Information-Theoretic Foundations of DNA Data Storage

Ilan Shomorony[1] and Reinhard Heckel[2]

[1] *University of Illinois at Urbana-Champaign, USA; ilans@illinois.edu*
[2] *Technical University of Munich, Germany, and Rice University, USA; reinhard.heckel@tum.de*

ABSTRACT

Due to its longevity and enormous information density, DNA is an attractive medium for archival data storage. Natural DNA more than 700.000 years old has been recovered, and about 5 grams of DNA can in principle hold a Zetabyte of digital information, orders of magnitude more than what is achieved on conventional storage media. Thanks to rapid technological advances, DNA storage is becoming practically feasible, as demonstrated by a number of experimental storage systems, making it a promising solution for our society's increasing need of data storage.

While in living things, DNA molecules can consist of millions of nucleotides, due to technological constraints, in practice, data is stored on many short DNA molecules, which are preserved in a DNA pool and cannot be spatially ordered. Moreover, imperfections in sequencing, synthesis, and handling, as well as DNA decay during storage, introduce random noise into the system, making the task of reliably storing and retrieving information in DNA challenging.

This unique setup raises a natural information-theoretic question: how much information can be reliably stored on

Ilan Shomorony and Reinhard Heckel (2022), "Information-Theoretic Foundations of DNA Data Storage", Foundations and Trends® in Communications and Information Theory: Vol. 19, No. 1, pp 1–106. DOI: 10.1561/0100000117.

and reconstructed from millions of short noisy sequences? The goal of this monograph is to address this question by discussing the fundamental limits of storing information on DNA. Motivated by current technological constraints on DNA synthesis and sequencing, we propose a probabilistic channel model that captures three key distinctive aspects of the DNA storage systems: (1) the data is written onto many short DNA molecules that are stored in an unordered fashion; (2) the molecules are corrupted by noise and (3) the data is read by randomly sampling from the DNA pool. Our goal is to investigate the impact of each of these key aspects on the capacity of the DNA storage system. Rather than focusing on coding-theoretic considerations and computationally efficient encoding and decoding, we aim to build an information-theoretic foundation for the analysis of these channels, developing tools for achievability and converse arguments.

1

Introduction

In recent years, the number of applications that require or are enabled by digital data storage and the amount of data generated by a variety of technologies have increased dramatically. This has spurred significant interest in new storage technologies beyond hard drives, magnetic tapes, and memory chips. In this context, DNA—the molecule that carries the genetic instructions of all living organisms—emerged as a promising storage medium. DNA has two key advantages over conventional digital storage technologies: extreme longevity and information density. This makes DNA an interesting storage medium, particularly for archival storage.

Data on DNA can last very long, if stored appropriately, as nature itself proves. As demonstrated by recently sequenced DNA extracted from a mammoth tooth found in the Siberian permafrost [113], the information in a DNA molecule can be preserved for more than a million years. In contrast, information on memory chips lasts no more than a few years, and data on hard drives and magnetic tapes lasts no more than a few decades. While conventional storage media could be redesigned to preserve data longer, the longevity of DNA is currently unmatched, as illustrated in Figure 1.1(a).

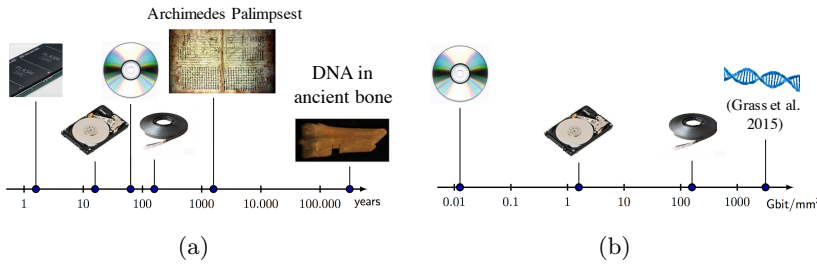

Figure 1.1: (a) Longevity of different data storage media. Archimedes Palimpsest containing "The Methods of Mechanical Theorems" survived more than 1000 years. Recently, 700,000 year old DNA from ancient horse bones has been successfully sequenced [91]. (b) Information density of different storage media. The proof-of-concept DNA-based storage system of [39] achieved an information density that is over one order of magnitude higher that of magnetic tape.

The information density of DNA is also extremely large. Just 5 grams of DNA contain about $4 \cdot 10^{21}$ nucleotides, which in principle could hold $8 \cdot 10^{21}$ bits, or one zettabyte. In a practical system, the redundancy required for error-correction coding required to build a reliable system reduces these numbers, but we can achieve information densities orders of magnitude larger than the highest information densities achieved on hard drives and tapes, as shown in Figure 1.1(b).

1.1 A brief history of DNA data storage

Computer scientists and engineers have dreamed of harnessing DNA's storage capabilities already in the 60s [9], [79], and in recent years DNA data storage, or more broadly, molecular information storage, developed into an active field of research. In 2012 and 2013 groups lead by Church [27] and Goldman [38] independently stored about a megabyte of data in DNA. Later, Grass, Heckel, Puddu, *et al.* [39] demonstrated that millennia-long storage times are possible by protecting the data both physically and information-theoretically, and designing a robust DNA data storage scheme using error-correcting codes. Yazdi, Yuan, Ma, *et al.* [119] showed how to selectively access files, and Erlich and Zielinski [33] demonstrated that a DNA storage system can achieve very high information densities, close to the absolute maximum of two bits

per nucleotide. In 2018, Organick, Ang, Chen, *et al.* [80] scaled up these techniques and stored about 200 megabytes of data. Together, these and other works demonstrated that writing, storing, and retrieving data using DNA as a medium is possible with today's technology, and achieves information densities and information lifetimes that are far beyond what state-of-the-art tapes and discs achieve.

DNA is a long molecule made up of four nucleotides (Adenine, Cytosine, Guanine, and Thymine) and, for storage purposes, can be viewed as a string over a four-letter alphabet. However, there are hard technological constraints for writing on DNA and for reading DNA, which need to be considered in the design of a practical DNA-based storage system. While in a living cell a DNA molecule may consist of millions of bases (the human chromosome 1, for example, is 250 million bases long), due to practical technological constraints, it is difficult and inefficient to synthesize long strands of DNA. For that reason, all recent works that have demonstrated working DNA storage systems stored information on molecules of no longer than 100-200 nucleotides [27], [33], [38], [39], [80], [119].

The process of determining the order of nucleotides in a DNA molecule, or DNA sequencing, suffers from similar length constraints. State-of-the-art sequencing platforms such as Illumina cannot sequence DNA segments longer than a few hundred nucleotides. While recently developed, so-called third-generation technologies such as Pacific Biosciences and Oxford Nanopore can provide reads that are several thousand bases long, their error rates and reading costs are significantly higher [30], [117].

Due to those constraints in the length of the DNA molecules that can be synthesized, stored and sequenced, a practical DNA-based storage system consists of many *short* DNA molecules stored in an *unordered fashion* in a solution.

Another technological limitation to DNA-based storage comes from the fact that state-of-the-art sequencing technologies rely on the *shotgun* sequencing paradigm. This corresponds to (randomly) sampling and reading sequences from the DNA pool. Furthermore, sequencing is usually preceded by several cycles of Polymerase Chain Reaction (PCR) amplification. In each cycle, the amount of each DNA molecule is

Table 1.1: Parameters of a few DNA storage systems using array-based synthesis, listed chronologically.

	length of seqs.	number of seqs	data stored (in MB)	error correction
Church, Gao, and Kosuri [27]	115	54,898	0.65	None
Goldman, Bertone, Chen, *et al.* [38]	117	153,335	0.75	Repetition
Grass, Heckel, Puddu, *et al.* [39]	117	4,991	0.08	RS
Blawat, Gaedke, Hütter, *et al.* [12]	190	900,000	22	RS
Bornholt, Lopez, Carmean, *et al.* [14]	120	45,652	0.15	RS
Erlich and Zielinski [33]	152	72,000	2.14	Fountain
Organick, Ang, Chen, *et al.* [80]	150	$13.4 \cdot 10^9$	200.2	RS
Chandak, Tatwawadi, Lau, *et al.* [17]	150	13,716	0.192	LDPC
Heckel and Grass [46]	105	$3.88 \cdot 10^9$	63.1	RS
Antkowiak, Lietard, Darestani, *et al.* [7]	60	16,383	0.1	RS

amplified by a factor between 1.6 and 1.8, and this factor can be sequence-dependent, thus leading to very different concentration of distinct DNA molecules. Last but not least, the DNA molecules in a DNA-based storage system are subject to errors such as insertions, deletions, and substitutions of nucleotides at the time of synthesis, during the storage period, and during sequencing.

1.2 Overview of existing DNA storage systems

In the last decade, several research groups have shown that using today's technologies, it is possible to store on the order of megabytes of data reliably. All systems that stored megabytes of data and demonstrated correct recovery relied on array-based synthesis, where data is stored on a set of many short sequences. In Table 1.1, we list several of these implementations with some of their key parameters in chronological order. All systems listed in the table stored a unique index on each sequence to deal with the shuffling character of the channel at decoding, and starting from Grass, Heckel, Puddu, *et al.* [39], all systems used outer error-correcting codes to deal with the loss of individual sequences.

There are alternatives to DNA data storage based on array-based synthesis. For example, Yazdi, Yuan, Ma, *et al.* [119] used column-based synthesis, where individual sequences are generated one-by-one. With this approach, the authors successfully stored 0.017 MB on 32

sequences of length 1000 each. Another example is the implementation by Lee, Kalhor, Goela, *et al.* [63], which stored 18 bytes using enzymatic synthesis. Tabatabaei, Wang, Athreya, *et al.* [109] avoided synthesis altogether, and stored data in form of nicks at certain positions on the backbone of existing DNA. Yet another example is the work by Yim, McBee, Song, *et al.* [121], which stored data (about 72 bits) in living cells via CRISPR arrays. All these approaches are conceptually interesting alternatives to array-based synthesis, but scaling them to store more than a few bytes of data currently looks challenging.

Array-based synthesis generates many copies of each sequence, by building up each sequence on a different spot of the array. It is also possible to modify array-based synthesis to grow sequences at one spot of the array so that a predefined fraction of the sequences contain, say, nucleotide A but others contain, say, nucleotide G at a given position. Anavy, Vaknin, Atar, *et al.* [5] explored this idea and proposed the notion of *composite DNA letters* to reduce the number of synthesis cycles. However, the use of composite letters comes at a higher sequencing cost to guarantee that composite letters can be identified from the sequenced DNA molecules, and also comes at a higher decoding complexity.

While the proof-of-concept implementations listed in Table 1.1 demonstrate that DNA storage can be practical, their overall cost is still an obstacle for them to become practically viable. For example, using the architecture proposed in Erlich and Zielinski [33], the estimated cost of synthesizing 1GB of data was $3.27 million [33, Supplementary Material], and the current cost of storing a Megabyte of DNA is around $500 [7]. However, until DNA storage becomes viable for commercial archival storage applications, there are already applications of DNA storage that are not possible with other storage media. For example, Koch, Gantenbein, Masania, *et al.* [54] demonstrated that data stored in DNA can be embedded into any product made of plastic, providing information about the product inside the product.

1.3 Information Theory and DNA storage

DNA-based storage is a fundamentally new way of storing data, due to the way DNA is written, stored, and read. The technology is still

under development, and details such as error profiles, the exact length of synthesized DNA molecules, and the sequencing throughput are likely to change. An information-theoretic perspective and an understanding of the fundamental limits of DNA storage will enable a system design based on key conceptual insights and tradeoffs.

Joint design of physical system and coding schemes: Conceptual advances in terms of how to optimally code for this new storage paradigm can inform biochemists in their development of new synthesis and sequencing technologies for building DNA storage systems. For example, is it worth developing very expensive technologies to allow long DNA molecules to be synthesized? Or is it possible to store data at high densities with very short molecules? An information-theoretic perspective may provide the foundations to answering these questions, enabling the design of an efficient "physical layer" for DNA storage systems.

Emerging technologies beyond DNA: There are many other interesting media for future storage. For example, synthetic polymers are also a potential substrate for data storage [83], in which case tandem mass spectrometry could be used instead of sequencing for data retrieval [62]. Futhermore, the idea of storing data in quartz glass [6] also promises to achieve incredibly high information densities.

As new technologies are proposed and compared, it is important to obtain a basic understanding of their capabilities. In this context, an information-theoretic perspective may allow these emerging approaches to be compared at a more fundamental level, rather than based on specific prototypes. Moreover, design principles may be transferable between them, and different technologies may be more suitable to specific applications depending on basic tradeoffs between cost, computational complexity, and reading and writing speeds. In Section 7.4 we briefly discuss additional storage capacity problems that are motivated by recent technological advances.

Connections with classical information theory problems: The growing interest on DNA data storage has sparked renewed interest in classical

problems in information theory. As several DNA sequencing technologies suffer from insertion and deletion errors, new attention has been given to the capacity of insertion/deletion channels and "sticky" channels [25]. Moreover, many of the topics explored in this monograph will be connected with permutation channels [3], [10] and, in Section 7.3, we discuss a connection between DNA storage in the short-molecule regime and discrete-time Poisson channels [60].

1.4 Organization of this monograph

In Section 2, we formalize and discuss a general class of channels to model DNA storage systems. This general model, called the noisy shuffling-sampling channel, will be the main object of our information-theoretic analysis of DNA-based data storage. In Section 3, we start the exploration of the capacity of DNA storage systems by studying a simple noiseless shuffling-sampling channel, where the input sequences are shuffled and sampled before being observed at the channel output. The capacity expression provides a precise understanding of the storage rate costs that the shuffling and sampling operations incur, and provide intuition on how to develop optimal codes for such channels. In the same section, we also consider channels that break the sequences at random points and shuffle the resulting pieces.

In Section 4, we study the impact of adding noise to the shuffling-sampling channels. Specifically we will study the capacity of channels where the sequences are not only shuffled and randomly sampled, but also contain errors, and we will discuss the impact of different noisy channel models on the overall capacity.

In Section 5, we study the multi-draw nature of DNA storage channels. Since multiple copies of each sequence are typically stored in DNA storage systems, they can be sequenced with potentially different noise patterns, which can help in error correction. This is captured in our information-theoretic framework by considering multi-draw channels, where each sequence in the DNA library may yield multiple copies with independent noise patterns at the output. In this setting, a natural approach is to first cluster the sequences and then solve several *trace reconstruction* problems. We will study whether such clustering-based

schemes are optimal, and we will provide results on the fundamental limits of DNA storage channels with multi-draws.

DNA storage is a relatively new field and many important questions remain unanswered. In Section 7, we end the monograph with a collection of important open problems and a discussion of connections to existing channel models and classical results in information theory.

2

Channel Model

As a consequence of the unique way in which data is stored in DNA and retrieved from DNA, studying the fundamental limits of DNA-based data storage requires the development of new channel models.

