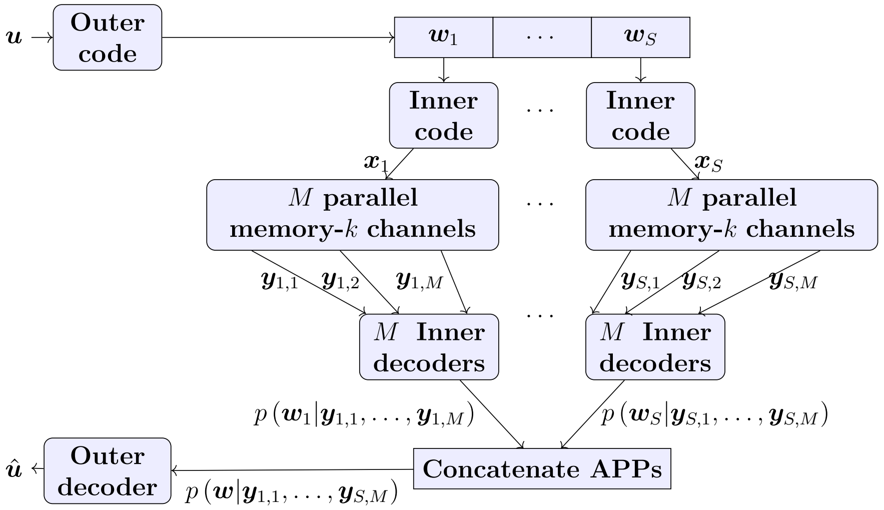


Figure III.2: System model including the coding scheme and the memory-k nanopore channel model depicted in Fig. III.1.

above, but considering that $k\text{mer}_i = x_i$. Moreover, as in [17], we consider $i = 1$ (the beginning) and $i = N$ (the end) separately from the middle $(1 < i < N)$ when estimating the channel parameters as the probability to get an error is typically higher at the beginning and at the end compared to the middle.

2.2 Multiple Reads

We consider the relevant scenario in which the sequencing process outputs multiple reads (M) of the DNA strand. In particular, we model this by assuming that a strand is transmitted over M independent channels.

3 Coding Scheme and System Model

We consider the concatenated coding scheme and the low-complexity *separate decoding* scheme proposed in [4], which decodes separately each noisy strand and combines a posteriori probabilities (APPs) from all reads before passing them to the outer decoder. The primary goal of the inner code is to maintain synchronization with the transmitted sequence and provide likelihoods to the outer decoder. The outer code then corrects remaining errors.

The system model is shown in Fig. III.2. We consider a low-density parity-check (LDPC) code for the outer code and a convolutional code for the inner code. The information data $\boldsymbol{u} = (u_1, \dots, u_K)$, $u_i \in \mathbb{F}_{q_\mathsf{o}}$, of length K, is first encoded by an $[N_\mathsf{o}, K]_{q_\mathsf{o}}$ outer LDPC code over $\mathbb{F}_{q_\mathsf{o}}$ into the codeword $\boldsymbol{w} = (w_1, \dots, w_{N_\mathsf{o}})$, $w_i \in \mathbb{F}_{q_\mathsf{o}}$, where $\mathbb{F}_{q_\mathsf{o}}$ is a binary extension field of size $2^{k_{\text{CC}}}$ and k_{CC} is the (binary) dimension of the inner convolutional code. Since current sequencing technologies cannot handle long sequences, the codeword $\boldsymbol{w}$ is split into S subsequences $\boldsymbol{w}_1, \dots, \boldsymbol{w}_S$, each of length $N_\mathsf{o}^\mathsf{b} = {}^{N_\mathsf{o}}\!/\!_S$, where $\boldsymbol{w}_j = (w_{(j-1)N_\mathsf{o}^\mathsf{b}+1}, \dots, w_{jN_\mathsf{o}^\mathsf{b}})$. Each $\boldsymbol{w}_j$ is encoded, with an optional offset [4], by an $(n, k_{\mathrm{CC}}, m)_4$ inner convolutional code of binary dimension k_{CC}, memory m, and output alphabet Σ_4 into the inner codeword $\boldsymbol{x}_j$ of length $N = (N_{\mathsf{o}}^{\mathsf{b}} + m)n$, where m is due to terminating the convolutional code to the all-zero state. The overall code rate is given as $R = R_{\mathsf{o}}R_{\mathsf{i}} = {}^{Kk_{\mathrm{CC}}}/_{SN}$, where $R_{\mathsf{o}} = {}^{K}/_{N_{\mathsf{o}}}$ is the rate of the outer code and $R_{\mathsf{i}} = {}^{k_{\mathrm{CC}}N_{\mathsf{o}}^{\mathsf{b}}}/_{N}$ the rate of the inner code.

Each $\boldsymbol{x}_j$ is transmitted independently over M identical memory-k nanopore channels in order to model the multiple copies of a DNA strand at the output of the sequencing process. At the receiver, each noisy read $\boldsymbol{y}_{j,1}, \ldots, \boldsymbol{y}_{j,M}$ for transmitted sequence $\boldsymbol{x}_j$ is decoded separately with a MAP inner decoder. The APPs at their output are then combined to provide approximate APPs for $\boldsymbol{w}_j$. The (approximate) APPs for the symbols in $\boldsymbol{w} = (\boldsymbol{w}_1, \ldots, \boldsymbol{w}_S)$ are concatenated and then passed to the outer decoder. The outer decoder uses these APPs to decide on an estimate $\hat{\boldsymbol{u}}$ of $\boldsymbol{u}$.

4 Symbolwise MAP Decoding for Channel/Inner Code (Inner Decoding)

In this section, we derive the optimum (MAP) decoder for the combination of the memory-k nanopore channel model and the inner code. To this end, we use the fact that the combination of the channel and the inner code can be seen as a hidden Markov model (HMM) by introducing a *drift* variable as in [7]. The drift d_i, $0 \le i < N_{\mathsf{o}}^{\mathsf{b}} + m$, is defined as the number of insertions minus the number of deletions that occurred before symbol x_{ni+1} is to be acted on by the event z_{ni+1}, while $d_{N_{\mathsf{o}}^{\mathsf{b}}+m}$ is defined as the number of insertions minus deletions that occurred after the last symbol x_N has been processed by the channel. Thus, by definition, $d_0 = 0$ and $d_{N_{\mathsf{o}}^{\mathsf{b}}+m} = N' - N$, both known to the decoder. Adding the drift to the joint state of the channel and the inner code, i.e., to $(k\mathrm{mer}_{i+1}, z_i, s_i)$, where s_i denotes the state variables of the inner convolutional code, gives the state variable $\sigma_i = (k\mathrm{mer}_{i+1}, z_i, s_i, d_i)$ of the HMM. In this HMM, a transition from time $i-1$ to time i corresponds to a transmission of symbols $\boldsymbol{x}_{(i-1)n+1}^{in}$, where $\boldsymbol{x}_a^b = (x_a, x_{a+1}, \ldots, x_b)$. Further, when transitioning from a state with drift d_{i-1} to a state with drift d_i, the HMM emits $n + d_i - d_{i-1}$ output symbols depending on both the previous and new drift.

In the following, to simplify notation, the subsequence index of $\boldsymbol{w}$ and $\boldsymbol{y}$ is omitted and w_i simply refers to the i-th symbol of an arbitrary subsequence $\boldsymbol{w}_j$, while $\boldsymbol{y} = (\boldsymbol{y}_1, \ldots, \boldsymbol{y}_M)$ refers to the corresponding received sequences.

4.1 Decoding for a Single Received Sequence

The APP for outer code symbol w_i can be computed as $p(w_i|\boldsymbol{y}) = {}^{p(\boldsymbol{y},w_i)}/_{p(\boldsymbol{y})}$, where the joint probability $p(\boldsymbol{y}, w_i)$ can be computed by marginalizing the trellis states, corresponding to the HMM, of the channel/inner code that correspond to symbol w_i. Then, we can write $p(\boldsymbol{y}, w_i) = \sum_{(\sigma,\sigma'):w_i} p(\boldsymbol{y}, \sigma, \sigma')$, where σ and σ' are realizations of the random variables σ_{i-1} and σ_i, respectively. Here, the summation is over all pairs of states that correspond to symbol w_i. We can use the Markov property to decompose the probability $p(\boldsymbol{y}, \sigma, \sigma')$ into three parts as

$$p(\boldsymbol{y},\sigma,\sigma')=p\Big(\boldsymbol{y}_1^{(i-1)n+d},\sigma\Big)p\Big(\boldsymbol{y}_{(i-1)n+d+1}^{in+d'},\sigma'\Big|\sigma\Big)p\Big(\boldsymbol{y}_{in+d'+1}^{N'}\Big|\sigma'\Big).$$

We abbreviate the first, second, and third term of the above equation with $\alpha_{i-1}(\sigma)$, $\gamma_i(\sigma, \sigma')$, and $\beta_i(\sigma')$, respectively. Then, the first and third term can computed recursively as

$$\alpha_i(\sigma') = \sum_{\sigma} \alpha_{i-1}(\sigma)\gamma_i(\sigma, \sigma'), \beta_{i-1}(\sigma) = \sum_{\sigma'} \beta_i(\sigma')\gamma_i(\sigma, \sigma').$$

The term $\gamma_i(\sigma, \sigma')$ (the branch metric) can be decomposed as

$$\gamma_i(\sigma, \sigma') = p(w_i)p(z'|k\text{mer}, z)p\left(\boldsymbol{y}_{(i-1)n+d+1}^{in+d'}, d'\Big|d, s, s', z', z, k\text{mer}', k\text{mer}\right) ,$$

where $p(w_i)$ is the a priori probability of symbol w_i. For simplicity, define the state variable $\zeta_i = (k\text{mer}_{i+1}, z_i)$. To compute $p(\boldsymbol{y}_{(i-1)n+d+1}^{in+d'}, d'\big|d, s, s', \zeta', \zeta)$, we need to consider each possible event for z'. For simplicity, we limit our derivation to $n = 1$. The case for general n follows in a straightforward manner.