Figure 2.1 illustrates the key features of a DNA-based storage system. The data to be stored, described by a sequence of bits, is first encoded into a set of short strings over the alphabet $\{\mathsf{A}, \mathsf{C}, \mathsf{G}, \mathsf{T}\}$. The strings have length of typically less than 200 nucleotides, because existing sequencing technologies can only synthesize short sequences. These strings are synthesized as actual DNA molecules, via a noisy synthesis process. The synthesis process typically generates multiple noisy copies of each DNA molecule, as it is currently not possible to synthesize individual molecules efficiently. The resulting pool of DNA molecules is either stored directly or first amplified with Polymerase Chain Reaction (PCR) and then stored. Prior to being stored, the DNA molecules are often protected in some way, for example they can be encapsulated in silica [39] (see Organick, Nguyen, McAmis, *et al.* [81] for an empirical comparison of a few preservation options). In this form, the DNA molecules suffer relatively small amounts of deterioration over long periods of time under appropriate environmental conditions.

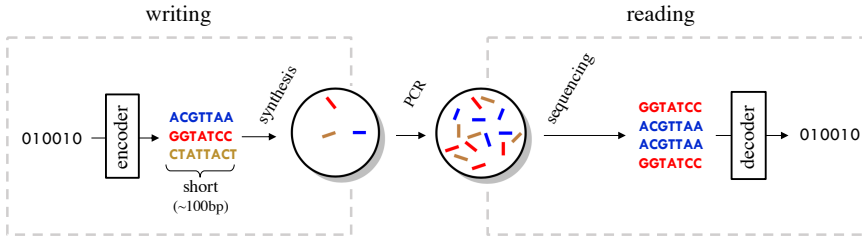

Figure 2.1: Illustration of a DNA-based storage system. In the writing phase, a sequence of bits is encoded into a multi-set of short sequences over the alphabet $\{\mathsf{A}, \mathsf{C}, \mathsf{G}, \mathsf{T}\}$, which are then synthesized into actual DNA molecules and stored. In the reading phase, PCR is often used to replicate the DNA molecules, followed by sequencing, which randomly samples molecules from the DNA library and reads them. The decoder is applied to the resulting multi-set of sequences in order to recover the original sequence of bits.

At the time of reading, the DNA is usually first amplified with PCR, effectively replicating each of the molecules a large number of times, and then sequenced. Sequencing can be thought of as randomly sampling from the mixture. Since multiple copies of each molecule are present due to the amplification step (possibly corrupted by different noise patterns), the same molecule can be observed multiple times at the output (or not at all). The decoder operates on the set of sequenced strings with the goal of recovering the original bit string that encodes the data.

2.1 The noisy shuffling-sampling channel model

In order to study the fundamental limits of DNA-based data storage, we propose a general channel model, which we call the noisy shuffling-sampling channel. This channel model, illustrated in Figure 2.2, captures three key elements of DNA storage:

- **Sampling**: Since the reading of a DNA library is done via sequencing, strings are randomly sampled from the pool. Moreover, since synthesis technologies generate multiple copies of each sequence and PCR replicates the molecules in the pool, the same input sequence can be observed multiple times at the output.

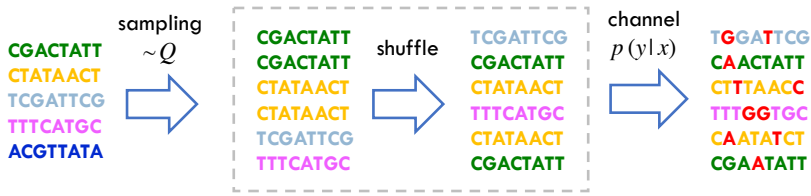

Figure 2.2: The general noisy sampling-shuffling channel. The input are M sequences of length L. The ith input sequence is sampled a number N_i of times, where each N_i is distributed according to a distribution Q. The resulting N sequences are shuffled out of order and each one passed through a noisy channel $p(y|x)$.

- **Shuffling**: The ordering of the DNA molecules is lost during storage, effectively causing them to be shuffled.
- **Noise**: DNA molecules are corrupted by noise at the synthesis, storage and sequencing steps. The type of error (substitutions, insertions, deletions) varies according to the technologies used.

Our general channel model for DNA storage is obtained by concatenating a sampling channel (which creates a random number of copies of each string), a shuffling channel (which shuffles a set of sequences uniformly) and a noisy channel (which corrupts each of the strings).

We formalize the noisy shuffling-sampling channel as follows. The input to the channel is a list $[x_1^L, \ldots, x_M^L]$ of M sequences of length L over an alphabet Σ. The channel performs three operations:

- **Sampling:** Each sequence x_i^L is sampled a number $N_i \sim Q$ of times, for some distribution $Q = (q_0, q_1, \ldots)$, where $q_n = \Pr(N_i = n)$ is the probability that n copies of the sequence x_i^L are drawn. The sampling of different strings is assumed to be independent. We let $N = \sum_{i=1}^{M} N_i$ be the total number of resulting strings, and we define $\lambda \coloneqq \mathbb{E}[N]/M = \mathbb{E}[N_i]$ to be the sequencing *coverage depth* [59], [77].
- **Shuffling:** The N strings are shuffled uniformly at random.
- **Noise:** Each of the N strings is independently passed through a discrete memoryless channel $p(y|x)$, producing a list of N output

sequences $[y_1^L, \ldots, y_N^L]$. Equivalently, the output of the channel can be taken to be the (unordered) multi-set $\{y_1^L, \ldots, y_N^L\}$.

Different specifications of the sampling distribution Q and the noisy channel $p(y|x)$ lead to a rich class of information-theoretic channels whose capacity is in general unknown and nontrivial. As described throughout this monograph, several important questions related to the joint encoding of information across multiple strings arise in the context of this general channel model.

We point out that, since synthesis, storage and sequencing are all error-prone procedures, an even more general channel model would have another noisy channel prior to the sampling step. For simplicity, we will focus on the case with a single, final, noisy channel as this is already a good modeling assumption.

2.2 Motivation for the channel model and error sources

The channel model in Section 2.1 is motivated by the technologies used to synthesize, process, store, handle, and sequence DNA, and errors arise in all those steps. In this section, we discuss the technologies and sources of errors at these stages. We refer to Heckel, Mikutis, and Grass [47] and references therein for a more detailed descriptions of the error mechanisms.

Synthesis. Three different array-based synthesis methods are currently used for DNA data storage. Array-based synthesis methods grow several strands simultaneously on an array. One end of the DNA molecule is attached to the array, and the nucleotides are added one by one through the following mechanisms. The first method is a material deposition or printing-based technology commercialized by the companies Agilent and Twist Biosciences. This method was used for DNA storage by Erlich and Zielinski [33] and Organick, Ang, Chen, *et al.* [80]. The second is electro-chemical, commercialized by CustomArray, and was used for DNA storage by Grass, Heckel, Puddu, *et al.* [39]. The third is based on light-directed placement of nucleotides [103] and was used for DNA storage by Antkowiak, Lietard, Darestani, *et al.* [7].

Each of these technologies produces thousands to millions of copies of each sequence. The printing based and electrochemical synthesis technologies produce relatively few errors within the sequences, but the light-directed technology produces many insertion, deletion, and substitution errors within the sequences. In summary, DNA synthesis produces a pool of DNA sequences, which contain many noisy copies of each sequence.

Processing and storage. The DNA pool is either stored as it is, or we may only select a small part of the synthesized DNA pool for storage and dilute it so that the physical redundancy is relatively small. As mentioned previously, the DNA might be protected through encapsulation or by other means. Either way, during the processing and storage of the DNA, the DNA sequences are prone to chemical decay, which causes substitution errors and occasionally results in the full loss of sequences.

Finally, PCR cycles are often used in a DNA storage system to amplify the DNA. In each PCR cycle, a given sequence is amplified on average by a factor slightly less than two, and through multiple cycles this can lead to a significant imbalance in the sequences in the pool.

Sequencing. The currently most utilized sequencing technology is commercialized by Illumina. This technology induces relatively few substitution errors within the sequences and even fewer insertion and deletion errors. Heckel, Mikutis, and Grass [47] estimated the reading error probabilities based on the two-sided reads to be between 0.15% and 0.4% and those errors are almost exclusively substitution errors. Schirmer, D'Amore, Ijaz, *et al.* [94] estimated a per-base error probability of 0.2% for Illumina reads, a very similar number.

Another popular sequencing technology is nanopore sequencing. Nanopore sequencing has significantly higher error probabilities, and introduces substitution, deletion, and insertion errors. The per-pase error of the Oxford Nanopore Technologies (ONT) MinION sequencer is about 10%, albeit the exact number depends on the operating mode. The relative amount of deletions, substitutions, and insertions is similar for this technology.

2.3 Channel statistics

As described in Section 2.1, the channel parameters that need to be specified are the sampling distribution Q and the noisy channel $p(y|x)$. The sampling and noise distributions depend on the error sources discussed in Section 2.2 and are determined by the technologies that are used. Here, we discuss the sampling and noise statistics for a few real-world DNA storage systems.

Sampling distribution. The sampling distribution is determined by several different factors. The first one is the number of physical copies of each sequence initially stored in the DNA pool, known as the physical redundancy (PR). For example, if the physical redundancy is 100, then the DNA pool consists of $100M$ DNA molecules, where M is the original number of synthesized sequences. The second is the number of PCR cycles (PCRC) used. Each PCR cycle on average multiplies each sequence by a factor slightly less than two. Furthermore, the sequencing read coverage (i.e., the number of sequencing reads divided by M) can be adjusted.

Figure 2.3 depicts the empirical sampling distribution from Heckel, Mikutis, and Grass [47] for four different DNA storage experiments from the literature. The read coverage of all four experiments is relatively similar (about 300-500). However, they have vastly different physical redundancies (PR), i.e., number of physical sequences in the stored pool per original sequence. DNA libraries with smaller physical redundancy typically require more PCR cycles prior to sequencing. However, each PCR cycle increases the imbalance of the sequences in the dataset, because they may lead to some sequences being amplified more than others. This leads to a loss of sequences and increases the tail of the distribution. Figure 2.3 illustrates that the lower the physical redundancy, the more long-tailed is the distribution, and more of the sequences are never seen at the output. More details on the quantification of the sampling distribution can be found in Chen, Takahashi, Organick, *et al.* [21] and Heckel, Mikutis, and Grass [47].

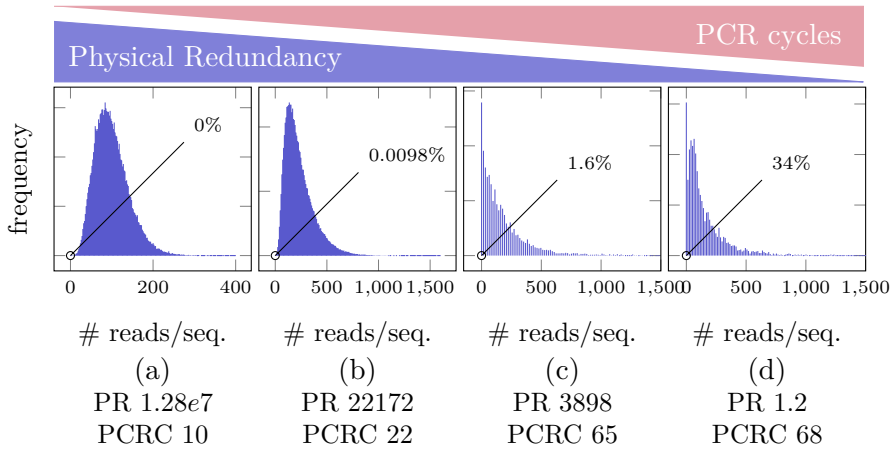

Figure 2.3: The empirical distributions (Figure 5 in Heckel, Mikutis, and Grass [47]) of the number of reads of four different datasets from Erlich and Zielinski [33], Goldman, Bertone, Chen, *et al.* [38], and Grass, Heckel, Puddu, *et al.* [39] per each given sequence that has been synthesized, along with the physical redundancy (PR) and PCR cycles (PCRC). The data in (a) and (b) is approximately negative binomial distributed, whereas the data in (c) and (d) have a long tail and a peak at zero. The percentages in the figure are the fraction of sequences that are never seen at the output. The read coverage (number of reads per original sequence) of all datasets is comparable. Likely, the difference in distribution of (a) and (b) to (c) and (d) is due to the significantly more cycles of PCR in (a) and (b), while the difference of (a) and (b) is due to low physical redundancy and dilution.

Errors within sequences. Errors within sequences are due to synthesis, sequencing, and decay of the DNA. Table 2.1 contains the estimated error probabilities from four DNA storage experiments in the literature. The probabilities indicate the error probability per nucleotide, e.g., a 0.56% substitution error probability means that roughly one out of 200 nucleotides contains a substitution error. In all experiments, the data was stored and read without any artificial aging [39]. Also, in all experiments low-error Illumina sequencing technology was used. Therefore, the errors within sequences in those experiments are to a large extent due to the errors in synthesis. It can be seen that for the established printing-based and electrochemical synthesis technologies, the errors within sequences are relatively small, while the errors for light-directed synthesis are very large.

Table 2.1: Error probabilities within sequences in four data storage experiments.

Experiment	Subst.	Del.	Insert.	Length	Synthesis
Grass, Heckel, Puddu, *et al.* [39]	0.56%	0.99%	0.09%	117	CustomArr.
Erlich and Zielinski [33]	0.36%	0.16%	0.18%	152	Twist
Goldman, Bertone, Chen, *et al.* [38]	0.08%	0.54%	0.02%	117	Agilent
Antkowiak, Lietard, Darestani, *et al.* [7]	2.6%	6.2%	5.7%	60	light-dir.

Note that in the first three datasets, the majority of sequences that contain insertion and deletion errors can be identified easily by their length being longer or shorter than the original sequence, and have been removed in the respective experiments. This results in a set of DNA sequences which are almost free of deletion and insertion errors. For the dataset in Antkowiak, Lietard, Darestani, *et al.* [7] this wouldn't make sense, as most sequences contain insertion or deletion errors, and too many sequences would be removed.

For more error statistics, see the papers by Erlich and Zielinski [33], Heckel, Mikutis, and Grass [47], and Sabary, Orlev, Shafir, *et al.* [88].

2.4 Definitions and notation

In Section 2.1, we formally defined the noisy shuffling-channel, which will be the main object of study of this monograph. In this section we provide additional definitions and notation required to formally analyze the capacity of this channel.

A code $\mathcal{C}$ for a noisy shuffling-sampling channel is a set of codewords, each of which is a list $[x_1^L, \ldots, x_M^L]$ of M strings of length L, together with a decoding procedure. The alphabet Σ of practical interest is typically $\{\mathsf{A}, \mathsf{C}, \mathsf{G}, \mathsf{T}\}$, corresponding to the four nucleotides that compose DNA. However, to simplify the exposition we will often focus on the binary case $\Sigma = \{0, 1\}$. Most of the results can be extended to a quaternary alphabet in a straightforward way, as we briefly discuss in Section 7.1.

In practice, a codeword $[x_1^L, \ldots, x_M^L] \in \mathcal{C}$, which is a set of M sequences each of length L, is stored as a physical mixture of M synthesized DNA molecules. Hence, we use the words molecule and sequence interchangeably.

Our main parameter of interest is the storage rate

$$R \coloneqq \frac{\log |\mathcal{C}|}{ML}, \tag{2.1}$$

i.e., the number of bits stored per DNA base synthesized. To formally define the storage capacity, we consider an asymptotic regime where $M \to \infty$. We let the sequences have length

$$L \coloneqq \beta \log M \tag{2.2}$$

for a fixed parameter $\beta > 0$. This is motivated by the practical constraint that the synthesized molecules should be short. As we will see in the next section, if L grows slower than $\log M$, no positive rate is achievable. Moreover, as our results show, $L = \Theta(\log M)$ is the asymptotic regime of interest for this problem.

We say that storage rate R is achievable if there exists a sequence of DNA storage codes $\{\mathcal{C}_M\}$, each with rate R, such that the decoding error probability tends to 0 as $M \to \infty$. The storage capacity is the supremum over all achievable storage rates.

Additional notation. Throughout this monograph, $\log(\cdot)$ is the logarithm base 2 and $\ln(\cdot)$ is the natural logarithm. For a real number x, we let $(x)^+ = \max(x, 0)$. We let $\mathbb{Z}_+$ be the set of non-negative integers, and for a positive integer N, we let $[1 : N] = \{1, \ldots, N\}$. For a vector $\mathbf{x} = (x_1, \ldots, x_d)$, we let $\|\mathbf{x}\|_2 = \sqrt{\sum_{i=1}^d x_i^2}$ be its ℓ_2 norm, $\|\mathbf{x}\|_1 = \sum_{i=1}^d |x_i|$ its ℓ_1 norm, and $\|\mathbf{x}\|_\infty = \max_i |x_i|$ its ℓ_∞ norm.

We say that a random variable X has the Bernoulli(p) distribution if $\Pr(X = 1) = p$ and $\Pr(X = 0) = 1 - p$. We say that a non-negative integer-valued random variable X has the Poisson(λ) distribution if $\Pr(X = k) = \frac{e^{-\lambda}\lambda^k}{k!}$, for $k \geq 0$.

3

Shuffling Channels

As discussed in Section 2, a distinguishing aspect of DNA-based storage is the fact that the data stored in the DNA library is read out of order, in a shuffled manner. In this section, we focus on the coding and capacity problems that arise in settings where ordering information is lost.

To build intuition, we first briefly discuss a simple shuffling channel which just shuffles the set of input sequences. We then consider the (noise-free) shuffling-sampling channel, where all reads are noise-free; i.e., the discrete memoryless channel $p(y|x)$ in Figure 2.2 is just the identity channel. This will enable us to identify the costs and challenges that arise from the unordered nature of DNA-based storage. Finally, we consider a generalization of the shuffling channel that allows variable-length pieces. This generalization is motivated by the fact that the stored DNA molecules may also be subject to random breaks.

3.1 The shuffling channel

To begin our information-theoretic analysis of DNA-based data storage, we first put aside the noise that corrupts each of the individual DNA molecules and assume that the writing and reading processes are perfect; i.e., every single molecule is synthesized and sequenced correctly. With

these assumptions, the DNA storage channel is described by a shuffling channel, which corresponds to the dashed rectangle in Figure 2.2. The input to a shuffling channel is a list of M strings of length L and the output is an unordered, or shuffled, version of these strings.

The capacity of this noise-free shuffling channel can be shown to be

$$\lim_{M\to\infty}\left(\log|\Sigma| - \frac{\log M}{L}\right)^+ = \left(\log|\Sigma| - \frac{1}{\beta}\right)^+, \tag{3.1}$$

as we argue below. This expression supports the intuition that, when the length L of the DNA molecules is large, the impact of shuffling is small, and we achieve close to the 2 bits/nucleotide that can be achieved for $|\Sigma| = 4$ if there is no shuffling and no noise. Furthermore, equation (3.1) shows that when L scales logarithmically in M, the capacity of the shuffling channel is nontrivial.

The capacity expression in equation (3.1) can be proved with a simple counting argument. Notice that if we view the input and output of the channel as multi-sets of M strings of length L, then the channel does not affect the input at all. Hence, the capacity is simply the logarithm of the number of distinct multi-sets of M strings of length L, divided by the number of nucleotides stored ML.

The operation of counting the number of distinct multi-sets of a given size will be used often throughout this monograph. Notice that multi-sets can be equivalently represented by a histogram, which records how many times each element, out of a list of possible elements, appears. If the number of possible distinct elements is a, a multi-set with b elements can be represented by a vector $\mathbf{t} \in \mathbb{Z}_+^a$ with $\|\mathbf{t}\|_1 = b$.

Lemma 3.1. The number of distinct vectors $\mathbf{t} \in \mathbb{Z}_+^a$ with $\|\mathbf{t}\|_1 = b$ is

$$\mathcal{T}[a,b] \coloneqq \binom{a+b-1}{b} < \left(\frac{e(a+b-1)}{b}\right)^b.$$

Proof. Notice that vectors $\mathbf{t} \in \mathbb{Z}_+^a$ with $\|\mathbf{t}\|_1 = b$ are in one-to-one correspondence with binary strings containing $(a-1)$ zeros and b ones. For $\mathbf{t} = (t_1, \ldots, t_a)$, the corresponding string is

$$\underbrace{1\ldots1}_{t_1}\,0\,\underbrace{1\ldots1}_{t_2}\,0\ldots0\,\underbrace{1\ldots1}_{t_a}. \tag{3.2}$$

Such a string has $(a-1)$ zeros and b ones, and distinct strings with $(a-1)$ zeros and b ones correspond to distinct vectors $\mathbf{t}$. The number of distinct strings of this form is

$$\frac{(a-1+b)!}{(a-1)!\,b!} = \binom{a+b-1}{b}.$$

Finally, the upper bound in the statement of the lemma is a standard bound for binomial coefficients. □

Since the input to the shuffling channel is a multi-set of M strings of length $L = \beta \log M$, the capacity of the shuffling channel is given by

$$\lim_{M\to\infty} \frac{1}{ML} \log \mathcal{T}[|\Sigma|^L, M].$$

Using Lemma 3.1 and the bounds $k\log(n/k) \le \log\binom{n}{k} < k\log(en/k)$, it follows that the capacity is given by

$$\begin{aligned}
\lim_{M\to\infty} \frac{1}{ML} \log \binom{|\Sigma|^L + M - 1}{M} &= \lim_{M\to\infty} \frac{1}{L} \log \left(\frac{|\Sigma|^L + M - 1}{M} \right) \\
&= \lim_{M\to\infty} \frac{1}{L} \log \left(\frac{|\Sigma|^L}{M} + 1 \right) \\
&= \lim_{M\to\infty} \left(\log|\Sigma| - \frac{\log M}{L} \right)^+ \\
&= (\log|\Sigma| - 1/\beta)^+, \qquad (3.3)
\end{aligned}$$

as we wanted to show. In particular, if $\beta \le \frac{1}{\log|\Sigma|}$, the capacity is zero. Notice that this was not obvious a priori since, even for $\beta \le \frac{1}{\log|\Sigma|}$, one still has a large number M of length-L strings with $L \to \infty$ as $M \to \infty$, and it might have been possible to encode data on them at a positive rate.

3.2 Capacity of the shuffling-sampling channel

The simple analysis in Section 3.1 shows that the shuffling aspect of DNA storage effectively causes the capacity to be reduced by $1/\beta$. In this section we analyze the impact of also bringing sampling into the picture. The main result of this section generalizes the simple capacity

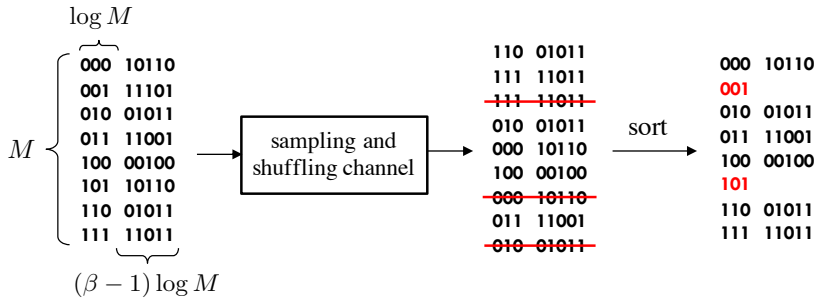

Figure 3.1: Index-based scheme for the shuffling-sampling channel. All M input sequences are prefixed with a unique index of length $\log M$. At the output of the channel, the decoder uses the indices to remove duplicates and sort the molecules. Notice that the missing indices (001 and 101) can be thought of as block erasures. Hence, this scheme effectively creates a block-erasure channel, where blocks of size L are erased with probability q_0.

expression in (3.1) and settles the capacity of the noise-free shuffling-sampling channel, for a general sampling distribution Q. As described in Section 2, the input to the channel are M sequences of length $L = \beta \log M$. The channel samples the ith sequence $N_i \sim Q$ times, and shuffles all sampled sequences. Notice that q_0 is the probability that zero copies of x_i^L are drawn; i.e., q_0 is the expected fraction of input sequences never seen at the output. We focus on the case $\Sigma = \{0, 1\}$ for simplicity of exposition, but the extension to general Σ is straightforward, as discussed in Section 7.1.

Theorem 3.2. The capacity of the shuffling-sampling channel is

$$C = (1 - q_0)\,(1 - 1/\beta)\,. \tag{3.4}$$

In particular, if $\beta \leq 1$, no positive rate is achievable.

The capacity expression in equation (3.4) can be intuitively understood through the achievability argument. A storage rate of $R = (1 - q_0)\,(1 - 1/\beta)$ can be easily achieved by prefixing all the molecules with a distinct tag or index, which effectively converts the channel to a block-erasure channel, as illustrated in Figure 3.1.

More precisely, we use the first $\log M$ bits of each molecule to encode a distinct index. Then we have $L - \log M = (\beta - 1) \log M$ symbols left

per molecule to encode data. The decoder can use the indices to remove duplicates and sort the molecules that are sampled. This effectively creates an erasure channel, where molecule i is erased if it is not drawn (i.e., $N_i = 0$) which occurs with probability q_0. Since the expected number of erasures is

$$\mathbb{E}\left[\frac{1}{M}\sum_{i=1}^{M}\mathbb{1}\left\{N_i = 0\right\}\right] = q_0,$$

we achieve storage rate

$$\frac{(1-q_0)M(L-\log M)}{ML} = (1-q_0)(1-1/\beta). \tag{3.5}$$

The surprising aspect of Theorem 3.2 is that this simple index-based scheme is optimal. It is also worth noting that the capacity expression only depends on the sampling distribution Q through the parameter q_0, i.e., the fraction of sequences that is not seen at the output.

In order to gain intuition on a practical implication of this theorem, suppose that each sequence is drawn according to a Poisson distribution with mean λ, so that in expectation $E[N] = \lambda M$ sequences in total are drawn and λ can be thought of as the sequencing coverage depth. The probability that a sequence is never drawn is $e^{-\lambda}$ and the capacity is

$$C = (1-e^{-\lambda})(1-1/\beta). \tag{3.6}$$

This suggests that practical systems should not operate at a very high coverage depth λ, as high coverage depth significantly increases the time and cost of reading, but only provides little storage gains, according to the capacity expression. Notice that, in order to guarantee that all M sequences are observed at least once, we need $N = \Omega(M \log M)$ [59], [77]. When M is large, it is wasteful to operate in this regime, as this only gives a marginally larger storage capacity, but the sequencing costs can be exorbitant.

The result in Theorem 3.2 is flexible to allow different sampling models. In particular, one can consider separating the PCR amplification performed on each synthesized molecule from the sequencing step. Since one cannot control the PCR amplification factor precisely, it is reasonable to assume that a molecule x^L is first randomly amplified and

a total of $A \geq 0$ copies are stored. If we consider a Poisson sampling model for the sequencing step, the effective coverage depth is $\lambda/\mathbb{E}[A]$ (since we are actually sampling from $M\mathbb{E}[A]$ molecules). In this case, the probability that none of the copies of x^L is sampled at the output is $\mathbb{E}\left[(e^{-\lambda/\mathbb{E}[A]})^A\right] = \mathbb{E}\left[(e^{(-\lambda/\mathbb{E}[A])A}\right]$. This can be recognized as the moment-generating function of A evaluated at $-\lambda/\mathbb{E}[A]$. In particular, when PCR is also modeled as a Poisson random variable with mean $\mathbb{E}[A] = \alpha$, $\mathbb{E}\left[e^{\theta A}\right] = e^{\alpha(e^{\theta}-1)}$, and the capacity of the resulting noise-free shuffling-sampling channel is

$$C = \left(1 - e^{-\alpha(1-e^{-\lambda/\alpha})}\right)(1 - 1/\beta). \tag{3.7}$$

The capacity can be similarly computed based on other sampling distributions (such as the Negative Binomial distribution observed in Section 2.3).

3.2.1 Motivation for converse

One can attempt to prove a converse by following the approach in Section 3.1. Notice that one can view the noise-free shuffling-sampling channel as a channel where the encoder chooses a histogram over the 2^L sequences of length L and the decoder observes a noisy version of this histogram where the frequencies are perturbed according to Q. From this angle, the question becomes "how many histograms $\mathbf{t} \in \mathbb{Z}_+^{2^L}$ with $\|\mathbf{t}\|_1 = M$ can be reliably decoded?"

Following the counting argument in Section 3.1, the total number of distinct histograms $\mathbf{t} \in \mathbb{Z}_+^{2^L}$ with $\|\mathbf{t}\|_1 = M$ is $\mathcal{T}[2^L, M]$. Since this is the total number of histograms that can be used as an input, (3.3) implies that a bound to the shuffling-sampling channel capacity is

$$C \leq 1 - 1/\beta. \tag{3.8}$$

Therefore, if we had a shuffling-sampling channel where the decoder observed exactly the M stored molecules (i.e., $q_0 = 0$ and $q_1 = 1$), the index-based approach would be optimal from a rate standpoint.

A simple outer bound that involves q_0 can be obtained by considering a genie that provides the decoder with the "true" index of each sampled molecule. In other words, $[x_1^L, \ldots, x_M^L]$ are the stored molecules, and

the decoder observes $[y_1^L, \ldots, y_N^L]$ and the mapping $\sigma\colon \{1, \ldots, N\} \to \{1, \ldots, M\}$ so that $y_j^L = x_{\sigma(j)}^L$. This converts the channel into an erasure channel with block-erasure probability q_0, which yields

$$C \leq 1 - q_0. \tag{3.9}$$

It is intuitive that the bound (3.9) should not be achievable, as the decoder in general cannot sort the molecules and create an effective erasure channel. However, it is not clear a priori whether prefixing every molecule with an index is optimal.

Combining (3.8) and (3.9) implies that $C \leq \min(1 - q_0, 1 - 1/\beta)$, but there is still a gap to the achievable rate in (3.5). The converse presented in the next section utilizes a more careful genie that does not give us the permutation σ (but something weaker) to show that $C \leq (1 - q_0)(1 - 1/\beta)$, implying the optimality of index-based coding approaches.