1. $z' = \texttt{Ins}$. Then,

$$\begin{aligned} p\left(\boldsymbol{y}_{(i-1)+d+1}^{i+d'}, d'\Big|d, s, s', \zeta', \zeta\right) &= p\left(\boldsymbol{y}_{(i-1)+d+1}^{i+d'}, L = d' - d\Big|d, s, s', \zeta', \zeta\right) \\ &\quad \cdot p\left(L = d' - d\Big|d, s, s', \zeta', \zeta\right) \\ &= \left(\frac{1}{4}\right)^L \cdot p(L|k\text{mer}, z' = \texttt{Ins}) \,. \end{aligned}$$

2. $z' = \texttt{Del}$. Then, $d' = d - 1$, which means that $p(\boldsymbol{y}_{(i-1)+d+1}^{i+d'}, d' \neq d - 1\big|d, s, s', \zeta', \zeta) = 0$ and $p(\boldsymbol{y}_{(i-1)+d+1}^{i+d'}, d' = d - 1\big|d, s, s', \zeta', \zeta) = 1$.

3. $z' = \texttt{Sub}$. Then, $d' = d$, which means that $p(\boldsymbol{y}_{(i-1)+d+1}^{i+d'}, d' \neq d\big|d, s, s', \zeta', \zeta) = 0$, and

$$\begin{aligned} &p\left(\boldsymbol{y}_{(i-1)+d+1}^{i+d'}, d' = d\Big|d, s, s', \zeta', \zeta\right) \\ &= \begin{cases} p\left(\boldsymbol{y}_{i+d}^{i+d}\Big|k\text{mer}, s, s', z' = \texttt{Sub}\right) & \text{if } \boldsymbol{y}_{i+d}^{i+d} \neq \boldsymbol{x}_{i+d}^{i+d} \\ 0 & \text{otherwise} \,. \end{cases} \end{aligned}$$

4. $z' = \texttt{NoErr}$. Then, $d' = d$, which means that $p(\boldsymbol{y}_{(i-1)+d+1}^{i+d'}, d' \neq d\big|d, s, s', \zeta', \zeta) = 0$, and

$$p\left(\boldsymbol{y}_{(i-1)+d+1}^{i+d'}, d' = d\Big|d, s, s', \zeta', \zeta\right) = \begin{cases} 1 & \boldsymbol{y}_{i+d}^{i+d} = \boldsymbol{x}_{i+d}^{i+d} \\ 0 & \text{otherwise} \,. \end{cases}$$

4.2 Decoding for Multiple Received Sequences

We consider the separate decoding strategy for multiple received sequences $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_M$ proposed in [4]. Following [4], the APP $p(w_i|\boldsymbol{y}_1, \ldots, \boldsymbol{y}_M)$ can be approximated as

$$p(w_i|\boldsymbol{y}_1, \ldots, \boldsymbol{y}_M) \propto \frac{\prod_{j=1}^{M} p(w_i|\boldsymbol{y}_j)}{p(w_i)^{M-1}} \, ,$$

where $p(w_i|\boldsymbol{y}_j)$ is computed as outlined in Section 4.1. This decoder, although suboptimal, is efficient and practical for our scenario, as its complexity grows linearly with M [4].[8]

4.3 Decoding Complexity

Since the overall decoding complexity is dominated by the combination of the inner code and the channel, we will disregard the complexity of the outer decoder in the complexity analysis. In order to limit the inner decoding complexity, we limit the drift d_i to a fixed interval $[d_{\min}, d_{\max}]$ and the number of insertions per symbol L to $L_{\max}$; recall also that $L > 1$. For simplicity, we limit our derivation to $n = 1$. The complexity of the BCJR algorithm on the joint trellis of the inner code and the channel is directly proportional to the number of trellis edges at each trellis section, which is upper bounded by $2^{\nu + k_{\mathrm{CC}}} 4^{k+1} \Delta(\delta + 1)$, where $\Delta = d_{\max} - d_{\min} + 1$ is the number of drift states, $\delta = L_{\max} + 1$ is the number of possible drift transitions, and ν is the number of binary memory elements of the convolutional encoder. Hence, the complexity of decoding a single block is $(N_{\mathsf{o}}^{\mathsf{b}} + m) 2^{\nu + k_{\mathrm{CC}}} 4^{k+1} \Delta(\delta + 1)$, and the complexity of separate decoding (for all S blocks) becomes $S(N_{\mathsf{o}}^{\mathsf{b}} + m) 2^{\nu + k_{\mathrm{CC}}} 4^{k+1} \Delta(\delta + 1) M$. Note that the complexity increases exponentially with the channel memory k.

5 Achievable Information Rates

The MAP decoder for the memory-k nanopore channel (including the inner code) derived in Section 4 allows us to compute AIRs for this channel. In particular, we compute *BCJR-once* rates [20–22], defined as the symbolwise mutual information between the input of the channel and the log-likelihood ratios (LLRs) produced by a symbolwise MAP (i.e., optimum) detector. For a given inner code, the BCJR-once rate, denoted by $R_{\text{BCJR-once}}$, is a rate achievable by an outer code that does not exploit possible correlations between the LLRs and when no iterations between the inner and outer decoder are performed.

The BCJR-once rate under separate decoding can be estimated as

$$R_{\text{BCJR-once}} \approx R_{\mathsf{i}} \log q_{\mathsf{o}} + \frac{R_{\mathsf{i}}}{N_{\mathsf{o}}^{\mathsf{b}} + m} \sum_{i=1}^{N_{\mathsf{o}}^{\mathsf{b}} + m} \log \frac{\mathrm{e}^{L_i^{\text{BCJR-sep}}(w_i)}}{\sum_{a \in \mathbb{F}_{q_{\mathsf{o}}}} \mathrm{e}^{L_i^{\text{BCJR-sep}}(a)}}$$

[8]Alternatively, one may decode all M reads jointly using a single inner decoder [4]. However, the complexity of this decoder grows exponentially with the number of sequences M, and becomes infeasible for $M > 2$.

Table III.1: Optimized Protographs Found by DE for Different M

M	$R = R_{\mathsf{o}} R_{\mathsf{i}}$	Protograph
1	${}^{8}/_{10} \cdot {}^{3}/_{2} = {}^{6}/_{5}$	$\left(\begin{smallmatrix} 1 & 2 & 0 & 1 & 2 & 2 & 1 & 1 & 1 & 3 \\ 2 & 0 & 3 & 2 & 1 & 0 & 1 & 2 & 2 & 0 \end{smallmatrix}\right)$
2	${}^{7}/_{8} \cdot {}^{3}/_{2} = {}^{21}/_{16}$	$\left(\begin{smallmatrix} 2 & 2 & 3 & 3 & 2 & 3 & 3 & 3 \end{smallmatrix}\right)$
5	${}^{14}/_{15} \cdot {}^{3}/_{2} = {}^{7}/_{5}$	$\left(\begin{smallmatrix} 3 & 2 & 3 & 3 & 2 & 2 & 3 & 3 & 2 & 2 & 3 & 3 & 3 & 2 & 3 \end{smallmatrix}\right)$

by sampling an input sequence $\boldsymbol{w}$ and corresponding output sequence $\boldsymbol{y} = (\boldsymbol{y}_1, \dots, \boldsymbol{y}_M)$ and computing the (mismatched) LLRs $L_i^{\text{BCJR-sep}}(a) = \sum_{j=1}^{M} \ln \frac{q(w_i = a|\boldsymbol{y}_j)}{q(w_i = 0|\boldsymbol{y}_j)}$, $a \in \mathbb{F}_{q_{\mathsf{o}}}$, where $q(w_i|\boldsymbol{y}_j)$ is a (mismatched) inner decoding metric.

BCJR-once rates can also be computed for the true DNA storage channel using a dataset of DNA traces by averaging over pairs of input and output sequences. In this case, given an inner code and a decoder that assumes that the channel has memory k, the BCJR-once rate is an AIR for the DNA storage channel of a *mismatched* decoder where the inner decoder is matched to the memory-k nanopore channel model.

For both cases, we assume separate decoding and an inner MAP decoder matched to the memory-k nanopore channel model, i.e., using $q(w_i|\boldsymbol{y}_j) = p(w_i|\boldsymbol{y}_j)$ where $p(w_i|\boldsymbol{y}_j)$ is computed as described in Section 4.1.

6 Concatenated Coding Scheme Design

6.1 Inner Code

For the inner code we use the $(1,1,2)_4$ convolutional code with generator polynomial $g = [5,7]_{\text{OCT}}$ and punctured in order to have a higher rate. In particular, we use the puncturing matrix $\boldsymbol{P} = \left(\begin{smallmatrix} 1 & 0 & 1 \\ 1 & 1 & 0 \end{smallmatrix}\right)$, which gives an inner code rate of $R_{\mathsf{i}} = {}^{3}/_{2}$ (in bits per DNA symbol). Moreover, we add a pseudo-random sequence to the output of the inner code.

6.2 Outer Code

We consider a protograph-based binary LDPC code as the outer code, which we optimize (also in terms of code rate) via density evolution (DE) using the algorithm proposed in [20] for the memory-5 nanopore channel model for $M = 1$, 2, and 5. Moreover, we set $N = 110$ as the dataset in [6] contains input sequences with this length. The optimized protographs, when limiting the entries to at most 3, are shown in Table III.1. The outer LDPC codes are constructed by lifting the protographs using circulants that are optimized using the progressive edge-growth algorithm [23].

7 Numerical Results and Discussion

In this section, we give AIRs and frame error rate (FER) results for our designed optimized concatenated codes for the memory-k nanopore channel model. We further compute AIRs and FER results for a real DNA channel using the experimental dataset in [6]. The dataset consists of 269709 output sequences taken from the output of an ONT MinION sequencer and also the corresponding input sequences (before synthesis). There are in total 10000 input sequences of length $N = 110$.

In all our simulations, we use $L_{\max} = 2$ and $d_{\max} = -d_{\min} = 5\sqrt{N\frac{\max(p_{\mathsf{I}},p_{\mathsf{D}})}{1-\max(p_{\mathsf{I}},p_{\mathsf{D}})}}$, where p_{I} and p_{D} are the average insertion and deletion probabilities based on the dataset in [6]. Moreover, we use separate decoding as described in Section 4.2. The AIRs are computed by averaging over 10000 sequences of length $N = 110$.