3.2.2 Converse

Let $[x_1^L, \ldots, x_M^L]$ be the M length-L molecules written into the channel and $[y_1^L, \ldots, y_N^L]$ be the length-L molecules observed by the decoder. Notice that, whenever the channel output is such that $y_i^L = y_j^L$ for $i \neq j$, the decoder cannot determine whether both y_i^L and y_j^L were sampled from the same molecule x_ℓ^L or from two different molecules that obey $x_\ell^L = x_k^L, \ell \neq k$. In order to derive the converse, we consider a genie-aided channel that removes this ambiguity. As illustrated in Figure 3.2, before sampling the N molecules, the genie-aided channel appends a unique index of length $\log M$ to each molecule x_i^L, which results in the set of tagged molecules $\{(x_i^L, z_i)\}_{i=1}^M$. We emphasize that the indices z_i are all unique, and are chosen randomly and independently of the input sequences $\{x_i^L\}_{i=1}^M$. Notice that, in contrast to the naive genie discussed in Section 3.2.1, this genie does *not* reveal the index i of the molecule x_i^L from which y_ℓ^L was sampled. Therefore, the channel is not reduced to an erasure channel, and intuitively the indices are only useful for the decoder to determine whether two equal samples $y_\ell^L = y_k^L$ came from the same molecule or from distinct molecules.

The output of the genie-aided channel, denoted by $\{(y_i^L, z_{\sigma(i)})\}_{i=1}^N$, is then obtained by sampling from the set of tagged molecules $\{(x_i^L, z_i)\}_{i=1}^M$,

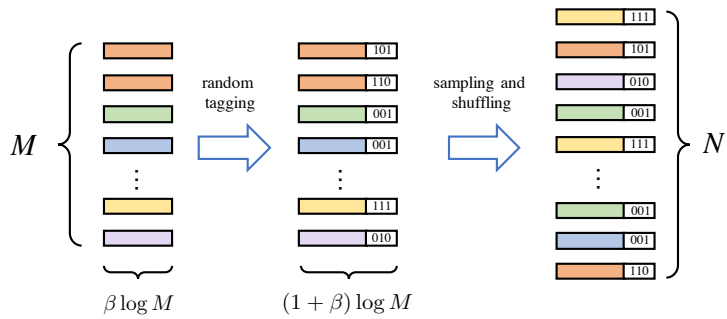

Figure 3.2: Genie-aided channel for converse for the shuffling-sampling channel. The genie appends a random (but unique) index to each of the input sequences $x_1^L, \ldots, x_M^L$. This allows the decoder to identify which outputs sequences originated from the same input sequences, and remove them.

in the same way as the original channel samples the original molecules. The mapping $\sigma\colon [1:N] \to [1:M]$ is such that y_i^L was sampled from $x_{\sigma(i)}^L$. Notice that the actual mapping σ is not revealed to the decoder.

It is clear that any storage rate achievable in the original channel can be achieved on the genie-aided channel, as the decoder can simply discard the indices, or stated differently, the output of the original channel can be obtained from the output of the genie-aided channel.

Notice that $\{(y_i^L, z_{\sigma(i)})\}_{i=1}^N$ is in general a multi-set. We will let $\mathrm{set}(\{(y_i^L, z_{\sigma(i)})\}_{i=1}^N)$ be the set obtained from $\{(y_i^L, z_{\sigma(i)})\}_{i=1}^N$ by removing any duplicates. Then $\mathrm{set}(\{(y_i, z_{\sigma(i)})\}_{i=1}^N)$ is a sufficient statistic for $\{x_i^L\}_{i=1}^M$ since all tagged molecules are distinct objects, and sampling the same tagged molecule (x_i^L, z_i) does not yield additional information on $\{x_i^L\}_{i=1}^M$. More formally, conditioned on $\mathrm{set}(\{(y_i^L, z_{\sigma(i)})\}_{i=1}^N)$, $\{x_i^L\}_{i=1}^M$ is independent of the genie's channel output $\{(y_i^L, z_{\sigma(i)})\}_{i=1}^N$.

Next, we define the frequency vector $\mathbf{f} \in \mathbb{Z}_+^{M^\beta}$ (note that $|\Sigma^L| = 2^{\beta \log M} = M^\beta$) that is obtained from $\mathrm{set}(\{(y_i^L, z_{\sigma(i)})\}_{i=1}^N)$ as follows. The entry of $\mathbf{f}$ corresponding to the molecule $y^L \in \Sigma^L$ is given by

$$\mathbf{f}[y^L] := \left| \left\{ (y_j^L, z_{\sigma(j)}) \in \mathrm{set}(\{(y_i^L, z_{\sigma(i)})\}_{i=1}^N) \colon y_j^L = y^L \right\} \right|.$$

The frequency vector $\mathbf{f}$ is essentially a histogram that counts the number of occurrences of y^L in the set of tagged molecules $\{(y_i^L, z_{\sigma(i)})\}_{i=1}^N$. Notice that the entries of $\mathbf{f}$ can take values greater than one, because at the input we can choose to use the same molecule for multiple x_i^L.

Since $\mathrm{set}(\{(y_i^L, z_{\sigma(i)})\}_{i=1}^N)$ is a sufficient statistic for $\{x_i^L\}_{i=1}^M$ and the tags added by the genie were chosen at random and independently of $\{x_i^L\}_{i=1}^M$, it follows that $\mathbf{f}$ is also a sufficient statistic for $\{x_i^L\}_{i=1}^M$. Hence, we can view the (random) frequency vector $\mathbf{f}$ as the output of the channel without any loss. Notice that $|\mathrm{set}(\{(y_i^L, z_{\sigma(i)})\}_{i=1}^N)| = \|\mathbf{f}\|_1$, and we have $\|\mathbf{f}\|_1 \leq M$ and $\mathbb{E}[\|\mathbf{f}\|_1/M] = 1 - q_0$. The following lemma asserts that $\|\mathbf{f}\|_1$ does not exceed its expectation by much.

Lemma 3.3. For any $\delta > 0$, the frequency vector $\mathbf{f}$ at the output of the genie-aided channel satisfies

$$\Pr\left(\frac{\|\mathbf{f}\|_1}{M} > 1 - q_0 + \delta\right) \to 0, \text{ as } M \to \infty.$$

Proof. Note that the number of distinct fragments drawn is

$$\frac{\|\mathbf{f}\|_1}{M} = \frac{1}{M}\sum_{i=1}^{M} \mathbb{1}\{N_i > 0\}.$$

Since $\mathbb{1}\{N_i > 0\}$ are independent random variables with expectation $1 - q_0$, Hoeffding's inequality yields

$$\Pr\left(\frac{\|\mathbf{f}\|_1}{M} \geq (1 - q_0) + \delta\right) \leq e^{-2M\delta^2},$$

which concludes the proof. □

We remark that an analogue of Lemma 3.3 can be proved for the sampling-with-replacement model, as described in Shomorony and Heckel [98].

We now append the coordinate $f_0 = (1 - q_0 + \delta)M - \|\mathbf{f}\|_1$ to the beginning of $\mathbf{f}$ to construct $\mathbf{f}' = (f_0, \mathbf{f})$. Notice that when $\|\mathbf{f}\|_1 \leq (1 - q_0 + \delta)M$ (which by Lemma 3.3 happens with high probability), we have $\|\mathbf{f}'\|_1 = (1 - q_0 + \delta)M$. This construction of $\mathbf{f}'$ will allow us to utilize Lemma 3.1 below. Fix $\delta > 0$, and define the event

$$\mathcal{E} = \{\|\mathbf{f}\|_1 > (1 - q_0 + \delta)M\} \tag{3.10}$$

with indicator function $\mathbb{1}_{\mathcal{E}}$. By Lemma 3.3, $\Pr(\mathcal{E}) \to 0$ as $M \to \infty$. Consider a sequence of codes $\{\mathcal{C}_M\}$ with rate R and vanishing error

probability. Let W be the message to be encoded, chosen uniformly at random from $\{1, \ldots, 2^{MLR}\}$. From Fano's inequality we have

$$\begin{aligned} MLR_s = H(W) &= I(W; \mathbf{f}') + H(W|\mathbf{f}') \\ &\leq H(\mathbf{f}') + 1 + P_e MLR_s, \end{aligned} \tag{3.11}$$

where P_e is the probability of a decoding error, which by assumption goes to zero as $M \to \infty$. We can then upper bound the achievable storage rate R as

$$\begin{aligned} MLR(1 - P_e) &\leq H(\mathbf{f}') + 1 \\ &\leq H\left(\mathbf{f}', \mathbb{1}_{\mathcal{E}}\right) + 1 \\ &\leq \Pr\left(\mathcal{E}\right) H\left(\mathbf{f}' \,|\, \mathcal{E}\right) + \Pr\left(\bar{\mathcal{E}}\right) H\left(\mathbf{f}' \,|\, \bar{\mathcal{E}}\right) + H(\mathbb{1}_{\mathcal{E}}) + 1. \end{aligned} \tag{3.12}$$

Note that the vector $\mathbf{f}'$ above has dimension $M^\beta + 1$ and, given the event $\bar{\mathcal{E}}$ occurs, $\|\mathbf{f}'\|_1 = (1 - q_0 + \delta)M$, and we have $H(\mathbf{f}'|\bar{\mathcal{E}}) \leq \log \mathcal{T}[M^\beta + 1, (1 - q_0 + \delta)M]$, where $\mathcal{T}[a, b]$ is the number of vectors $x \in \mathbb{Z}_+^a$ with $\|x\|_1 = b$. From Lemma 3.1,

$$\begin{aligned} &\log \mathcal{T}[M^\beta + 1, (1 - q_0 + \delta)M] \\ &\quad \leq (1 - q_0 + \delta)M \log\left(e + \frac{eM^{\beta-1}}{(1 - q_0 + \delta)}\right) \\ &\quad \leq (1 - q_0 + \delta)M \log\left(\alpha M^{\beta-1}\right) \\ &\quad \leq (1 - q_0 + \delta)M[(\beta - 1)\log M + \log \alpha], \end{aligned}$$

where α is a positive constant. Moreover, we notice that $\mathbf{f}'$ is a function of $\mathbf{f}$, which is a vector in $\mathbb{Z}_+^{M^\beta}$ with $\|\mathbf{f}\|_1 \leq M$. Next, we define $\mathbf{f}'' = (f_0, \mathbf{f})$, where $f_0 = M - \|\mathbf{f}\|_1$ so that $\|\mathbf{f}''\|_1 = M$ and we can apply Lemma 3.1. We note that

$$\begin{aligned} H(\mathbf{f}'|\mathcal{E}) = H(\mathbf{f}''|\mathcal{E}) &\leq \log \mathcal{T}[M^\beta + 1, M] \\ &\leq M \log\left(\frac{e(M + M^\beta}{M}\right) \\ &\leq M((\beta - 1)\log M + \log \alpha'), \end{aligned}$$

where α' is another positive constant. Dividing (3.12) by ML and applying the bounds above yields

$$\begin{aligned} R(1-P_e) &\le \Pr(\mathcal{E})\frac{M[(\beta-1)\log M+\log\alpha']}{ML} \\ &\quad+\frac{(1-q_0+\delta)M[(\beta-1)\log M+\log\alpha]}{ML}+\frac{2}{ML} \\ &\le \Pr(\mathcal{E})\left(\frac{\beta-1}{\beta}+\frac{\log\alpha'}{\beta\log M}\right) \\ &\quad+(1-q_0+\delta)\left(1-\frac{1}{\beta}+\frac{\log\alpha}{\beta\log M}\right)+\frac{2}{ML}. \end{aligned}$$

Finally, letting $M\to\infty$ yields

$$R \le (1-q_0+\delta)\,(1-1/\beta)\,,$$

since $\Pr(\mathcal{E})\to 0$ by Lemma 3.3. Since $\delta>0$ can be chosen arbitrarily small, this concludes the converse proof.

3.3 Storage-sequencing tradeoff

Most studies on DNA-based storage emphasize the storage rate (or storage density) as the main figure of merit, while sequencing costs are often disregarded. This is due to the fact that current costs of sequencing technologies are orders of magnitude lower than synthesis costs. However, it is still important to understand, for a given storage rate, how much sequencing is required for reliable decoding, as this determines the time and cost required for retrieving the data.

From this perspective, it makes sense to analyze the tradeoff between storage rates and the amount of sequencing required for reliable recovery. To do this, we can consider, in addition to the storage rate, the *recovery rate*, defined as the number of bits recovered per DNA base sequenced,

$$R_r := \frac{\log|\mathcal{C}|}{NL}. \tag{3.13}$$

In a practical setting, one can control the amount of sequencing performed, typically specified in terms of the coverage depth N/M. If we consider the error-free shuffling-sampling channel from Section 3.2, in the case where Q is a Poisson distribution with mean λ, then $\lambda = N/M$

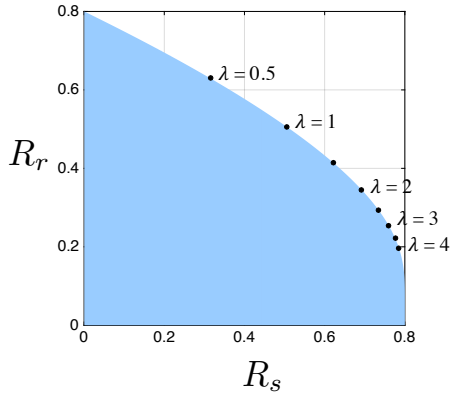

Figure 3.3: (R_s, R_r) feasibility region for $\beta = 5$.

is the coverage depth, and one would like to choose a value of λ that achieves a good trade-off between storage rate and recovery rate.

If we let R_s be the storage rate (previously just R, see (2.1)), from Theorem 3.2 and the fact that $R_s = \lambda R_r$, the (R_s, R_r) feasibility region can be fully characterized.

Corollary 1. For the error-free shuffling-sampling channel with $Q \sim \text{Poisson}(\lambda)$, rates (R_s, R_r) are achievable if and only if, for some $\lambda > 0$,

$$R_s \leq (1 - e^{-\lambda})\,(1 - 1/\beta)\,,$$
$$R_r \leq \frac{1 - e^{-\lambda}}{\lambda}\,(1 - 1/\beta)\,.$$

This region is illustrated in Figure 3.3. This tradeoff suggests that a good operating point is achieved by not trying to maximize the storage rate (which technically requires $\lambda \to \infty$). Instead, by using some modest coverage depth $\lambda = 1, 2, 3$, most of the storage rate $(63\%, 86\%, 95\%$, respectively) can be achieved. This is in contrast to what has been done in practical DNA storage systems that have been developed thus far, where the decoding phase utilizes very deep sequencing.

To be concrete, suppose we are interested in minimizing the overall cost of storing data on DNA. Synthesis costs can be several orders of magnitude higher than sequencing costs. Suppose that per-base synthesis costs are α times larger than per-base sequencing costs. Then,

if our goal is to minimize the cost for synthesizing and sequencing a given number of bits, the overall cost is proportional to

$$\frac{q}{R_s} + \frac{1}{R_r} = \frac{q+\lambda}{(1-e^{-\lambda})(1-1/\beta)}.$$

This quantity can be minimized over λ, yielding the optimal cost per bit. For example, if $q = 100$, $\lambda = 4.7$ and if $q = 10000$, $\lambda = 9.2$. This suggests that it is possible to achieve reliable DNA storage that minimizes overall costs using only moderate coverage depths. We point out that, in practice, one may be interested in optimizing other quantities such as reading time or considering a scenario where the data is read more than once.

3.4 DNA breaks and variable-length pieces

During synthesis, storage, and sequencing, the DNA molecules in a DNA storage system are subject to random breaks. While a careful handling of the DNA library can minimize the occurrence of such breaks, from a system design point of view, it is interesting to study the impact that such breaks may have on the overall system capacity. Such an understanding can enable system designs that combat DNA degradation through coding, and that therefore do not require a tight control of the temperature of the DNA library. Furthermore, in current implementations of DNA storage systems, the data is read via high-throughput sequencing, which is typically preceded by physical fragmentation of the DNA with techniques like *sonication*, which utilizes sound vibrations to fragment the DNA in random locations [84].

A basic initial model to study breaks in DNA molecules is the *torn-paper channel*, proposed by Shomorony and Vahid [99] and illustrated in Figure 3.4. In this setting, the channel input is a length-n binary sequence x^n that is torn into pieces by the channel. A tearing point between any two consecutive symbols in x^n is assumed to occur with a fixed probability $p_n \in (0,1)$ and independently from all other possible tearing locations. As a result, the lengths $N_1, N_2, \ldots$ of each of the fragments produced by the channel has a Geometric(p_n) distribution. The channel output is a shuffled list of these pieces or, equivalently, an unordered set containing these pieces.

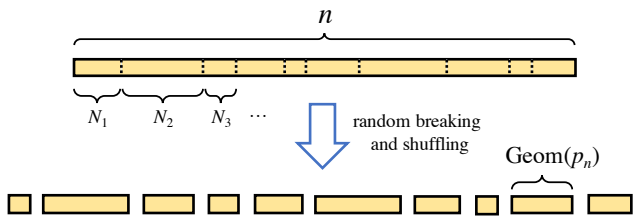

Figure 3.4: In the torn-paper channel, a single (binary) sequence of length n is sent through the channel. The channel tears the input sequence into pieces of random sizes. If tearing points occur according to an i.i.d. Bernoulli process, as considered in Shomorony and Vahid [99], then the fragment sizes will follow a Geometric(p_n) distribution.

As it turns out, characterizing the capacity of this channel is non-trivial even in this noise-free setting. To build some intuition, notice that the expected fragment length is $E[N_i] = 1/p_n$. Suppose that, instead of random-length fragments, we have fragments of a deterministic length $1/p_n$. In this case, the channel breaks the length-n sequence x^n into np_n pieces of equal length. Notice that since, in this case, the tearing locations are known a priori, the channel is identical to the error-free shuffling channel discussed in Section 3.2, with every piece being sampled exactly once. In the notation from Section 3.2, the capacity of an error-free shuffling channel where each piece is drawn exactly once is $1 - 1/\beta$, where β is the ratio between the logarithm of the number of pieces and the piece length. This implies that, for the torn-paper channel with pieces of a deterministic size $1/p_n$, the capacity is

$$\lim_{n\to\infty} 1 - \frac{\log(np_n)}{1/p_n} = \lim_{n\to\infty} 1 - p_n \log n - p_n \log p_n.$$

We see that the regime of interest (where the capacity is non-trivial) is when p_n scales as $1/\log n$, in which case the capacity reduces to

$$C_{\text{deterministic length}} = (1 - \lim_{n\to\infty} p_n \log n)^+. \tag{3.14}$$

Mimicking the notation from previous sections, it is reasonable to let

$$\beta \coloneqq \lim_{n\to\infty} \frac{1/p_n}{\log(np_n)} = \lim_{n\to\infty} \frac{1}{p_n \log n},$$

where the last equality assumes the regime $p_n \sim 1/\log n$. The quantity β plays the same role of a "normalized fragment length" (normalized by the logarithm of the number of pieces) as it did in previous sections. From (3.14), we now have that the capacity for the case of deterministic fragment lengths becomes

$$\left(1 - \frac{1}{\beta}\right)^+, \tag{3.15}$$

analogous to the noise-free shuffling channel considered in Section 2.1.

It is not clear a priori whether the capacity of the torn-paper channel with random fragment lengths should be higher or lower than $(1-1/\beta)^+$. The fact that the tearing points are not known to the encoder makes it challenging to place a unique index in each fragment, suggesting that the torn-paper channel is "harder" and should have a lower capacity. However, this intuition is incorrect and the capacity of the torn-paper channel with Geometric(p_n)-length fragments is in fact higher than $(1 - 1/\beta)^+$. More precisely, we have the following result.

Theorem 3.4. The capacity of the torn-paper channel is

$$C_{\text{TPC}} = e^{-1/\beta}, \tag{3.16}$$

where $\beta = \lim_{n\to\infty} \frac{1}{p_n \log n}$.

The comparison between C_{TPC} and (3.14) is shown in Figure 3.5. Intuitively, this boost in capacity comes from the tail of the geometric distribution, which guarantees that a fraction of the fragments will be significantly larger than the mean $E[N_i] = 1/p_n$. This allows the capacity to be positive even for $\beta \leq 1$, in which case the capacity of the deterministic-tearing case in (3.14) becomes 0. We refer to Shomorony and Vahid [99] for the proof of Theorem 3.4.

General torn-paper channel capacity. The capacity expression of the torn-paper channel can be generalized to the case of arbitrary fragment length distribution. Moreover, it can be generalized to allow some of the fragments to be lost and not be observed at the output.

Consider a torn-paper channel that breaks the length-n sequence x^n into pieces of lengths $N_1, N_2, \ldots$, following a common distribution. In

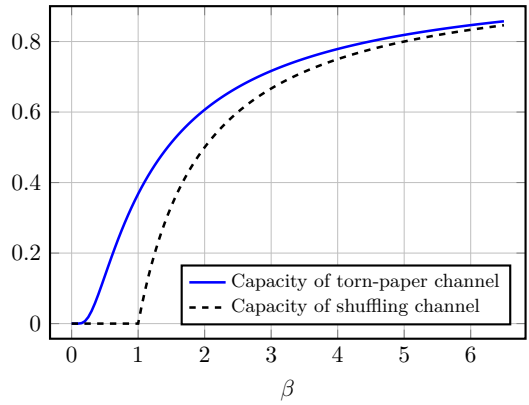

Figure 3.5: Comparison between the capacity of the torn-paper channel $C = e^{-1/\beta}$ and the capacity of the shuffling channel with fragments of fixed length $1/p_n$.

addition, pieces of length ℓ are deleted according to a length-dependent deletion probability $d(\ell)$. As shown by Ravi, Vahid, and Shomorony [86], the capacity of this channel is described by the informal expression

$$C_{\text{general TPC}} = \text{coverage} - \text{reordering-cost}. \tag{3.17}$$

Here, "coverage" refers to the fraction of x^n that is covered by pieces of length at least $\log n$ that are observed at the channel output, and "reordering cost" refers to the fraction of bits that would need to be dedicated to indices in order to "unshuffle" the pieces longer than $\log n$.

In the case of the error-free shuffling channel where each piece is observed exactly once, as long as the pieces have length at least $\log n$, the coverage is 1, and the reordering cost is the fraction of bits needed for adding $\log(np_n)$ indexing bits to each of the np_n fragments; i.e.,

$$\lim_{n\to\infty} \frac{np_n \log(np_n)}{n} = \lim_{n\to\infty} p_n \log n = \frac{1}{\beta},$$

yielding the capacity formula $(1 - 1/\beta)^+$. In the case of piece lengths distributed as Geometric(p_n), using the Exponential approximation to a Geometric random variable, the coverage by pieces of length at least $\log n$ can be shown to be $(1 + 1/\beta)e^{-1/\beta}$, and the reordering cost can be shown to be $\beta^{-1}e^{-1/\beta}$, yielding the capacity expression $e^{-1/\beta}$.

The recipe in (3.17) can be used to derive other closed-form capacity expressions for different piece-length distributions and deletion probability functions $d(\ell)$. As shown in Ravi, Vahid, and Shomorony [86], the capacity depends on $d(\ell)$ through the asymptotic behavior of $d(\ell)$, which is captured by

$$\hat{d}(\xi) \coloneqq \lim_{n\to\infty} d(\xi \log n) \log n.$$

Table 3.1 shows several examples of capacity expressions for torn-paper channel with specific choices of fragment length distribution and deletion probability $\hat{d}(\xi)$.

Table 3.1: Capacity for different torn-paper channels

N_i	$\hat{d}(\xi)$	Capacity
Geometric(p_n)	0	$e^{-1/\beta}$
Geometric(p_n)	ϵ	$(1-\epsilon)e^{-1/\beta}$
Geometric(p_n)	$e^{-\gamma\xi}$	$e^{-1/\beta}\left(1-\frac{\beta^{-2}e^{-\gamma}}{(\beta^{-1}+\gamma)^2}\right)$
Unif$[0:\gamma\log n]$, $\gamma \geq 1$	0	$((\gamma-1)/\gamma)^2$
fixed ℓ_n, $\ell_n \geq \log n$	0	$(1-1/\beta)^+$

4

Noisy Shuffling Channels

In Section 3, we studied the impact of two key aspects of DNA storage systems: the natural shuffling that occurs due to the fact that molecules are stored in an unordered fashion, and the output sampling that occurs due to the sequencing operation. In this section, we study the additional effect of errors within the stored sequences.

In order to focus on the impact of noise, we consider a simpler sampling distribution Q than the general one considered in Section 3.2. Specifically, we focus on a "single-draw" setting, where each sequence is drawn either once or not at all (but never multiple times). The main result in this section characterizes the capacity of the noisy shuffling-sampling channel in Figure 2.2 when Q is a Bernoulli$(1-q)$ distribution and the noisy channel is a binary symmetric channel with crossover probability p. We then discuss extensions to the binary erasure channel and to more general channels. In all these cases, the capacity expression, at least in some parameter regime, is shown to be

$$(1-q)(C_{\text{noisy}} - 1/\beta)^+, \tag{4.1}$$

where C_{noisy} is the capacity of the noisy channel $p(y|x)$ in Figure 2.2, leading to the conjecture that (4.1) holds for general discrete memoryless channels (see Section 7.1).

4.1 Noisy shuffling-sampling channel with single draws

In this section we study the capacity of the noisy shuffling-sampling channel where the sampling distribution is Bernoulli$(1-q)$ for a fixed parameter q. Hence we have $\Pr(N_i = 0) = q$ and $\Pr(N_i = 1) = 1 - q$, for $i = 1, \ldots, M$, and the number of output sequences N satisfies $E[N] = (1-q)M$. Moreover, we will assume that the molecules are each corrupted by a BSC with crossover probability p. We refer to this channel as the BSC shuffling-sampling channel.

As in the error-free shuffling-sampling channel considered in Section 3.2, we consider an index-based coding scheme. As we will show, for a large set of parameters p and β, this scheme is capacity-optimal. More precisely, we describe a scheme based on an outer and an inner code and argue that it achieves a rate arbitrary close to

$$R_{\text{index}} = (1-q)(C_{\text{BSC}} - 1/\beta), \tag{4.2}$$

where $C_{\text{BSC}} = 1 - H(p)$ is the capacity of a BSC with crossover probability p. This scheme is depicted in Figure 4.1. As the outer code, we take an erasure-correcting code with block length M and rate $(1-q)$, where each symbol is itself a binary string of length $L(1-H(p)-1/\beta-\epsilon) \approx LC_{\text{BSC}} - \log M$, for some small $\epsilon > 0$. As inner code, we take a code designed for a BSC with codewords of length L and rate $R_{\text{BSC}} = 1 - H(p) - \epsilon \approx C_{\text{BSC}}$. We first encode the data using the outer code, which yields M symbols given as binary strings of length

$$L(1 - H(p) - 1/\beta - \epsilon) = LR_{\text{BSC}} - \log M.$$

We take each symbol, add a unique binary index of length $\log M$ and encode the resulting sequence using the BSC code, which yields M length-L sequences. With this scheme, we encode a total of $(1-q)M(LR_{\text{BSC}} - \log M)$ data bits, with a data rate of

$$\frac{(1-q)M\,(LR_{\text{BSC}} - \log M)}{ML} = (1-q)(R_{\text{BSC}} - 1/\beta). \tag{4.3}$$

Since $\epsilon > 0$ can be chosen arbitrarily small, this scheme achieves a rate arbitrarily close to the rate given in (4.2), as claimed. For simplicity, in this short argument we did not take into account that the inner

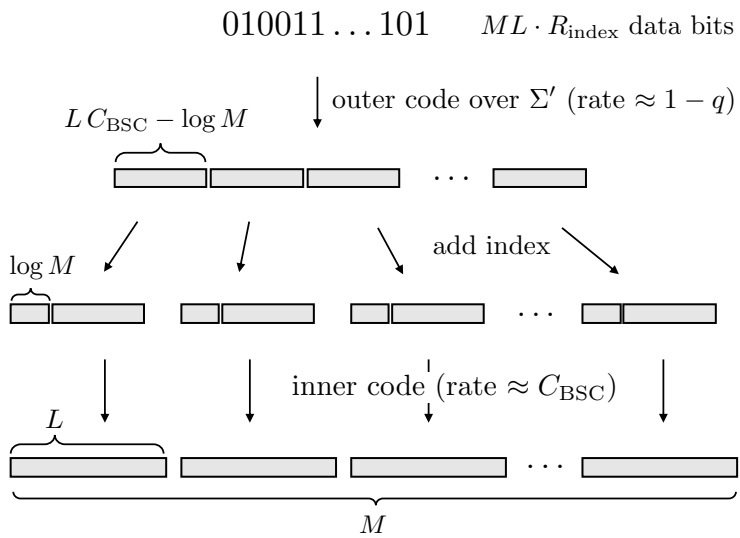

Figure 4.1: Index-based encoding for the BSC shuffling-sampling channel. First, $ML \cdot R_{\text{index}}$ data bits are encoded using an erasure outer with rate $1-q$, where each codeword comprises M symbols from an alphabet Σ' with $\log|\Sigma'| = LC_{\text{BSC}} - \log M$ (or equivalently, each symbol is a binary string of length $LC_{\text{BSC}} - \log M$). A unique $\log M$-bit index is added to each symbol, and a capacity-achieving BSC code is applied to each length-LC_{BSC} binary string, producing M length-L binary strings.

codeword is decoded in error with a vanishing probability; we refer to Shomorony and Heckel [98] for a more formal achievability argument taking this into account.

On the other hand, the result from Section 3.2, with $Q \sim \text{Ber}(1-q)$ implies that $C \leq (1-q)(1-1/\beta)$, since the error-free shuffling-sampling channel cannot be worse than the noisy shuffling-sampling channel. Furthermore, a simple genie-aided argument where the decoder observes the shuffling map can be used to establish that $C \leq (1-q)C_{\text{BSC}}$, where $C_{\text{BSC}} = 1 - H(p)$ is the capacity of a BSC with crossover probability p. Hence, a capacity upper bound is given by

$$C \leq (1-q)\min\left[1 - H(p), 1 - 1/\beta\right]. \tag{4.4}$$

Our main result improves on the upper bound in (4.4), and establishes that for parameters (p, β) in a certain regime, the lower bound in equation (4.2) is the capacity.