In Fig. III.3, we plot BCJR-once rates for the memory-k nanopore channel model (see Section 2) with the convolutional code of Section 6.1 as a synchronization inner code (dashed curves) for different values of k. For each k, the inner decoder is matched to the combination of the inner code and the memory-k nanopore channel model. Further, for each k, we estimated the transition probabilities of the memory-k nanopore channel model as described in Section 2.1 using the dataset in [6]. We observe that the AIRs decrease with increasing k. This is expected, as increasing the memory k makes the channel more complex.

In the figure, we also plot AIRs for the true DNA storage channel using the dataset in [6] (solid curves). (The random sequence added to the output of the inner code can be used to match the transmitted coded sequences to the input sequences of the dataset. Hence, the dataset can be used for the true channel with an inner convolutional code as well.) In this case, the curve for a given value of k corresponds to an AIR for the true DNA storage channel of a mismatched decoder where the inner decoder is matched to the combination of the inner code and the memory-k nanopore channel model. We observe the opposite effect to the AIRs for the memory-k nanopore channel, i.e., the AIRs increase with increasing k. Again, this behavior is expected: If the memory-k nanopore channel model models well the DNA storage channel (i.e., the assumption of a Markovian model is good), increasing k makes the decoder better matched to the true DNA storage channel, hence the AIR increases. Equivalently, a low value of k corresponds to a decoder that is more *mismatched* with respect to the true DNA storage channel, resulting in a lower AIR. Our results hence support that this channel model is good. In fact, although not directly apparent from the figure, the AIRs saturate when k increases, e.g., for the case of no inner code (results not shown here) the AIRs saturate for k around 7. Furthermore, interestingly, comparing the dashed and solid curves, we observe a sandwich effect, where the AIR curve for the memory-k nanopore channel model (dashed curve with green circles) and the AIR for the true DNA storage channel with a mismatched decoder (matched to the combination of the inner code and the memory-5 nanopore channel model) are very close. This indicates that increasing k beyond 5 does not bring much further gains in AIR for the true DNA storage channel.

In Fig. III.3, we also plot the BCJR-once rate for the true DNA storage channel

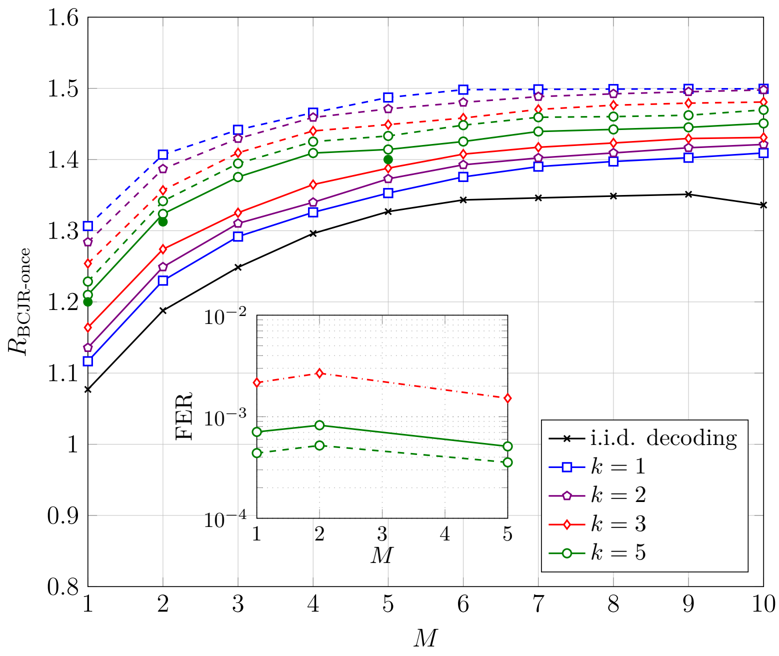


Figure III.3: BCJR-once rates with a rate-3/2 inner convolutional code for different values of M and $N = 110$. Solid curves are for the true channel and dashed curves for the memory-k nanopore channel model. The dash-dotted line is for decoding with $k = 3$ while the channel is set to $k = 5$. The green solid circles represent the rates for the outer LDPC codes with the optimized protographs in Table III.1.

of a decoder that assumes i.i.d. IDS errors (black curve with star markers), as most decoders in the literature. We observe that this results in a significant performance loss.

Finally, in the figure, we also plot the rate R obtained via DE for the true DNA storage channel for the outer LDPC codes with the optimized protographs in Table III.1 (green filled circles), showing that our coding scheme gives excellent performance at code rates close to the BCJR-once rates.

In the inset of Fig. III.3, we plot the FER results of our designed optimized concatenated codes with the inner convolutional code of Section 6.1 and outer LDPC codes of length $N_{\mathsf{o}} = 10000$ based on the protographs in Table III.1. The outer codeword is split into 123 subsequences of length 81, resulting in 110 channel input symbols after the inner encoding, and a single shorter subsequence of length 37. The codes are simulated over both the true DNA storage channel with an inner decoder matched to the memory-5 nanopore channel model (solid curve) and the memory-5 nanopore channel model for a decoder matched to $k = 5$ (green dashed curve with circles) and $k = 3$ (red dashed-dotted curve with diamonds). The results are in agreement with the DE results.

The decoding complexity increases exponentially with k (see Section 4.3). Thus, the AIRs and FER results in Fig. III.3 allow us to quantify the performance loss incurred by a decoder assuming a given memory k and hence the trade-off between decoding complexity and error rate performance. We observe that considering memory 5 incurs almost no loss in terms of AIR, indicating that $k = 5$ is enough.

8 Conclusion

We derived the optimum MAP decoder for the memory-k nanopore channel model. Based on the MAP decoder, we derived AIRs for the true DNA storage channel of a mismatched decoder that is matched to the memory-k model and optimized coding schemes for this channel. We showed that, remarkably, the concatenated coding scheme in [4] (properly optimized) achieves excellent performance for the true DNA storage channel: Considering an optimal inner decoder for the memory-k nanopore channel yields significantly higher AIRs—hence higher storage density—for the true DNA storage channel than a decoder that assumes i.i.d. IDS errors, as usually assumed in the literature.

References

[1] G. M. Church, Y. Gao, and S. Kosuri, "Next-generation digital information storage in DNA," *Sci.*, vol. 337, no. 6102, p. 1628, Aug. 2012.

[2] R. N. Grass, R. Heckel, M. Puddu, D. Paunescu, and W. J. Stark, "Robust chemical preservation of digital information on DNA in silica with error-correcting codes," *Angew. Chem. Int. Ed.*, vol. 54, no. 8, pp. 2552–2555, Feb. 2015.

[3] S. M. H. T. Yazdi, R. Gabrys, and O. Milenkovic, "Portable and error-free DNA-based data storage," *Sci. Rep.*, vol. 7, no. 5011, pp. 1–6, Jul. 2017.

[4] I. Maarouf, A. Lenz, L. Welter, A. Wachter-Zeh, E. Rosnes, and A. Graell i Amat, "Concatenated codes for multiple reads of a DNA sequence," *IEEE Trans. Inf. Theory*, vol. 69, no. 2, pp. 910–927, Feb. 2023.

[5] I. Shomorony and R. Heckel, "Information-theoretic foundations of DNA data storage," *Found. Trends Commun. Inf. Theory*, vol. 19, no. 1, pp. 1–106, Feb. 2022.

[6] S. R. Srinivasavaradhan, S. Gopi, H. D. Pfister, and S. Yekhanin, "Trellis BMA: Coded trace reconstruction on IDS channels for DNA storage," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Melbourne, VIC, Australia, Jul. 2021, pp. 2453–2458.

[7] M. C. Davey and D. J. C. MacKay, "Reliable communication over channels with insertions, deletions, and substitutions," *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 687–698, Feb. 2001.

[8] J. A. Briffa, H. G. Schaathun, and S. Wesemeyer, "An improved decoding algorithm for the Davey-MacKay construction," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Cape Town, South Africa, May 2010.

[9] M. Inoue and H. Kaneko, "Adaptive synchronization marker for insertion/deletion/substitution error correction," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Cambridge, MA, USA, Jul. 2012, pp. 508–512.

[10] R. Shibata, G. Hosoya, and H. Yashima, "Design of irregular LDPC codes without markers for insertion/deletion channels," in *Proc. IEEE Glob. Commun. Conf. (GLOBECOM)*, Waikoloa, HI, USA, Dec. 2019.

[11] H. Koremura and H. Kaneko, "Insertion/deletion/substitution error correction by a modified successive cancellation decoding of polar code," *IEICE Trans. Fundam. Electron., Commun. Comput. Sci.*, vol. E103.A, no. 4, pp. 695–703, Apr. 2020.

[12] R. Sakogawa and H. Kaneko, "Symbolwise MAP estimation for multiple-trace insertion/deletion/substitution channels," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Los Angles, CA, USA, Jun. 2020, pp. 781–785.

[13] R. Shibata, G. Hosoya, and H. Yashima, "Concatenated LDPC/trellis codes: Surpassing the symmetric information rate of channels with synchronization errors," *IEICE Trans. Fundam. Electron., Commun. Comput. Sci.*, vol. E103.A, no. 11, pp. 1283–1291, Nov. 2020.

[14] H. D. Pfister and I. Tal, "Polar codes for channels with insertions, deletions, and substitutions," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Melbourne, VIC, Australia, Jul. 2021, pp. 2554–2559.

[15] M. F. Mansour and A. H. Tewfik, "Convolutional decoding in the presence of synchronization errors," *IEEE J. Sel. Areas Commun.*, vol. 28, no. 2, pp. 218–227, Feb. 2010.

[16] V. Buttigieg and N. Farrugia, "Improved bit error rate performance of convolutional codes with synchronization errors," in *Proc. IEEE Int. Conf. Commun. (ICC)*, London, U.K., Jun. 2015, pp. 4077–4082.

[17] B. Hamoum, E. Dupraz, L. Conde-Canencia, and D. Lavenier, "Channel model with memory for DNA data storage with nanopore sequencing," in *Proc. Int. Symp. Topics Coding (ISTC)*, Montréal, QC, Canada, Aug./Sep. 2021.

[18] B. McBain, E. Viterbo, and J. Saunderson, "Finite-state semi-Markov channels for nanopore sequencing," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Espoo, Finland, Jun./Jul. 2022, pp. 216–221.

[19] L. R. Bahl and F. Jelinek, "Decoding for channels with insertions, deletions, and substitutions with applications to speech recognition," *IEEE Trans. Inf. Theory*, vol. 21, no. 4, pp. 404–411, Jul. 1975.

[20] A. Kavčić, X. Ma, and M. Mitzenmacher, "Binary intersymbol interference channels: Gallager codes, density evolution, and code performance bounds," *IEEE Trans. Inf. Theory*, vol. 49, no. 7, pp. 1636–1652, Jul. 2003.

[21] R. R. Müller and W. H. Gerstacker, “On the capacity loss due to separation of detection and decoding,” *IEEE Trans. Inf. Theory*, vol. 50, no. 8, pp. 1769–1778, Aug. 2004.

[22] J. B. Soriaga, H. D. Pfister, and P. H. Siegel, “Determining and approaching achievable rates of binary intersymbol interference channels using multistage decoding,” *IEEE Trans. Inf. Theory*, vol. 53, no. 4, pp. 1416–1429, Apr. 2007.

[23] X.-Y. Hu, E. Eleftheriou, and D.-M. Arnold, “Progressive edge-growth Tanner graphs,” in *Proc. IEEE Glob. Telecommun. Conf. (GLOBECOM)*, San Antonio, TX, USA, Nov. 2001, pp. 995–1001.

Paper IV

Index-Based Concatenated Codes for the Multi-Draw DNA Storage Channel

Lorenz Walter, Issam Maarouf, Andreas Lenz, Antonia Wachter-Zeh, Eirik Rosnes, and Alexandre Graell i Amat

IEEE Information Theory Workshop (ITW), Saint-Malo, France, Apr. 2023.

The layout has been revised.

Abstract

We consider error-correcting coding for DNA-based storage. We model the DNA storage channel as a *multi-draw IDS channel* where the input data is chunked into M short DNA strands, which are copied a random number of times, and the channel outputs a random selection of N noisy DNA strands. The retrieved DNA strands are prone to insertion, deletion, and substitution (IDS) errors. We propose an index-based concatenated coding scheme consisting of the concatenation of an outer code, an index code, and an inner *synchronization* code, where the latter two tackle IDS errors. We further propose a mismatched joint index-synchronization code maximum a posteriori probability decoder with optional clustering to infer symbolwise a posteriori probabilities for the outer decoder. We compute achievable information rates for the outer code and present Monte-Carlo simulations for information-outage probabilities and frame error rates on synthetic and experimental data, respectively.

1 Introduction

This work aims at improving the reliability of DNA-based data storage. We analyze the *multi-draw IDS channel* which is an abstraction of the synthesis (writing), storage, and sequencing (reading) procedures, including insertion, deletion, and substitution (IDS) errors, thereby narrowing the modeling gap to the real DNA storage channel [1].

The capacity of the DNA storage channel has been studied in [2–5]; see [6] for an overview. For coping with IDS errors in a single-sequence transmission, the classical scheme in [7] with the improved decoding method from [8] is most relevant for our work. Independently, [9, 10] analyzed coded trace reconstruction under IDS errors and showed significant gains by leveraging multiple noisy copies of the same transmit sequence even with sub-optimal decoding techniques.

Inspired by experimental works [11–16], we propose an index-based concatenated coding scheme for the multi-draw IDS channel. Index-based schemes for the multi-draw IDS channel are sub-optimal, albeit very practical [6, 17]. Similar coding approaches are analyzed in [18, 19]. In our scheme, the information is encoded by an outer code to provide overall error protection, primarily coping with unresolved errors and strand erasures of the data. The resulting codeword is split into M data blocks. In each data block, index information is embedded after being encoded by a low-rate index code, whereas the data block itself is encoded by an inner code. Both codes are tailored to deal with IDS errors. The M resulting strands are then transmitted over the multi-draw IDS channel, which models the DNA storage channel. At the receiver, each noisy strand is decoded separately by a joint index and inner maximum a posteriori probability (MAP) decoder outputting symbolwise a posteriori probabilities (APPs). To leverage the multi-copy gain, we combine the APPs according to the subsequent clustering and index decoding. We optionally cluster the received strands based on the obtained APPs, benefiting from the error-correction capabilities of the index and inner codes. For each cluster, an index decision is made by jointly considering the received strands in a cluster. The APPs are ordered according to the index decisions and fed into a soft-input outer decoder.

We analyze our proposed scheme in terms of achievable information rates (AIRs). The AIRs provide insights to the performance of different index-inner coding and decoding techniques and a benchmark for an outer code. For different coding setups in the finite blocklength regime, we compute information-outage probabilities on synthetic data and present frame error rates (FERs) on experimental data from [9].

2 DNA Storage Channel Model

2.1 IDS Channel

We consider a model in which IDS errors are independent and identically distributed. Let $\boldsymbol{x} = (x_1, \dots, x_L)$ and $\boldsymbol{y} = (y_1, \dots, y_{L'})$ be the DNA strand to be synthesized and a single read at the output of the sequencing process, respectively, with $x_t, y_t \in \Sigma_4 = \{\mathsf{A}, \mathsf{C}, \mathsf{G}, \mathsf{T}\}$.

The input to the channel can be seen as a queue in which symbols x_t are successively enqueued for transmission. The output strand $\boldsymbol{y}$ is generated as follows: $\boldsymbol{y}$ is first initialized to an empty vector before x_1 is enqueued. Then, for each input symbol x_t, the following three events may occur: 1. A random symbol $a \in \Sigma_4$ is inserted with probability p_I. In this case, $\boldsymbol{y}$ is concatenated with symbol a as $\boldsymbol{y} \leftarrow (\boldsymbol{y}, a)$ and x_t remains in the queue. 2. The symbol x_t is deleted with probability p_D, $\boldsymbol{y}$ remains unchanged, and x_{t+1} is enqueued. 3. The symbol x_t is transmitted with probability $p_\mathrm{T} = 1 - p_\mathrm{I} - p_\mathrm{D}$. In this case, the symbol is received correctly with probability $1 - p_\mathrm{S}$ or incorrectly with probability p_S, in which case the symbol is substituted by a symbol $a' \in \Sigma_4 \setminus \{x_t\}$ picked uniformly at random. The output is set to $\boldsymbol{y} \leftarrow (\boldsymbol{y}, x_t)$ and $\boldsymbol{y} \leftarrow (\boldsymbol{y}, a')$, respectively, and x_{t+1} is enqueued. After x_L leaves the queue, we obtain the output strand $\boldsymbol{y}$, of length L'. Note that L' is random and depends on the channel realization. Further, the symbols in $\boldsymbol{y}$ are not synchronized anymore. Hence, y_t could be the result of transmitting a symbol $x_{t'}$ with $t' \neq t$.

2.2 Multi-Draw IDS Channel

The data sequence is divided into M short DNA strands $\boldsymbol{x}_1, \dots, \boldsymbol{x}_M$, each of length L, which are synthesized and stored in a *pool*. We define the input list $\boldsymbol{X} \triangleq (\boldsymbol{x}_1, \dots, \boldsymbol{x}_M)$. We assume that the PCR amplification and sequencing processes generate $N = cM$ reads from the strands in the pool, resulting in a random number of (noisy) reads for each strand $\boldsymbol{x}_i$. The factor c is a positive real number referred to as the *coverage depth* in the literature.

The channel between the input to be synthesized, $\boldsymbol{X}$, and the output of the sequencing process, $\boldsymbol{Y}$, can be modeled in three phases as explained in the following.

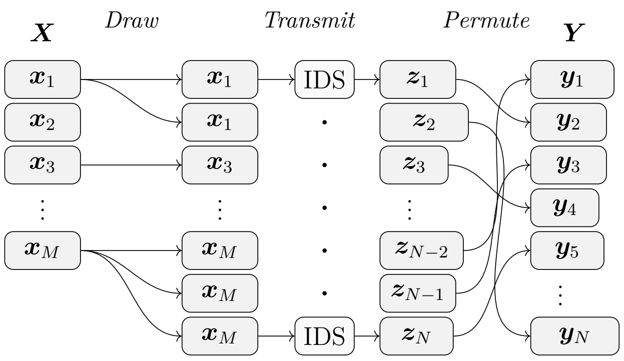


Figure IV.1: Multi-draw IDS channel.

1. *Draw:* N draws are performed from the list $\boldsymbol{X}$ uniformly at random with replacement. Let D_i be the random variable (RV) corresponding to the number of times strand $\boldsymbol{x}_i$ is drawn and define the random vector $\boldsymbol{D} \triangleq (D_1, \dots, D_M)$, with $\sum_i D_i = N$. Then, the random vector $\boldsymbol{D} = (D_1, \dots, D_M)$ is multinomially distributed as $\boldsymbol{D} \sim \mathsf{Multinom}(M, N, 1/M)$.

2. *Transmit:* The drawn strands are transmitted through independent IDS channels with identical parameters p_{I}, p_{D}, and p_{S}. We denote by $\boldsymbol{z}_j$ the output of the j-th IDS channel, $j \in \{1, \dots, N\}$, and define $\boldsymbol{Z} \triangleq (\boldsymbol{z}_1, \dots, \boldsymbol{z}_N)$.

3. *Permute:* The final output of the channel, denoted as the list $\boldsymbol{Y} \triangleq (\boldsymbol{y}_1, \dots, \boldsymbol{y}_N)$ is obtained by a permutation of $\boldsymbol{Z}$ chosen uniformly at random.

We illustrate the channel model in Fig. IV.1.[9] We define the channel parameter $\beta = \frac{\log_4(M)}{L}$ that relates the total number and the length of the strands and can be interpreted as the penalty the channel induces by the permutation effect.