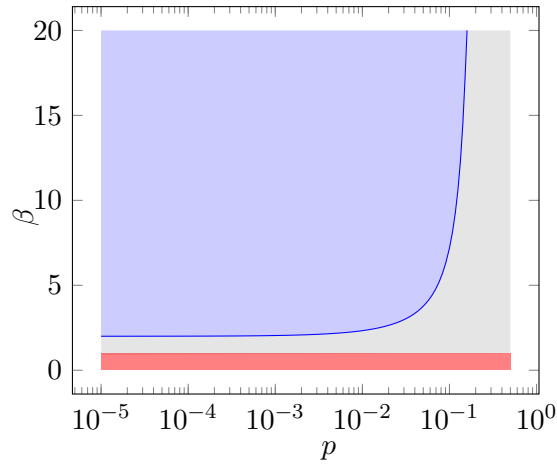

Figure 4.2: Parameter regions for which the capacity is characterized. The capacity in the blue region is given by $C = (1-q)(1-H(p)-1/\beta)$, and the capacity in the red region (i.e., for $\beta < 1$) is 0. In the gray region, it is still unknown.

Theorem 4.1. For the BSC shuffling-sampling channel,

$$C = (1-q)(1-H(p)-1/\beta), \tag{4.5}$$

as long as $p < 1/4$ and $1 - H(2p) - 2/\beta > 0$. Moreover, if $\beta \leq 1$, the capacity is $C = 0$.

The set of parameters (p, β) such that $1 - H(2p) - 2/\beta > 0$ and $p < 1/4$ is the blue region in Figure 4.2. In particular, (4.5) holds if $p \leq 0.1$ and $\beta \geq 6.4$, or if $p \leq 0.01$ and $\beta \geq 2.35$.

4.1.1 Converse

To derive the converse, we view the input to the channel as a binary string of length ML, denoted by

$$X^{ML} = \left[X_1^L, X_2^L, \ldots, X_M^L\right] \in \{0,1\}^{ML}$$

or, equivalently, M strings of length L concatenated to form a single string of length ML. Similarly, the output of the channel is

$$Y^{NL} = \left[Y_1^L, Y_2^L, \ldots, Y_N^L\right] \in \{0,1\}^{NL},$$

where $N = \sum_i N_i$. It is useful to define a vector $S^N \in \{1, \ldots, M\}^N$ indicating the input string from which each output string was sampled. Furthermore, we let $Z^{NL} = \left[Z_1^L, \ldots, Z_N^L\right]$ be the random binary error pattern created by the BSC on the N non-deleted strings. We can now define the input-output relationship

$$Y_k^L = X_{S(k)}^L \oplus Z_k^L, \quad \text{for } k = 1, \ldots, N, \tag{4.6}$$

where $\oplus$ indicates elementwise modulo 2 addition. Note that the N_i's are fully determined by the vector S^N since $N_i = |\{i\colon S(k) = i\}|$. Also note that, since $Q \sim \text{Ber}(1-q)$, $N \leq M$ with probability 1.

Consider a sequence of codes for the BSC shuffling-sampling channel with rate R and vanishing error probability. Let $X^{ML} = \left[X_1^L, \ldots, X_M^L\right]$ be the input to the channel when we choose one of the 2^{MLR} codewords from one such code uniformly at random, and $Y^{NL} = \left[Y_1^L, \ldots, Y_M^L\right]$ be the corresponding output. From Fano's inequality we have that $H(X^{ML}|Y^{ML}) \leq 1 + P_{e,M} ML \leq ML\epsilon_M$, where $P_{e,M}$ is the decoding error probability (of the code indexed by M) and $\{\epsilon_M\}$ is a sequence such that $\epsilon_M \to 0$ as $M \to \infty$. Thus,

$$MLR = H\left(X^{ML}\right) \leq I\left(X^{ML}; Y^{NL}\right) + ML\epsilon_M,$$

where $\epsilon_M \to 0$ as $M \to \infty$. Then,

$$\begin{aligned} ML(R - \epsilon_M) &= H\left(Y^{NL}\right) - H\left(Y^{NL}|X^{ML}\right) \\ &= H\left(Y^{NL}\right) - H\left(S^N, Z^{NL}, Y^{NL}|X^{ML}\right) \\ &\quad + H\left(S^N, Z^{NL}|X^{ML}, Y^{NL}\right) \\ &= H\left(Y^{NL}\right) - H\left(S^N, Z^{NL}, Y^{NL}|X^{ML}\right) \\ &\quad + H\left(S^N|X^{ML}, Y^{NL}\right) \end{aligned} \tag{4.7}$$

The last equality follows by noticing that, given (S^N, X^{ML}, Y^{NL}), one can compute $Z_k^L = Y_k^L \oplus X_{S(k)}^L$ for $1 \leq k \leq N$, and thus we have $H\left(Z^{NL}|X^{ML}, Y^{NL}, S^N\right) = 0$. Since N is a function of S^N, and S^N and Z^{NL} are independent of X^{ML}, the second term in (4.7) can be expanded as

$$
\begin{aligned}
&H\left(S^N, Z^{NL}, Y^{NL} | X^{ML}\right) \\
&= H\left(S^N | X^{ML}\right) + H\left(Z^{NL} | S^N, X^{ML}\right) + H\left(Y^{NL} | X^{ML}, S^N, Z^{NL}\right) \\
&\overset{(i)}{=} H\left(S^N, N\right) + H\left(Z^{NL} | S^N, N\right) + H\left(Y^{NL} | X^{ML}, S^N, Z^{NL}\right) \\
&\overset{(ii)}{=} H(N) + H\left(S^N | N\right) + H\left(Z^{NL} | N\right) \\
&\overset{(iii)}{=} H(N) + \sum_{n=1}^{M} \Pr(N = n) \left[\log \frac{M!}{(M-n)!} + nLH(p)\right] \\
&\overset{(iv)}{=} \sum_{n=1}^{M} \Pr(N = n) \left(n \log M + nLH(p)\right) + o(ML) \\
&= \mathbb{E}[N] M \left(\log M + LH(p)\right) + o(ML) \\
&= (1-q) \left[M \log M + MLH(p)\right] + o(ML). \qquad (4.8)
\end{aligned}
$$

In (i), we used the facts that S^N is independent of X^{ML}, N is a function of S^N, and Z^{NL} is independent of X^{ML} given S^N. Notice that X^{NL} is only dependent on S^N through N (which is a random variable). For (ii) we used that $H\left(Y^{NL} | X^{ML}, S^N, Z^{NL}\right) = 0$ since Y^{NL} is determined by X^{ML}, S^N, Z^{NL}, and (iii) follows from the fact that, given $N = n$, S^N is chosen uniformly at random from all vectors in $\{1, \ldots, M\}^n$ with distinct elements. For (iv), we used the fact that, from Stirling's approximation,

$$
\begin{aligned}
\log \frac{M!}{(M-n)!} &= M \log M - (M-n) \log(M-n) + o(ML) \\
&= M \log M - (M-n) \log M \\
&\quad + (M-n) \log \frac{M}{M-n} + o(ML) \\
&= n \log M + (M-n) \log \frac{M}{M-n} + o(ML),
\end{aligned}
$$

and, by Jensen's inequality,

$$
\begin{aligned}
0 &\leq \sum_{n>0} \Pr(N = n)(M-n) \log \frac{M}{M-n} \\
&\leq (M - \mathbb{E}[N]) \log \frac{M}{(M - \mathbb{E}[N])} \\
&= (1-q) M \log 1/q = o(ML).
\end{aligned}
$$

In order to finish the converse, we need to jointly bound the first and third terms in equation (4.7). This is summarized in a lemma.

Lemma 4.2. If β and $p < 1/4$ satisfy

$$1 - H(2p) - 2/\beta > 0, \tag{4.9}$$

then it holds that

$$H\left(Y^{NL}\right) + H\left(S^N | X^{ML}, Y^{NL}\right) \leq (1-q)ML + o(ML).$$

The parameter regime (p, β) for which (4.9) holds is the regime in which our capacity expression holds, illustrated in Figure 4.2. Combining (4.7), (4.8) and Lemma 4.2, we have

$$ML(R - \epsilon_M) \leq (1-q)\left(ML - MLH(p) - M\log M\right) + o(ML).$$

Dividing by ML and letting $M \to \infty$ yields the converse.

4.1.2 Intuition for Lemma 4.2

Rather than providing a full proof of Lemma 4.2, here we discuss the intuition behind it, and refer to Shomorony and Heckel [98] for the complete proof. To discuss the intuition for Lemma 4.2, let us focus on the case $q = 0$; i.e., none of the molecules are lost at the output. In this case, $N = M$, and S^M is chosen uniformly at random from all permutations of $[1, \ldots, M]$. If we naively bound each term separately, we obtain

$$H\left(Y^{ML}\right) + H\left(S^N | X^{ML}, Y^{ML}\right) \leq ML + M\log M.$$

However, intuitively, the bound $H\left(S^M | X^{ML}, Y^{ML}\right) \leq M\log M$ is too loose because, as we argue below, if the entropy term $H\left(Y^{ML}\right)$ is large then we expect $H\left(S^M | X^{ML}, Y^{NL}\right)$ to be small and vice versa.

To see this, first note that given $X^{ML} = x^{ML}$ and $Y^{ML} = y^{ML}$, one can estimate the permutation S that maps each output string to the corresponding input string, S^M, by finding, for each y_i^L, the x_j^L that is closest to it and setting $S(i) = j$. This is a good estimate if no other x_k^L is close to x_j^L. There are two regimes, illustrated in Figure 4.3, one where

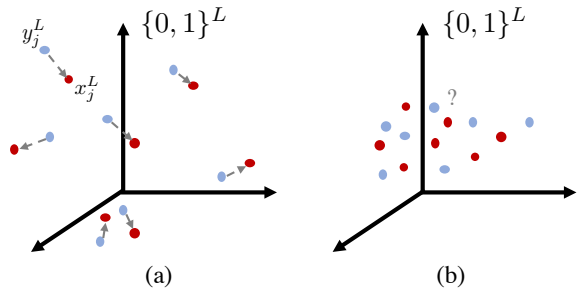

Figure 4.3: Two opposite scenarios for estimating S^N from $\left(X^{ML}, Y^{NL}\right)$.

S^N can be estimated well and one where it cannot. In the first regime, the strings $x_1^L, \ldots, x_M^L$ are all sufficiently distant from each other (in the Hamming sense). Hence, the maximum likelihood estimate of S^N given $X^{ML} = x^{ML}$ and $Y^{NL} = y^{ML}$ is "close" to the truth and we expect $H\left(S^N | X^{ML} = x^{ML}, Y^{NL} = y^{ML}\right)$ to be small. In the second regime, illustrated in Figure 4.3(b), many of the sequences $x_1^L, \ldots, x_M^L$ are close to each other. So we have less information about S^N, and $H\left(S^N | X^{ML} = x^{ML}, Y^{NL} = y^{ML}\right)$ may be large.

On the other hand, the term $H\left(Y^{NL}\right)$ is maximized if the sequences $\{X_i^L\}$ are independent and if their values are uniformly distributed in $\{0,1\}^L$. Hence, in order for $H\left(Y^{NL}\right)$ to be large, we expect to be in the regime in Figure 4.3(a) instead of the regime of Figure 4.3(b). This leads to a tradeoff of the terms $H\left(Y^{NL}\right)$ and $H\left(S^N | X^{ML}, Y^{NL}\right)$, which we exploit to prove Lemma 4.2.

4.2 Different noise models

Given that the capacity expression for the noisy shuffling-sampling channel given in Theorem 4.1 is $(1-q)(C_{\text{BSC}} - 1/\beta)$, where $C_{\text{BSC}} = 1 - H(p)$ is the capacity of a BSC, it is natural to ask whether for a different noisy channel with capacity C_{noisy}, the corresponding noisy shuffling-sampling channel has capacity $(1-q)(C_{\text{noisy}} - 1/\beta)$. Notice that, when the sampling distribution Q is Bernoulli$(1-q)$, the index-based achievability scheme described in Section 4.1 can be extended in a straightforward way to achieve any rate below $(1-q)(C_{\text{noisy}} - 1/\beta)$.

Based on this, it is natural to conjecture that the capacity of a noisy shuffling-sampling channel with sampling distribution Bernoulli$(1-q)$ and noisy channel with capacity C_{noisy} is given by

$$(1-q)(C_{\text{noisy}} - 1/\beta)^+. \tag{4.10}$$

In Section 7, we present an even more general conjecture than this one. Since achieving the rate in (4.10) is straightforward, the challenging technical question is whether the converse argument in Section 4.1.1 can be generalized. Next we discuss specific noisy channel cases where this conjecture can be proved.

4.2.1 BEC Shuffling-Sampling Channel

Consider the setting studied in Section 4.1 except that, instead of a BSC, we have a binary erasure channel (BEC) with erasure probability p. While erasures are not observed in sequencing data in practice, they can be practically motivated by the fact that sequencing technologies often provide a quality score for each base [13]. Such score can in principle be thresholded to identify reliable base calls and unreliable ones, which could then be treated as erasures.

As shown in Shin, Heckel, and Shomorony [96], ideas similar to those used in the converse in the BSC case in Section 4.1.1 can be used in the BEC case as well. While the result in Shin, Heckel, and Shomorony [96] is stated for the case of perfect sampling ($q_1 = 1$), it can be generalized to the case of Bernoulli$(1-q)$ sampling using the ideas from Section 4.1.1, yielding the following result.

Theorem 4.3. The capacity of the BEC shuffling-sampling channel is

$$C = (1-q)(1-p-1/\beta), \tag{4.11}$$

as long as $1 - 2p - 2/\beta > 0$. If $\beta \leq 1$, then the capacity is $C = 0$.

Notice that, similar to the BSC case, the theorem only holds for a specific regime of p and β. However, the simplicity of the the erasure setting allows us to establish the converse for a larger set of parameters. In particular, while Theorem 4.1 holds for $p < 1/4$ and $1 - H(2p) - 1/\beta > 0$, Theorem 4.3 holds for the strictly larger regime $p < 1/2$ and $1 - 2p - 1/\beta > 0$.

4.2.2 General Noisy Channels

Given that (4.10) holds (for some parameter regime) for both the BSC and the BEC, it is natural to attempt to generalize the result to more general discrete memoryless channels. To understand the challenges of this general setting, consider the simpler case of ideal sampling, i.e., $N_i = 1$ with probability 1 for $i = 1, ..., M$. Suppose we have an arbitrary, not necessarily memoryless, channel $p(y^L|x^L)$ that maps length-L input strings to length-L output strings (which may not be memoryless) and capacity C_{noisy} (which requires the channel to be defined for $L \to \infty$). Consider the corresponding noisy shuffling channel. The index-based scheme achieves any rate $R < C_{\text{noisy}} - 1/\beta$. However, extending the proof in Section 4.1.1 to establish $C_{\text{noisy}} - 1/\beta$ as the capacity is challenging. Following similar steps to those in (4.7),

$$\begin{aligned} ML(R - \epsilon_M) &= I\left(X^{ML}; Y^{ML}\right) \\ &= H\left(Y^{NL}\right) - H\left(S^M, Y^{ML}|X^{ML}\right) + H\left(S^M|X^{ML}, Y^{ML}\right) \\ &= H\left(Y^{NL}\right) - H\left(Y^{ML}|X^{ML}, S^M\right) - H\left(S^M\right) + H\left(S^M|X^{ML}, Y^{ML}\right) \\ &= I\left(X^{ML}, S^M; Y^{ML}\right) - H\left(S^M\right) + H\left(S^M|X^{ML}, Y^{ML}\right). \end{aligned}$$

Since $H(S^M) = M \log M + o(ML) = ML/\beta + o(ML)$, an outer bound to the noisy shuffling channel capacity in this general case is

$$C \leq \lim_{M\to\infty} \sup_{p(x^{ML})} \frac{I\left(X^{ML}, S^M; Y^{ML}\right)}{ML} + \frac{H\left(S^M|X^{ML}, Y^{ML}\right)}{ML} - 1/\beta. \tag{4.12}$$

The main challenge in establishing a general converse is the optimization over distributions of the channel input X^{ML}. Intuitively, for "well-behaved" channels, choosing the input distribution $p(x^{ML})$ that maximizes the first term, $I\left(X^{ML}, S^M; Y^{ML}\right)$, causes the output strings Y_i^L, $i = 1, \ldots, M$, to be spread out in the output space, making the second term, $H\left(S^M|X^{ML}, Y^{ML}\right)$, small. In Section 4.1.1, we explained how the tension between these two terms can be exploited to jointly bound them. The same idea is used in Shin, Heckel, and Shomorony [96] in the case of the BEC. In both cases, this joint bounding allows

us to show that (4.12) is optimized by choosing $p(x^{ML})$ to be ML i.i.d. Bernoulli(1/2) random variables.

Notice that, if we know that the optimal distribution in (4.12) satisfies $p(x^{ML}) = p(x^L) \times \cdots \times p(x^L)$ (i.e., independently encoding each of the input strings with the same $p(x^L)$), then the first term in the optimization becomes

$$\begin{aligned}\frac{I\left(X^{ML}, S^M; Y^{ML}\right)}{ML} &= \frac{H\left(Y^{ML}\right) - \left(Y^{ML}|X^{ML}, S^M\right)}{ML} \\ &= \frac{\sum_{i=1}^{M} H\left(Y_i^L\right) - \left(Y_i^L|X_{S(i)}^L\right)}{ML} \\ &= \frac{I(X^L; Y^L)}{L},\end{aligned}$$

and by choosing the distribution $p(x^L)$ that achieves the capacity of the noisy channel $p(y^L|x^L)$, this term becomes C_{noisy}. However, establishing that this is the optimal input distribution for general channels remains an open question. We refer to Section 5.1.4 for additional discussion on extensions to general discrete memoryless channels, but in the multi-draw setting.

4.3 Index-based coding and independent decoding

All achievability schemes discussed in Sections 3 and 4 are based on the same approach, which is illustrated in Figure 4.1. In particular, we highlight two key elements of the achievability schemes:

1. *Index-based coding:* Distinct indices are placed in each stored sequence (prior to the encoding with an error-correcting code). At the decoder, the indices are used to order the input sequences and to identify missing ones.

2. *Independent decoding:* Each of the stored sequences is a codeword from an error-correcting code, and is independently decoded at the output (possibly followed by the decoding of an outer code).

Notice that both index-based coding and independent decoding are highly desirable from a computational standpoint. However, from a capacity standpoint it is surprising that optimal codes exhibit these desirable features.

Is index-based coding always optimal? A key question that naturally arises from the results in Sections 3 and 4 is whether index-based coding is always optimal. In the parameter regime in the gray region of Figure 3.5, the capacity remains unknown and it is possible to achieve rates higher than the $(1-q)(1-H(p)-1/\beta)$ that is achieved by index-based schemes. Notice that, this region corresponds to a high-noise short-molecule regime, where rates are expected to be low. In a short-molecule regime, it may be reasonable that dedicating a significant fraction of each molecule for an index is wasteful from a data rate standpoint. Furthermore, in the high-noise regime, a significant amount of redundancy needs to be added to guarantee that the indices are decoded error-free. Hence, it may be possible to devise a scheme that uses less redundancy for indices, but the decoder only decodes the indices approximately, which may be enough to order the sequences once all indices are considered jointly. Furthermore, as we will show in the next section, once we move away from the single-draw setting, and allow multiple copies of each input sequence to be observed at the output with independent noise patterns, index-based schemes are no longer optimal, and "global" decoding schemes are needed to achieve the noisy shuffling-sampling channel capacity.

A similar question can be posed regarding the optimality of independent decoding. Such a question was in fact studied in detail in the context of the *bee identification problem* [111].

4.3.1 Independent decoding and the bee identification problem

The bee-identification problem is motivated by the task where one observes a noisy picture of a massive number of bees, uniquely labeled with barcodes, and wishes to simultaneously identify all bees. This can be modeled as a problem in which a single list of M strings of length L, $[x_1^L, ..., x_M^L]$, is passed through a noisy shuffling channel (Tandon,

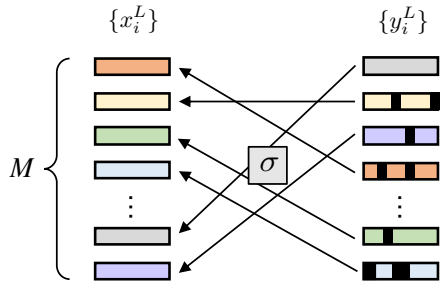

Figure 4.4: In the bee identification problem, we observe a set of output sequences $\{y_1^L, ..., y_M^L\}$ and want to find a mapping σ such that $x_{\sigma(i)}^L$ was the input sequence that resulted in y_i^L after the noisy shuffling channel.

Tan, and Varshney [111] consider the BSC shuffling channel described in Section 4.1). The goal is to find the correct matching σ between the output sequences $[y_1^L, ..., y_M^L]$ and the input sequences; i.e., finding the shuffling induced by the channel on the set of strings, as illustrated in Figure 4.4.

A key contribution of Tandon, Tan, and Varshney [111] is to find upper and lower bounds for the error exponent of the bee identification problem under two decoding approaches: independent decoding and joint decoding. In independent decoding, only the ith output sequence y_i^L is used to determine $\sigma(i)$, while in joint decoding, the entire set of output sequences $\{y_1^L, ..., y_M^L\}$ is used to determine σ. They show that the error exponent for joint decoding is strictly larger than the error exponent for independent decoding. They also compare the error exponents of random code ensembles and typical random codes and show that typical random codes provide a significant improvement. Since then, follow-up works have refined these error exponents, considered different scenarios in terms of what constitutes an error when recovering σ and whether all bees are observed at the output [110], [112], and have proposed efficient ways to perform the joint decoding [53].

In the context of DNA-based storage, the goal is not to recover the permutation σ, but rather to decode the message encoded by the shuffled set of sequences. However, all capacity-achieving schemes discussed in

Sections 3 and 4 place unique indices on every sequence, with the goal of allowing the decoder to recover the correct permutation σ (which will then allow the message to be recovered). Hence, the results from Tandon, Tan, and Varshney [111] suggest that, also in the context of DNA storage, joint decoding can improve the error exponent with respect to independent decoding.

Nevertheless, it is shown in Tandon, Tan, and Varshney [112] that, in the setting with absentee bees (i.e., when some bees are not observed at the output), the advantage of joint decoding with respect to independent decoding disappears, and the error exponents achieved are the same. This suggests that the advantage of joint decoding may depend on the unrealistic assumption that all sequences are observed at the output. As discussed throughout this monograph, this is not realistic based on current DNA data storage prototypes, suggesting that the potential gains of joint decoding may be minor, particularly given the significant computational costs of joint decoding.

4.4 Designing practical codes for DNA data storage

In this section, we investigated the fundamental limits of noisy shuffling channels. While we considered standard noisy channels such as the BSC and the BEC, the design of practical codes for DNA storage systems must be tailored to the technologies being employed. Several recent works have focused on designing error-correcting codes for the specific types of errors that may arise during writing, storing and reading data on DNA. Some important aspects addressed in the recent literature include DNA synthesis constraints such as sequence composition [33], [51], [56], [119], the asymmetric nature of the DNA sequencing error channel [34], the need for codes that correct insertions and deletions [19], [26], [52], [57], [92], the need for codes that avoid homopolymers (consecutive repeated nucleotides) as they are conducive to sequencing errors [33], [51], [56], and the need for codes with balanced GC-content (i.e., roughly the same number of G/C as A/T) [20]. An LDPC-based joint design for the inner and outer codes needed for DNA data storage was proposed in Chandak, Tatwawadi, Lau, *et al.* [17].

The shuffling aspect of DNA storage systems has also been addressed from a coding-theoretic perspective. In particular, the problem of coding over sets (or multi-sets) has been studied [58], [67], [100]–[102], [104], [105], [115], with the goal of characterizing basic properties such as minimum distance and maximum size of codes in the space of multi-sets, and with the goal of providing bounds on code parameters. We point out that the shuffling channel is also connected with a *permutation channel* that permutes the symbols in a message [3], [10], which recently received renewed attention due to the emergence of DNA-based data storage [73].

The problem of designing indices that allow efficient retrieval of data stored on DNA has also been investigated [67]. In particular, Lenz, Siegel, Wachter-Zeh, *et al.* [66] explores the idea of adding "anchor sequences" to the indices, with the goal of reducing the total amount of redundancy needed for the indices. Furthermore, the careful design of indices that allow *random access* via DNA *hybridization* has also been explored [20], [80], [119], [120].

5

Multi-draw Channels: Clustering Output Sequences

In Section 4, we studied noisy shuffling-sampling channels with single draws. More precisely, each sequence in the DNA library can either be drawn once or not at all. However, as shown in Figure 2.2, a more complete model for DNA-based storage systems should incorporate the fact that each DNA sequence can be sequenced multiple times. This multi-draw setting arises because synthesis technologies generate multiple copies in the first place, and PCR used at the time of sequencing multiplies the number of copies further, which results in a large number of copies of each DNA molecule at the output. As illustrated in Figure 5.1, at the output of the DNA storage channel, one may thus observe several different noisy copies of each of the input sequences, and a "clustering problem" arises from the need to identify which of these sequences correspond to the same input sequence.

In the multi-draw setting, establishing the capacity of noisy shuffling-sampling channels is significantly more challenging. Intuitively, the presence of multiple noisy copies of the same string should increase the capacity. More precisely, if we are able to correctly cluster the output sequences, it is possible to combine the sequences in each cluster and perform error correction, effectively reducing the error rate of the

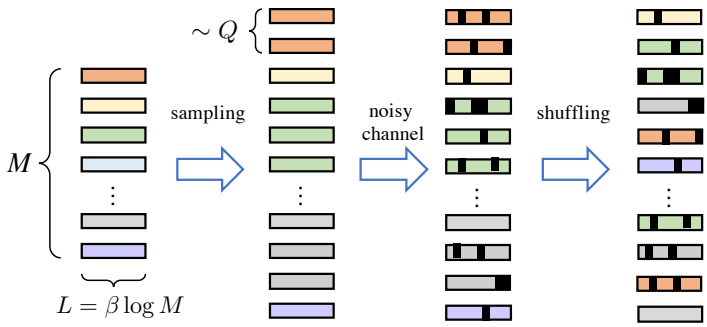

Figure 5.1: When the sampling distribution Q (see Figure 2.2) allows more than one copy of each input sequence to be created (e.g., Q is Poisson(λ)), the output sequences can intuitively be clustered according to the input sequence of origin.

system. The following questions arise. Is it still optimal to utilize an index-based scheme? Is it information-theoretically optimal to first cluster the output sequences and then attempt to decode the message?

We begin our discussion on this problem by focusing on the setting where each sequence is passed through a Binary Erasure Channel (BEC), in Section 5.1. It turns out that erasures are easier to treat than subsitutions or insertions and deletions, and this simplicity enables us to gain basic insights into the nature of the clustering problem that arises in the multi-draw context. In addition, we discuss connections with the trace reconstruction problem in Section 5.2, and coding-theoretic considerations in Section 5.3.

5.1 Noisy shuffling-sampling channel with multi-draws

As a first step to studying the capacity of a noisy shuffling-sampling channel with multi-draws, we consider a noise model described by a binary erasure channel (BEC), studied by Levick, Heckel, and Shomorony [70]. As it turns out, the simplicity of the BEC (relative to the binary symmetric channel (BSC)) allows us to utilize a linear coding scheme to achieve the capacity of the system. Moreover, the achievability proofs are intuitive and based on simple equation-counting arguments.

We consider the general DNA storage channel illustrated in Figure 2.2, where the sampling distribution Q is arbitrary, and the memory-

less channel $p(y|x)$ is a BEC, i.e., each symbol is erased with probability p (leading to an erasure symbol ε). Recall that the capacity of the BEC is $1-p$ and, as discussed in Section 4.2.1, the capacity of the BEC shuffling-sampling channel with single draws (with a probability q of not seeing a sequence) is given by

$$C_{\text{BEC, single-draw}} = (1-q)(1-p-1/\beta). \tag{5.1}$$

Now consider instead an arbitrary sampling distribution Q, with $q_n = \Pr(N_i = n)$. Notice that, in expectation, a fraction q_0 of the input sequences is not sampled, and thus we expect a total of $(1-q_0)M$ output clusters.

Let us first suppose we have a genie-aided channel that provides us with the correct clustering of the output strings. In this case, it is intuitive that the optimal decoding scheme starts by combining all the sequences in each cluster into a single "consensus" sequence. The consensus sequence (of length L) is obtained by taking, for the ith position, the ith symbol of any sequence in the cluster that is not erased. For an output cluster with n sequences, the resulting effective erasure rate is p^n. Therefore, after the consensus step, in expectation we have $(1-q_0)M$ output sequences and, for $n = 1, 2, \ldots$, we expect that $q_n M$ of them have an erasure rate of p^n.

Notice that the resulting effective channel after the consensus step is similar to the BEC shuffling-sampling channel discussed in Section 4.2.1 but, instead of a single erasure probability p across all sequences drawn at the output, the $(1-q_0)M$ output strings experience an average erasure probability of

$$p_{\text{eff}} \coloneqq E[p^{N_1}|N_1 \geq 1] = \frac{\sum_{n=1}^{\infty} q_n p^n}{1-q_0} = \frac{\sum_{n=1}^{\infty} q_n p^n}{\sum_{n=1}^{\infty} q_n} \tag{5.2}$$

Since the capacity of this genie-aided channel (which has $(1-q_0)M$ output sequences) is larger than the capacity of the BEC shuffling-sampling channel, we can use the converse approach from Section 4.1.1 (see also Shin, Heckel, and Shomorony [96]) to show that the capacity of the BEC shuffling-sampling channel satisfies

$$C_{\text{BEC, multi-draw}} \leq (1-q_0)\,(1-p_{\text{eff}}-1/\beta)\,, \tag{5.3}$$

as long as p and β satisfy $1 - 2p - 2/\beta > 0$. We point out that, when the converse approach from Section 4.1.1 is applied in this multi-draw context, the condition $1 - 2p - 2/\beta > 0$ depends on p, and not p_{eff}, but a more careful analysis may be able to relax this condition.

The main result in this section shows that, for β large enough, this upper bound can be achieved.

Theorem 5.1. The capacity of the BEC shuffling-sampling channel is

$$C_{\text{BEC, multi-draw}} = (1 - q_0)\,(1 - p_{\text{eff}} - 1/\beta)\,, \tag{5.4}$$

as long as $1 - 2p - 2/\beta > 0$.