3 Index-Based Concatenated Coding Scheme

We propose an index-based concatenated coding scheme consisting of three codes: an outer spatially-coupled low-density parity-check (SC-LDPC) code, providing overall protection to the data and against non-drawn strands, an index code, which counteracts the loss of ordering in the presence of IDS errors, and an inner code whose main goal is to maintain synchronization.

The information $\boldsymbol{u} \in \mathbb{F}_4^{k_\mathsf{o}}$ is encoded by an $[n_\mathsf{o}, k_\mathsf{o}, \frac{M}{2}, m]_4$ SC-LDPC code of output length n_o, input length k_o, coupling length $\frac{M}{2}$, and coupling memory m over the field $\mathbb{F}_4$ [20, 21]. Note that there is a one-to-one mapping of the

[9]In [3, 5, 6], the *draw* and *permute* steps are considered as a joint process referred to as *sampling.* We split the step for the sake of presentation; our channel model is essentially the same channel model, only that we consider the noise channel to be an IDS channel.

field $\mathbb{F}_4$ to the DNA alphabet $\Sigma_4 = \{\mathsf{A}, \mathsf{C}, \mathsf{G}, \mathsf{T}\}$. We interpret it as a field to allow linear operations over vectors in the DNA alphabet. The resulting codeword $\boldsymbol{w} = (w_1, \ldots, w_{n_\mathsf{o}}) \in \mathbb{F}_4^{n_\mathsf{o}}$ is split into M equal-length data blocks as $\boldsymbol{w} = (\boldsymbol{w}^{(1)}, \ldots, \boldsymbol{w}^{(M)})$ such that $\boldsymbol{w}^{(i)} = (w_1^{(i)}, \ldots, w_{L_\mathsf{o}}^{(i)}) \in \mathbb{F}_4^{L_\mathsf{o}}$, $L_\mathsf{o} = \frac{n_\mathsf{o}}{M}$. Each index i is encoded by an $[n_\mathsf{ix}, k_\mathsf{ix}]_4$ index code over $\mathbb{F}_4$ of even output length n_ix and even input length $k_\mathsf{ix} \geq \log_4(M)$ to an index codeword. Independently, the data block $\boldsymbol{w}^{(i)}$ is encoded by an $[n_\mathsf{i}, k_\mathsf{i}]_4$ inner code over $\mathbb{F}_4$ of output length n_i, input length k_i, and rate $R_\mathsf{i} = \frac{k_\mathsf{i}}{n_\mathsf{i}}$, generating an inner data codeword of length $\frac{L_\mathsf{o}}{R_\mathsf{i}}$. We consider the marker-repeat (MR) codes presented in [9, 22] for the inner code, where the k_i-th input symbol is repeated once such that $n_\mathsf{i} = k_\mathsf{i} + 1$ and $R_\mathsf{i} = 1 - \frac{1}{n_\mathsf{i}}$. Due to our decoding technique, we have higher error protection at the beginning and end of the DNA strands. Thus, we split the index codeword in half and insert it at the beginning and the end of the inner codeword of the respective data block. Finally, a random offset sequence is added to the generated strand resulting in the final encoded output strand $\boldsymbol{x}_i$, of length $L = L_\mathsf{o} R_\mathsf{i}^{-1} + n_\mathsf{ix}$. The random offset is known to the decoder and supports the synchronization capability of the index and inner decoder and ensures that the nucleotides of the stored DNA strands are uniformly distributed over Σ_4. The final codeword list is then described by $\boldsymbol{X} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_M)$. The overall code rate is $R = 2 \cdot R_\mathsf{o} \frac{L_\mathsf{o}}{L}$ in bits/nucleotide, with individual rates $R_\mathsf{o} = \frac{k_\mathsf{o}}{n_\mathsf{o}}$, $R_\mathsf{ix} = \frac{k_\mathsf{ix}}{n_\mathsf{ix}}$, and recall $R_\mathsf{i} = \frac{k_\mathsf{i}}{n_\mathsf{i}}$.

For the decoding procedure we introduce the following equivalent interpretation of the index and inner encoding as a joint process. We transform the integer i to a vector $\mathrm{ind}(i) \in \mathbb{F}_4^{k_\mathsf{ix}}$. This vector is split in half and placed at the beginning and end of the vector $\boldsymbol{w}^{(i)}$ resulting in the vector $\boldsymbol{v}^{(i)} = (v_1^{(i)}, \ldots, v_{k_\mathsf{ix}+L_\mathsf{o}}^{(i)})$, of length $k_\mathsf{ix} + L_\mathsf{o}$. Moreover, the $[n_\mathsf{ix}, k_\mathsf{ix}]_4$ index code is generated by a serial concatenation of two equal $[\frac{n_\mathsf{ix}}{2}, \frac{k_\mathsf{ix}}{2}]_4$ codes. Subsequently, the vector $\boldsymbol{v}^{(i)}$ is encoded at the index positions by the $[\frac{n_\mathsf{ix}}{2}, \frac{k_\mathsf{ix}}{2}]_4$ code and symbolwise by the inner MR code at the positions corresponding to $\boldsymbol{w}^{(i)}$. The coding/decoding scheme is depicted in Fig. IV.2.

4 Joint Index and Inner Decoding

The channel introduces two main impairments that need to be combated: the loss of ordering of the DNA strands due to the permutation effect, and the possibility that a strand is not drawn due to the drawing nature. However, the latter also provides inherent redundancy due to possible multiple copies of the same DNA strand that can be leveraged. Moreover, at the receiver side, the loss of synchronization within each strand due to insertion and deletion events is challenging. In general, we follow a *mismatched decoding* approach due to the decoder's channel uncertainty and complexity constraints. Instead of considering all DNA strands in the received list $\boldsymbol{Y}$ jointly to produce an estimate of the information, $\hat{\boldsymbol{u}}$, we propose the following sub-optimal decoding scheme.

First, we infer symbolwise APPs using a joint index and inner MAP decoder by means of the BCJR algorithm for each received strand $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_N$ independently. Optionally, clustering of the DNA strands using the APPs is performed. In

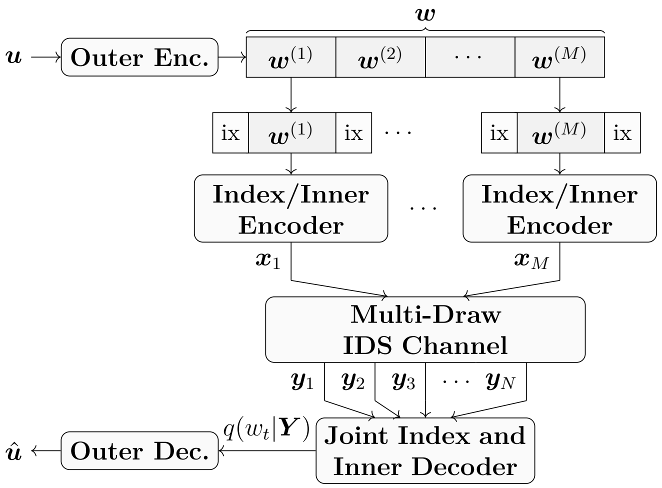


Figure IV.2: Concatenated coding scheme for the communication over the multi-draw IDS channel. The index is split into two parts and is inserted at the beginning and the end of the block $\boldsymbol{w}^{(i)}$. The term $q(w_t|\boldsymbol{Y})$ denotes the mismatched metric computed by the joint index-inner decoder.

that way the clustering makes use of the coding gain provided by the index and inner code. For combining the APPs within a cluster, the APPs are symbolwise multiplied, generating the mismatched APPs for one cluster. For a given cluster, by making hard-decisions on the index APPs, we estimate the block index $\hat{\imath}$ within $\boldsymbol{w}$. Therefore, assuming optimal clustering, we can actually benefit from the multi-copy gain of the channel for the index estimation. Once more, the APPs of the data block belonging to the same index $\hat{\imath}$ are multiplied, generating the mismatched APPs $q(w_t^{(i)}|\boldsymbol{Y})$. The overall ordered APPs $q(w_t|\boldsymbol{Y})$ are then passed to the outer decoder, which neglects any possible correlations of the symbolwise APPs. The outer code is specifically designed to handle *block-fading* effects, as described in Section 6. Combining APPs after inner and before outer decoding by symbolwise multiplication is inspired by the well performing *separate decoding* strategy of [10].

4.1 Inner MAP Decoding

We consider the case of decoding a single output strand $\boldsymbol{y}$ given an input $\boldsymbol{v}$ by means of the BCJR decoder over the joint code trellis of the index and inner code. We drop the superscript of the sequence $\boldsymbol{v}$ due to the permutation effect of the channel. We follow the concept introduced in [7], refined in [8]. For a detailed view, we refer to our prior work [10].

The IDS errors introduce a synchronization loss that destroys the Markov property of the joint index-inner code. However, we can form a hidden Markov model by introducing the drift d_t, defined as the number of insertions minus the number of deletions that occurred at time t, as a hidden state variable. The essence of the drift states is that the sequence $d_0, \ldots, d_{L_\mathsf{o}+k_\mathsf{ix}}$ forms a Markov chain and the outputs after time $t+1$ are independent of the previous states, restoring the Markov property. Hence, we can formulate forward and backward recursions for a BCJR decoder, whose final output APPs are $p(v_t|\boldsymbol{y}) = \frac{p(v_t,\boldsymbol{y})}{p(\boldsymbol{y})} \propto \sum_{d,d'} p(v_t, \boldsymbol{y}, d, d')$, where d and d' denote realizations of the RVs d_t and d_{t+1}, respectively.

4.2 Clustering

Common approaches perform clustering on the received strands before error-correction decoding using approximation techniques for the Levenshtein distance (minimum number of IDS operations), e.g., [16, 23]. Here, we present a clustering approach of complexity $\mathcal{O}(N^2 L)$ that leverages the coding gain of the index and inner code.

Given the APPs $p(v_t|\boldsymbol{y}_j)$, for each strand $\boldsymbol{y}_j$ a vector $\boldsymbol{r}^{(j)} = (r_1^{(j)}, \ldots, r_{L_\mathrm{B}}^{(j)}) \in \mathbb{R}^{L_\mathrm{B}}$ of binary log-likelihood ratios (LLRs) is obtained using marginalization, where $\mathbb{R}$ denotes the reals and L_B is the length of the corresponding binary vector of $\boldsymbol{v}$. The clustering decision is made by evaluating the pairwise Euclidean distance

$$d_\mathrm{E}^{(i,j)} = \sqrt{\sum_{\ell=1}^{L_\mathrm{B}} \left(r_\ell^{(i)} - r_\ell^{(j)}\right)^2}$$

for any two vectors $\boldsymbol{r}^{(i)}$ and $\boldsymbol{r}^{(j)}$, $i \neq j$. Let $D_\mathrm{E}^\mathrm{intra}$ be the RV of the Euclidean distance between two received sequences $\boldsymbol{r}^{(i)}$ and $\boldsymbol{r}^{(j)}$ originating from the same stored DNA strand. Assuming no correlations between the binary LLRs, $D_\mathrm{E}^\mathrm{intra}$ converges asymptotically in L_B to a normal distribution $\mathcal{N}(\mu_\mathrm{D}, \sigma_\mathrm{D}^2)$ with mean μ_D and variance σ_D^2 [24]. By sampling realizations of $D_\mathrm{E}^\mathrm{intra}$, it can be observed that this is a good approximation for $L_\mathrm{B} > 100$ for fitted parameters μ_D and σ_D^2. Consequently, given the computed pairwise distances $d_\mathrm{E}^{(i,j)}$ for $i, j \in \{1, \ldots, N\}$, $i \neq j$, strands are considered to stem from the same DNA strand if $d_\mathrm{E}^{(i,j)} \leq \mu_\mathrm{D} + \omega \cdot \sigma_\mathrm{D}$, where $\omega \in \mathbb{R}^+$ is a design parameter and $\mathbb{R}^+$ denotes the set of positive real numbers.

Let the k-th cluster, $1 \leq k \leq M'$, be formed as $\boldsymbol{C}_k \triangleq (\boldsymbol{y}_{k,1}, \ldots, \boldsymbol{y}_{k,|\boldsymbol{C}_k|})$, where $\boldsymbol{y}_{k,j}$ is the j-th strand placed in cluster $\boldsymbol{C}_k$ and $|\boldsymbol{C}_k| \neq 0$. Denote the overall clustering output as the list $\boldsymbol{C} = (\boldsymbol{C}_1, \ldots, \boldsymbol{C}_{M'})$, with $1 \leq M' \leq N$ and $\sum_{k=1}^{M'} |\boldsymbol{C}_k| = N$. For each cluster k, we multiply the respective APPs giving the mismatched rule

$$q(v_t|\boldsymbol{C}_k) \mathrel{\propto\!\!\!\!\!\sim} \prod_{j=1}^{|\boldsymbol{C}_k|} \frac{p(v_t|\boldsymbol{y}_{k,j})}{p(v_t)^{|\boldsymbol{C}_k|-1}} \,.$$

4.3 Index and Multiple DNA Strand Decoding

The index code and recovering the index information of each strand/cluster at the receiver side is the key point of tackling the permutation effect and leveraging the multi-copy gain of the channel. For each cluster $\boldsymbol{C}_k$, we obtain the soft information of the indices by extracting

$$\boldsymbol{q}(\mathrm{ind}(i)|\boldsymbol{C}_k) \triangleq \left(q(v_1|\boldsymbol{C}_k), \ldots, q(v_{k_\mathsf{ix}/2}|\boldsymbol{C}_k), q(v_{L_\mathsf{o}+k_\mathsf{ix}/2+1}|\boldsymbol{C}_k), \ldots, q(_{L_\mathsf{o}+k_\mathsf{ix}}|\boldsymbol{C}_k)\right) \,.$$

We perform hard-decisions on $\boldsymbol{q}(\mathrm{ind}(i)|\boldsymbol{C}_k)$ to compute an estimate index $\hat{i}_k$ for every cluster $1 \leq k \leq M'$, which will determine how the strand/cluster is grouped. Let $\mathcal{S}_i$ be the set of clusters with decision on index i, where $0 \leq |\mathcal{S}_i| \leq M'$. For all data positions of the i-th block, i.e., for symbols of $\boldsymbol{w}^{(i)}$, we compute the APPs according to the mismatched rule

$$q(w_t^{(i)}|\boldsymbol{Y}) \propto \prod_{k \in \mathcal{S}_i} \frac{q(w_t^{(i)}|\boldsymbol{C}_k)}{q(w_t^{(i)})^{|\mathcal{S}_i|-1}},$$

where $q(w_t^{(i)}|\boldsymbol{Y}) = \frac{1}{4}$ when $\mathcal{S}_i = \emptyset$.

As a final step, according to the position of block $\boldsymbol{w}^{(i)}$ in the sequence $\boldsymbol{w}$, the mismatched APPs are given to the outer decoder, which outputs an estimate $\hat{\boldsymbol{u}}$.

5 Achievable Information Rates

We compute AIRs of mismatched decoders for fixed index and inner codes. Specifically, we compute i.u.d. *BCJR-once* rates ($R_{\text{BCJR-once}}$), measured in bits/nucleotide, which are defined as the symbolwise mutual information between the input and its corresponding LLRs with uniform input distribution [25–27]. The BCJR-once rates serve as an appropriate measure of an AIR for an outer decoder that ignores possible correlations between the symbolwise estimates $q(w_t|\boldsymbol{Y})$ given by the inner MAP decoder. Moreover, by using a mismatched decoding metric $q(\boldsymbol{w}|\boldsymbol{Y})$ instead of the true metric $p(\boldsymbol{w}|\boldsymbol{Y})$, the rate $R_{\text{BCJR-once}}$ only decreases. Consider the mismatched-decoder decoding metric

$$q(\boldsymbol{w}|\boldsymbol{Y}) = \prod_{i=1}^{M} \prod_{t=1}^{L_\mathrm{o}} q(w_t^{(i)}|\boldsymbol{Y}).$$

The mutual information $\mathrm{I}(\boldsymbol{w};\boldsymbol{Y})$ can be bounded from below as (for clarity of presentation, we do not distinguish between RVs and their realizations in our notation)

$$\mathrm{I}(\boldsymbol{w};\boldsymbol{Y}) \geq \mathbb{E}_{\boldsymbol{D}}\left[\sum_{i,t} \mathbb{E}_{w_t^{(i)},\boldsymbol{Y}|\boldsymbol{D}}\left[\log_2 \frac{q(w_t^{(i)}|\boldsymbol{Y})}{p(w_t^{(i)})}\right]\right],$$

where $\mathbb{E}_X[\cdot]$ denotes expectation with respect to the RV X.

We define the LLR representation

$$\mathrm{LLR}_{i,t}(a) = \sum_{k \in \mathcal{S}_i} \sum_{j=1}^{|\boldsymbol{C}_k|} \ln \frac{q(w_t^{(i)} = a|\boldsymbol{y}_{k,j})}{q(w_t^{(i)} = 0|\boldsymbol{y}_{k,j})},$$

where $\boldsymbol{y}_{k,j}$ denotes the j-th strand of $\boldsymbol{C}_k$. Similar to [25], we use the mismatched LLR representation to formally define

$$R_{\text{BCJR-once}} \triangleq \mathbb{E}_{\boldsymbol{D}}\left[\lim_{ML\to\infty} \frac{1}{ML} \sum_{i=1}^{M} \sum_{t=1}^{L_\mathrm{o}} \mathrm{I}\left(w_t^{(i)}; \mathrm{LLR}_{i,t}(w_t^{(i)})\right)\right],$$

which depends on the multi-draw parameters β and c, IDS channel parameters p_{I}, p_{D}, and p_{S}, and is evaluated for a fixed index and inner coding/decoding scheme. Assuming ergodicity for a single strand, we follow a simulation-based approach to calculate $R_{\text{BCJR-once}}$. We simulate for long strand lengths $\widetilde{L} = 100\,000$ (correspondingly $\widetilde{L_{\mathsf{o}}} = (\widetilde{L} - n_{\mathsf{ix}})R_{\mathsf{i}}$ and $\widetilde{M} = 4^{\beta\widetilde{L}}$), reflecting $ML \to \infty$ for a fixed β, and approximate $\mathbb{E}_{\boldsymbol{D}}[\cdot]$ by the Monte-Carlo method. Hence, by averaging over a large number of draw realizations, Φ, we approximate $R_{\text{BCJR-once}}$ as

$$R_{\text{BCJR-once}} \approx \frac{1}{\Phi}\sum_{\phi=1}^{\Phi}\frac{1}{\widetilde{M}\widetilde{L}}\sum_{i=1}^{\widetilde{M}}\sum_{t=1}^{\widetilde{L_{\mathsf{o}}}}\left(\log_2 4 + \log_2 \frac{\mathrm{e}^{\mathrm{LLR}_{i,t}(w_t^{(i)})}}{\sum_{a\in\mathbb{F}_4}\mathrm{e}^{\mathrm{LLR}_{i,t}(a)}}\right) .$$

6 Outer Code

Due to its drawing effect, the DNA storage channel as seen by the outer code resembles a block-fading channel when considering a finite number of strands M [5]. Hence, it also shares its afflictions, most importantly its non-ergodic property. In particular, an *outage* event occurs when not enough strands are drawn for a specific block. Formally, an outage event occurs when the instantaneous mutual information between the input and output of the channel is lower than the transmission rate R_{o}. We consider the BCJR-once version of the information-outage probability adapted from [28], to which we refer as the *mismatched information-outage probability* q_{out}. It incorporates our mismatched decoding approach, the fixed i.u.d. input, and the dispersion due to the finite blocklength phenomena. Let $r^{\boldsymbol{d}}_{\text{BCJR-once}}$ denote the instantaneous BCJR-once information density for the finite length regime and fixed draw realization $\boldsymbol{d}$, computed as

$$r^{\boldsymbol{d}}_{\text{BCJR-once}} = \frac{1}{ML}\sum_{i=1}^{M}\sum_{t=1}^{L_{\mathsf{o}}}\mathrm{I}\left(w_t^{(i)};\mathrm{LLR}_{i,t}(w_t^{(i)})\right) .$$

Then, q_{out} is formally defined as

$$q_{\text{out}} = \Pr\left(r^{\boldsymbol{d}}_{\text{BCJR-once}} < R'_{\mathrm{o}}\right) \geq p_{\text{out}} ,$$

where $R'_{\mathrm{o}} = 2R_{\mathrm{o}}$ such that the outer code rate is measured in bits/nucleotide and p_{out} is the true outage probability. q_{out} gives a lower bound on the FER for a given encoder and mismatched decoder pair, i.e., using the mismatched metric described Section 5, for fixed finite number of strands, fixed finite blocklength, and fixed channel parameters. We approximate q_{out} by the Monte-Carlo method for a given R'_{o}.

The diversity order of the outer code is an important parameter that influences the slope of the FER curve for low IDS probabilities. By coding over blocks of $\boldsymbol{w}$, the outer code can provide a *diversity* gain. We consider protograph-based SC-LDPC codes for the outer code, as they achieve high diversity for the block-fading channel [29]. We optimize the protograph following the procedure in [29] based on the density evolution outage (DEO) probability bound. The optimization works by first finding a block LDPC protograph with DEO close to a target $q_{\text{out}} = 10^{-3}$ based on the joint index-inner code. Then, an optimization over the edge spreading

Table IV.1: Index Codes

Code	Codebook
$[3,1]_4$	$(0,3,3)$, $(1,0,2)$, $(2,1,1)$, $(3,2,0)$
$[6,2]_4$	$(0,0,1,0,1,1), (0,0,3,3,2,2), (0,2,2,2,2,0), (1,1,1,0,2,3),$ $(1,1,2,2,0,1), (1,1,3,3,3,0), (1,3,0,3,1,1), (2,0,1,1,0,0),$ $(2,0,3,2,3,3), (2,2,2,0,3,2), (2,2,3,1,1,1), (2,3,0,2,2,2),$ $(3,1,0,1,3,3), (3,1,1,2,1,2), (3,3,2,1,1,0), (3,3,3,0,0,3)$

to create the SC-LDPC protograph is performed. In order to design a non-binary code, random edge weights from $\mathbb{F}_4$ are assigned to the edges of the protograph throughout the optimization. In the optimization, we fix the coupling length to $\frac{M}{2}$ and the coupling memory to $m = 2$.

7 AIR and FER Results

We analyze our proposed coding scheme for the multi-draw IDS channel by means of AIRs, information-outage probability for the outer code, and FER simulations for different coverage depths c and fixed β. We set the IDS channel parameters to $p_\mathrm{I} = 0.017$, $p_\mathrm{D} = 0.020$, and $p_\mathrm{S} = 0.022$, motivated by the results of the experiment from [9]. Additionally, we present FER results on their experimental data. Moreover, we fix the inner code rate to $R_\mathrm{i} = \frac{10}{11}$ and perform full-window belief propagation decoding with a maximum of 100 iterations for the SC-LDPC outer code. The index codes $[3,1]_4$ (AIR simulations) and $[6,2]_4$ (information-outage/FER simulations), both with $R_\mathrm{ix} = \frac{1}{3}$, are obtained via an exhaustive graph search algorithm optimizing the code's Levenshtein distance spectrum (see Table IV.1) [10, 30].

Fig. IV.3 shows the AIR $R_\text{BCJR-once}$ of different coding/decoding schemes for $\beta = 2 \cdot 10^{-5}$. We observe the unavoidable rate loss due to the non-drawing strand and permutation effect for low coverage depth c which, however, diminishes for increasing values of c due to the multi-copy gain. For an overall benchmark, we include the AIR of an optimal index-based scheme over the noiseless channel (i.e., only drawing and permuting effects). With IDS noise, we include a benchmark for an index-based coding approach given an index genie in the decoding process, i.e., we artificially exclude the permutation loss of the channel. The rate gap to the noiseless scenario can be explained by the IDS noise and is also due to our chosen sub-optimal coding and mismatched decoding approach. Moreover, protecting the index with a strong code seems crucial since in the non-coded case received strands may be grouped incorrectly. For the given parameters, the designed index code performs very close to the genie ordering curve, while the non-coded index curve suffers from a big rate loss. Further, applying our clustering method in combination with a coded index attains the index genie benchmark since the clustering enhances the quality of the index decisions.

Fig. IV.4 shows the information-outage probability q_out versus the outer code

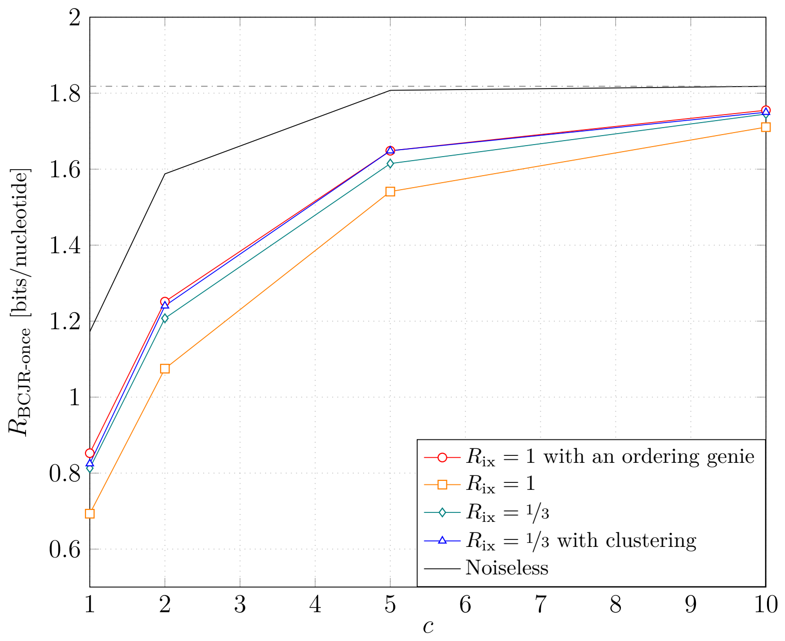


Figure IV.3: BCJR-once rates versus coverage depth c for different index coding rates R_{ix} and decoding techniques. Fixed parameters are $\beta = 2 \cdot 10^{-5}$, $p_{\text{I}} = 0.017$, $p_{\text{D}} = 0.020$, $p_{\text{S}} = 0.022$, and $R_{\text{i}} = \frac{10}{11}$. For clustering, we determined $D_{\text{E}}^{\text{intra}} \sim \mathcal{N}(938, 6.4^2)$ and use $\omega = 5$. The dash-dotted gray line represents the rate limit due to the inner code rate.

rate $R'_{\text{o}} = 2R_{\text{o}}$ for different coverage depths c and decoding approaches for $M = 256$ input strands and fixed index code of rate $R_{\text{ix}} = \frac{1}{3}$. For a fixed R'_{o}, the corresponding q_{out} serves as a lower bound for any code's FER with that rate and length, and a decoder following our mismatched decoding rule. In general, we see a similar behavior of our proposed coding/decoding schemes as for the AIRs. In the same figure, we show the FER performance of a protograph-based SC-LDPC outer code (stand-alone solid shaped markers). The SC-LDPC protograph is optimized individually for each considered coding/decoding scheme and then randomly lifted and assigned random edge weights from $\mathbb{F}_4$. We observe an expected rate loss that can be explained by the limited protograph search space. The corresponding optimized (block) LDPC photographs are shown in Table IV.2. Finally, we also include FER results on experimental data from [9] (stand-alone empty shaped markers). Notably, the penalty in performance, due to the fact that the experimental channel may suffer from other additional noise impairments, is small. In addition, the SC-LDPC protographs are optimized on synthetic samples and not on experimental data which also contributes to the loss in performance.

8 Conclusion

We proposed a practical index-based concatenated coding scheme for the multi-draw IDS channel, a model approximating the DNA storage process. Further, we

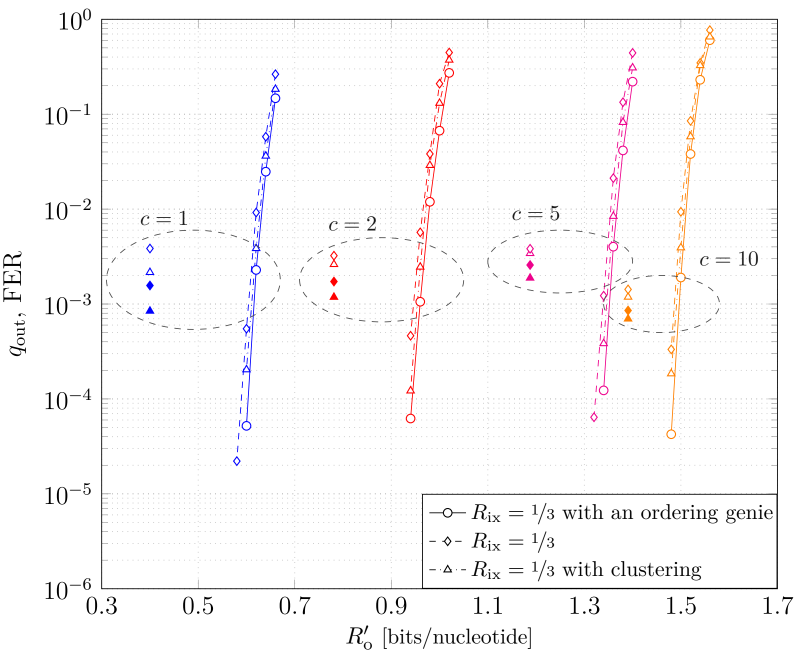


Figure IV.4: Information-outage probability q_{out} and FER versus outer code rate $R'_{\text{o}} = 2R_{\text{o}}$, for fixed $N_{\text{o}} = 23040$, $M = 256$, $L = 110$, $\beta \approx 0.36$, $p_{\text{I}} = 0.017$, $p_{\text{D}} = 0.020$, $p_{\text{S}} = 0.022$, and $R_{\text{i}} = 0.9183 \approx \frac{10}{11}$. For clustering, we determined $D_{\text{E}}^{\text{intra}} \sim \mathcal{N}(31.52, 5.44^2)$ and use $\omega = 2.5$. The different curves represent q_{out} for different coding/decoding schemes. The stand-alone markers correspond to the FER performance when using an optimized SC-LDPC outer code on synthetic (solid shaped) and experimental (empty shaped) data.

Table IV.2: Block LDPC Photographs for Different Values of c

c	Protograph[a]		R'_{o}
	Coded-index	Coded-index + clustering	
1	$\begin{pmatrix} 1&1&1&0&1 \\ 1&1&1&1&0 \\ 1&0&1&0&1 \\ 0&1&0&2&0 \end{pmatrix}$	$\begin{pmatrix} 1&1&1&0&1 \\ 0&0&2&2&0 \\ 2&0&0&0&1 \\ 0&2&0&1&0 \end{pmatrix}$	0.3750
2	$\begin{pmatrix} 1&2&0&0&1 \\ 0&0&1&1&1 \\ 2&1&2&1&0 \end{pmatrix}$	$\begin{pmatrix} 1&1&0&1&0 \\ 1&0&2&1&1 \\ 1&2&1&0&1 \end{pmatrix}$	0.7812
5	$\begin{pmatrix} 2&1&0&2&2 \\ 0&2&2&1&1 \end{pmatrix}$	$\begin{pmatrix} 2&1&0&2&2 \\ 0&2&2&1&1 \end{pmatrix}$	1.1876
10	$\begin{pmatrix} 1&2&1&1&0&0&0&1&1&2 \\ 1&0&2&0&2&1&2&0&1&0 \\ 0&1&0&1&1&2&0&2&1&1 \end{pmatrix}$	$\begin{pmatrix} 1&1&1&1&0&0&0&1&2&1 \\ 1&1&0&0&2&1&2&2&0&1 \\ 0&1&2&1&1&2&0&0&1&1 \end{pmatrix}$	1.3906

[a]Optimized edge spreading is performed on the given protographs, with a coupling length of 128 and a coupling memory of $m = 2$. Recall $R'_{\text{o}} = 2R_{\text{o}}$.

presented low-complexity decoding techniques for this setup. Our AIR results show that, in the presence of IDS errors, protecting the index with a strong code is crucial. Finally, we proposed explicit code constructions with FER performance of around 10^{-3} for synthetic and experimental data and 256 input strands.

References

[1] R. Heckel, G. Mikutis, and R. N. Grass, "A characterization of the DNA data storage channel," *Sci. Rep.*, vol. 9, no. 9663, pp. 1–12, Jul. 2019.

[2] R. Heckel, I. Shomorony, K. Ramchandran, and D. N. C. Tse, "Fundamental limits of DNA storage systems," in *Proc. IEEE Int. Symp. Inf. Theory*, Aachen, Germany, Jun. 2017.

[3] A. Lenz, P. H. Siegel, A. Wachter-Zeh, and E. Yaakobi, "An upper bound on the capacity of the DNA storage channel," in *Proc. IEEE Inf. Theory Workshop*, Visby, Sweden, Aug. 2019.

[4] ——, "Achieving the capacity of the DNA storage channel," in *Proc. IEEE Int. Conf. Acoust., Speech, Sig. Process.*, Barcelona, Spain, May 2020.

[5] N. Weinberger and N. Merhav, "The DNA storage channel: Capacity and error probability bounds," *IEEE Trans. Inf. Theory*, vol. 68, no. 9, pp. 5657–5700, Sep. 2022.

[6] I. Shomorony and R. Heckel, "Information-theoretic foundations of DNA data storage," *Found. Trends Commun. Inf. Theory*, vol. 19, no. 1, pp. 1–106, Feb. 2022.

[7] M. C. Davey and D. J. C. MacKay, "Reliable communication over channels with insertions, deletions, and substitutions," *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 687–698, Feb. 2001.

[8] J. A. Briffa, H. G. Schaathun, and S. Wesemeyer, "An improved decoding algorithm for the Davey-MacKay construction," in *Proc. IEEE Int. Conf. Commun.*, Cape Town, South Africa, May 2010.

[9] S. R. Srinivasavaradhan, S. Gopi, H. D. Pfister, and S. Yekhanin, "Trellis BMA: Coded trace reconstruction on IDS channels for DNA storage," in *Proc. IEEE Int. Symp. Inf. Theory*, Melbourne, Australia, Jul. 2021.

[10] I. Maarouf, A. Lenz, L. Welter, A. Wachter-Zeh, E. Rosnes, and A. Graell i Amat, "Concatenated codes for multiple reads of a DNA sequence," *IEEE Trans. Inf. Theory*, vol. 69, no. 2, pp. 910–927, Feb. 2023.

[11] G. M. Church, Y. Gao, and S. Kosuri, "Next-generation digital information storage in DNA," *Sci.*, vol. 337, no. 6102, p. 1628, Aug. 2012.

[12] N. Goldman, P. Bertone, S. Chen, C. Dessimoz, E. M. LeProust, B. Sipos, and E. Birney, "Towards practical, high-capacity, low-maintenance information storage in synthesized DNA," *Nature*, vol. 494, no. 7435, pp. 77–80, Jan. 2013.

[13] R. N. Grass, R. Heckel, M. Puddu, D. Paunescu, and W. J. Stark, "Robust chemical preservation of digital information on DNA in silica with error-correcting codes," *Angew. Chem. Int. Ed.*, vol. 54, no. 8, pp. 2552–2555, Feb. 2015.

[14] S. M. H. T. Yazdi, Y. Yuan, J. Ma, H. Zhao, and O. Milenkovic, "A rewritable, random-access DNA-based storage system," *Sci. Rep.*, vol. 5, no. 14138, pp. 1–10, Sep. 2015.

[15] L. Organick, S. D. Ang, Y.-J. Chen, R. Lopez, S. Yekhanin, K. Makarychev, M. Z. Racz, G. Kamath, P. Gopalan, B. Nguyen, C. N. Takahashi, S. Newman, H.-Y. Parker, C. Rashtchian, K. Stewart, G. Gupta, R. Carlson, J. Mulligan, D. Carmean, G. Seelig, L. Ceze, and K. Strauss, "Random access in large-scale DNA data storage," *Nature Biotechnol.*, vol. 36, no. 3, pp. 242–248, Mar. 2018.

[16] P. L. Antkowiak, J. Lietard, M. Z. Darestani, M. M. Somoza, W. J. Stark, R. Heckel, and R. N. Grass, "Low cost DNA data storage using photolithographic synthesis and advanced information reconstruction and error correction," *Nature Commun.*, vol. 11, no. 5345, pp. 1–10, Oct. 2020.

[17] A. Lenz, P. H. Siegel, A. Wachter-Zeh, and E. Yaakobi, "Coding over sets for DNA storage," *IEEE Trans. Inf. Theory*, vol. 66, no. 4, pp. 2331–2351, Apr. 2020.

[18] A. Lenz, L. Welter, and S. Puchinger, "Achievable rates of concatenated codes in DNA storage under substitution errors," in *Proc. IEEE Int. Symp. Inf. Theory Appl.*, Kapolei, USA, Oct. 2020.

[19] N. Weinberger, "Error probability bounds for coded-index DNA storage systems," *IEEE Trans. Inf. Theory*, vol. 68, no. 11, pp. 7005–7022, Nov. 2022.

[20] A. J. Felström and K. S. Zigangirov, "Time-varying periodic convolutional codes with low-density parity-check matrix," *IEEE Trans. Inf. Theory*, vol. 45, no. 6, pp. 2181–2191, Sep. 1999.

[21] M. Lentmaier, A. Sridharan, D. J. Costello, Jr., and K. S. Zigangirov, "Iterative decoding threshold analysis for LDPC convolutional codes," *IEEE Trans. Inf. Theory*, vol. 56, no. 10, pp. 5274–5289, Oct. 2010.

[22] M. Inoue and H. Kaneko, "Adaptive synchronization marker for insertion/deletion/substitution error correction," in *Proc. IEEE Int. Symp. Inf. Theory*, Cambridge, USA, Jul. 2012.

[23] C. Rashtchian, K. Makarychev, M. Rácz, S. D. Ang, D. Jevdjic, S. Yekhanin, L. Ceze, and K. Strauss, "Clustering billions of reads for DNA data storage," in *Proc. Adv. Neural Inf. Process. Syst.*, Long Beach, USA, Dec. 2017.

[24] B. A. Dawkins, T. T. Le, and B. A. McKinney, "Theoretical properties of distance distributions and novel metrics for nearest-neighbor feature selection," *PLOS ONE*, vol. 16, no. 2, pp. 1–67, Feb. 2021.

[25] A. Kavčić, X. Ma, and M. Mitzenmacher, "Binary intersymbol interference channels: Gallager codes, density evolution, and code performance bounds," *IEEE Trans. Inf. Theory*, vol. 49, no. 7, pp. 1636–1652, Jul. 2003.

[26] R. R. Müller and W. H. Gerstacker, “On the capacity loss due to separation of detection and decoding,” *IEEE Trans. Inf. Theory*, vol. 50, no. 8, pp. 1769–1778, Aug. 2004.

[27] J. B. Soriaga, H. D. Pfister, and P. H. Siegel, “Determining and approaching achievable rates of binary intersymbol interference channels using multistage decoding,” *IEEE Trans. Inf. Theory*, vol. 53, no. 4, pp. 1416–1429, Apr. 2007.

[28] D. Buckingham and M. C. Valenti, “The information-outage probability of finite-length codes over AWGN channels,” in *Proc. 42nd Annu. Conf. Inf. Sci. Syst.*, Princeton, USA, Mar. 2008.

[29] N. ul Hassan, M. Lentmaier, I. Andriyanova, and G. P. Fettweis, “Improving code diversity on block-fading channels by spatial coupling,” in *Proc. IEEE Int. Symp. Inf. Theory*, Honolulu, USA, Jun./Jul. 2014.

[30] E. C. Sewell, “A branch and bound algorithm for the stability number of a sparse graph,” *INFORMS J. Comput.*, vol. 10, no. 4, pp. 438–447, Nov. 1998.

Graphic design: Communication Division, UiB / Print: Skipnes Kommunikasjon AS

uib.no

ISBN: 9788230862230 (print)
9788230856970 (PDF)