5.1.1 Achievability via linear schemes

Unlike in Sections 3 and 4, to achieve the capacity in this multi-draw setting, we do not utilize an index-based scheme. Instead, we take advantage of the fact that random linear codes are known to achieve the capacity of the BEC [32], and show that linear schemes also achieve the capacity of the BEC shuffling-sampling channel.

We construct our linear coding scheme with rate R as follows. For a parameter B to be determined, we first generate a random binary $ML \times B$ matrix $\mathbf{G}$ with i.i.d. Bernoulli(1/2) entries. We then generate 2^{MLR} random binary vectors $\mathbf{t}_i$ of size B (also with i.i.d. Bernoulli(1/2) entries). For $i = 1, \ldots, 2^{MLR}$, the ith codeword is generated by computing $\mathbf{G}\,\mathbf{t}_i$ (over $\mathbb{F}_2$), and then breaking the resulting length-ML vector into M binary strings of length L (which serve as channel input). We point out that this is technically not a *linear code* as the set of codewords is not the entire range of $\mathbf{G}$ (only the vectors $\mathbf{G}\mathbf{t}_i$ for $\mathbf{t}_1, \ldots, \mathbf{t}_{2^{MLR}}$) and hence the code is not a linear subspace of $\{0,1\}^{ML}$.

A key property of $\mathbf{G}$ that we need is the following.

Lemma 5.2. Let $\mathbf{G}$ be an $ML \times B$ matrix with i.i.d. Bernoulli(1/2) entries. Fix any $\delta \in (0,1)$ and a submatrix $\mathbf{G}'$ formed by an arbitrary set of $(1-\delta)B$ rows of $\mathbf{G}$. Then $\mathbf{G}'$ is full rank (over the finite field $\mathbb{F}_2$) with probability tending to 1 as $B \to \infty$.

While a version of this lemma for a random matrix over the reals is relatively straightforward, for the case of $\mathbb{F}_2$ it requires some work to prove. We present the proof of Lemma 5.2 in Section 5.4.

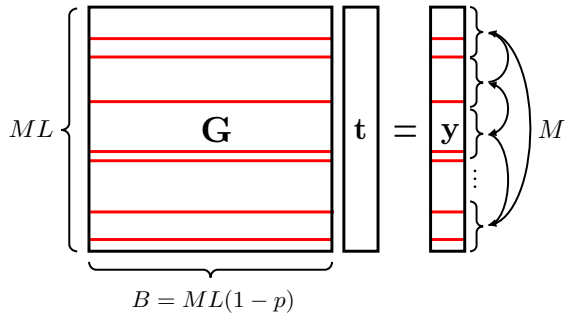

Figure 5.2: In the setting where each sequence is observed exactly once at the output, there are $M!$ potential systems of equations, each with $ML(1-p)$ non-erased equations. Choosing our rate to be approximately $1-p-1/\beta$ guarantees that, with high probability, only one of these systems of equations will have a solution that corresponds to a codeword.

Single-draw case

To illustrate this linear non-index-based scheme, let us first consider the case where each sequence is sampled exactly once (i.e., $q_1 = 1$). At the output, we observe M sequences, which we would like to use for recovering the vector $\mathbf{t}_i$, or simply for recovering the message index i.

Suppose we knew the correct ordering of the M output strings. Then we could concatenate them into a single string $\mathbf{y}$ with length ML, and then try to solve the system $\mathbf{G}\,\mathbf{t} = \mathbf{y}$. Since a fraction p of the entries of $\mathbf{y}$ would be erased, we would be able to solve this system (with high probability) as long as the number of remaining equations, which is roughly $(1-p)ML$, is greater than or equal to the number of variables B. We would then be able to set $B = (1-p-\epsilon)ML$ for any $\epsilon > 0$. We would also be able to choose all 2^B binary strings in $\{0,1\}^B$ to be $\mathbf{t}_1, \ldots, \mathbf{t}_{2^B}$, instead of choosing them randomly, which results in a code with rate of $R = 1-p-\epsilon$, which is the capacity of a standard BEC.

However, since in actuality we do not know the ordering of the M output strings, we instead consider all $M!$ possible orderings of the output strings. Each such ordering gives us a distinct concatenated vector $\mathbf{y}$ with a p fraction of erasures and a corresponding set of useless rows in the matrix $\mathbf{G}$, as illustrated in Figure 5.2. To guarantee that the

matrix $\mathbf{G}$ with erased rows is still invertible, we set $B = ML(1-p-\epsilon)$ for some small $\epsilon > 0$, and Lemma 5.2 guarantees that the true system has a unique solution $\mathbf{t}$. Moreover, to guarantee correct decoding, we must ensure that only one of the $M!$ systems (the true one) admits a solution $\mathbf{t}$ that is one of the original vectors $\mathbf{t}_i$ for $i = 1, \ldots, 2^{MLR}$.

Assume without loss of generality that message 1 is sent (i.e., the codeword defined by $\mathbf{G}\,\mathbf{t}_1$). Since all other vectors $\mathbf{t}_i$, $i \neq 1$, are chosen uniformly at random from $\{0,1\}^B$, by the union bound, the probability that one of the $M! - 1$ incorrect systems has a solution $\mathbf{t}$ that coincides with some $\mathbf{t}_i$, $i = 2, \ldots, 2^{MLR}$, is at most

$$(M!-1)2^{MLR}2^{-B} < 2^{M\log M + MLR - B} = 2^{ML[R-(1-p-\epsilon-1/\beta)]}.$$

By choosing ϵ arbitrarily small, we conclude that any rate $R < 1-p-1/\beta$ can be achieved with a vanishingly small error probability.

While this result simply recovers what the index-based approach in Section 4 already achieved, it can be extended to the case of a general sampling distribution Q as we see next.

Multi-draw case

Let us now consider the case of a general sampling distribution Q. At the output of the channel, we expect to see roughly $E[N_1]M$ sequences, which we would like to partition into roughly $(1-q_0)M$ clusters. A major advantage of the BEC setting is that it easier to reason about the clustering task than for the BSC setting, for example. Notice that all output strings originated from the same input string are *consistent* with each other; i.e., they agree on any non-erased positions. Hence, given N output strings $x_1^L, \ldots, x_N^L$, one can in principle build a graph with $x_1^L, \ldots, x_N^L$ as vertices, where strings x_i^L and x_j^L are connected by an edge if they are consistent. We refer to this graph as the *consistency graph*. A valid clustering of the output strings corresponds to any partition of $\{x_1^L, \ldots, x_N^L\}$ such that each group corresponds to a *clique* in the consistency graph.

Since each input string has a probability q_0 of not producing an output cluster, in expectation we have $(1-q_0)M$ clusters. Moreover, an application of Hoeffding's inequality implies that the probability

that there are more than $(1-q_0+\epsilon)M$ or less than $(1-q_0-\epsilon)M$ true clusters at the output is bounded as

$$\Pr\left(|\#\text{ true output clusters} - (1-q_0)M| > \epsilon M\right) < 2e^{-2M\epsilon^2},$$

which tends to 0 as $M \to \infty$ for any $\epsilon > 0$. Therefore, our decoder looks for ways cluster the output strings into at least $(1-q_0-\epsilon)M$ clusters and at most $(1-q_0+\epsilon)M$ clusters, for some fixed small ϵ. More precisely, our decoder considers *all* valid clusterings of the N output strings into at most $(1-q_0+\epsilon)M$ and at least $(1-q_0-\epsilon)M$ clusters. For each such clustering, a consensus step is performed, effectively converting the clusters into roughly $(1-q_0)M$ strings with a smaller overall erasure rate. After this point, the decoder proceeds similarly to the case of single draws. First, each cluster is assigned a distinct index from $\{1,\ldots,M\}$, which can be used to order the consensus strings into a vector $\mathbf{y}$ of length roughly $(1-q_0)ML$. All possible index assignments are considered. Each index assignment produces a vector $\mathbf{y}$ and a corresponding system of equations, which may yield a solution $\mathbf{t}$ (provided that the system has a solution). This clustering-based decoding is illustrated in Figure 5.3.

Notice that we need to choose B large enough to guarantee that the true system, obtained from the correct clustering and index assignment, has a unique solution, and R small enough so that only one of the systems (the true one) yields a solution that coincides with one of the vectors $\mathbf{t}_i$ used to construct the codebook. This allows the correct decoding of the message index.

In order to guarantee that the correct $\mathbf{t}_i$ can be recovered, we need to make sure that the true system has enough equations after the erasures have been discarded. Once the consensus step is performed on the true clustering, we end up with at least $M(1-q_0-\epsilon)$ sequences. Moreover, the expected erasure rate is given by p_{eff} in (5.2), and standard concentration inequalities can be used to show that it cannot deviate significantly from that. Hence, the true system of equations has at least $ML(1-q_0-\epsilon)(1-p_{\text{eff}}-\epsilon)$ equations with high probability, and if we set $B = ML(1-q_0-\epsilon)(1-p_{\text{eff}}-\epsilon)(1-\epsilon)$, by Lemma 5.2, the true system of equations has a unique solution with probability tending to 1 as $M \to \infty$, which yields the correct solution $\mathbf{t}_1$.

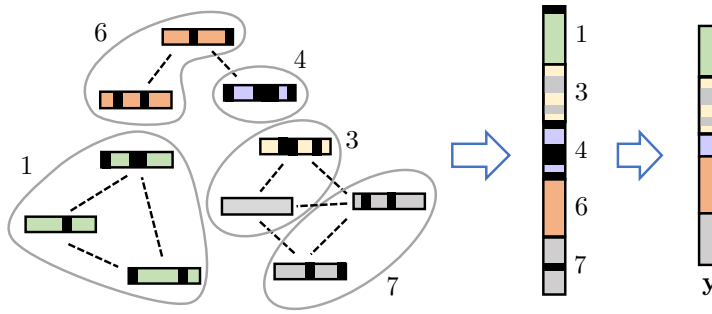

Figure 5.3: The decoder first builds a consistency graph between all received sequences, and considers all possible partitions of the graph into cliques, where the total number of cliques is between $(1 - q_0 - \epsilon)M$ and $(1 - q_0 + \epsilon)M$. For each such valid clustering, the decoder considers all possible assignments of the indices $\{1, ..., M\}$ to the clusters and uses those indices to order the consensus sequences of each cluster to form the vector $\mathbf{y}$. After removing erased entries and the rows of $\mathbf{G}$ corresponding to erased/missing rows in $\mathbf{y}$, the system $\mathbf{Gt} = \mathbf{y}$ can be solved.

In order to find the maximum rate R for which this scheme succeeds with vanishing error probability, we need to bound the number of valid clusterings of the output strings. To that end, we first analyze the total number of edges in the consistency graph. Notice that, each true cluster of size n produces $\binom{n}{2} \leq n^2/2$ "correct" edges in the graph. Hence, the expected number of correct edges is at most

$$\sum_{n=0}^{\infty} M q_n \frac{n^2}{2} = \frac{M}{2} E[N_1^2], \tag{5.5}$$

where $E[N_1^2]$ is the second moment of the sampling distribution Q. Let Z be the total number of incorrect edges in the consistency graph, and

$$\gamma := -\beta \log \left(1 - \tfrac{1}{2}(1-p)^2\right),$$

which is positive for any $p \in (0, 1)$.

Lemma 5.3. The number of incorrect edges Z satisfies

$$\Pr\left(Z > M^{2-\gamma+\epsilon}\right) \to 0 \tag{5.6}$$

as $M \to \infty$, for any $\epsilon > 0$.

Lemma 5.3 (whose proof is presented in Section 5.4) establishes that, as long as $\gamma > 1$, the number of incorrect edges grows slower than

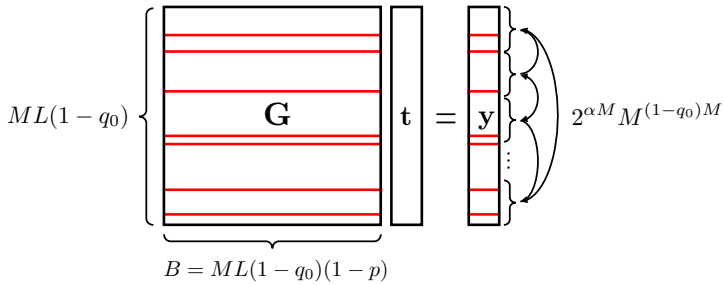

Figure 5.4: In the multi-draw setting, there are $2^{\alpha M} M^{(1-q_0)M}$ potential systems of equations, obtained by considering all valid ways to cluster the output strings into $(1-q_0)M$ clusters, and then assigning each cluster to an index. The true system (corresponding to the correct clustering and ordering) has $ML(1-q_0)(1-p_{\text{eff}})$ non-erased equations in expectation.

M and, hence, will be vanishingly small compared to the number of correct edges in (5.5). Next, we bound the number of valid ways to cluster the output strings given the number of edges in a consistency graph. As it turns out, a coarse bound suffices.

Lemma 5.4. Suppose the consistency graph has a total of U edges. Then there are at most 2^U valid ways to cluster the output sequences.

Proof. Each valid clustering corresponds to a partition of the output sequences such that each group corresponds to a clique in the consistency graph. Notice that a partition of the consistency graph into cliques is uniquely described by the set of edges that are part of each of the cliques. Hence, each partition corresponds to a distinct subset of the U edges in the graph. Since there are at most 2^U such subsets, it follows that there are at most 2^U valid ways to cluster the output sequences. □

Now from equation (5.5) and Lemma 5.3, we know that for $\gamma > 1$, the number of edges U in the consistency graph satisfies $U < \alpha M$ for some $\alpha > 0$, with high probability. Lemma 5.4 thus implies that the number of valid ways to cluster the output strings is at most $2^{\alpha M}$.

The achievability proof is concluded by following the steps from the single-draw case. We need to guarantee that no system of equations other than the true one yields $\mathbf{t}_i$, for some $i = 2, \ldots, 2^{MLR}$, as a solution.

The total number of systems of equations that the decoder will attempt to solve is upper-bounded by

$$2^{\alpha M} M^{(1-q_0+\epsilon)M}, \tag{5.7}$$

because we first need to cluster all the output strings into at most $(1-q_0+\epsilon)M$ clusters in a valid way and then assign each of them an index in $\{1,\dots,M\}$ for the ordering. This is illustrated in Figure 5.4. Therefore, if we assume that message 1 is selected, the probability that any of the incorrect systems yields a solution $\mathbf{t}$ that coincides with some $\mathbf{t}_i$, for $i=2,\dots,2^{MLR}$, is upper-bounded by

$$\begin{aligned} 2^{\alpha M} M^{(1-q_0+\epsilon)M} 2^{MLR} 2^{-B} &= 2^{\alpha M+(1-q_0+\epsilon)M\log M+MLR-B} \\ &= 2^{ML(\alpha/L+(1-q_0+\epsilon)/\beta+R-B/ML)}. \end{aligned} \tag{5.8}$$

This upper bound goes to zero as $M\to\infty$ as long as

$$R+\frac{\alpha}{\beta\log M}-(1-q_0-\epsilon)(1-p_{\text{eff}}-\epsilon)(1-\epsilon)+\frac{1-q_0+\epsilon}{\beta}<0.$$

Since $\alpha/\log M\to 0$ and ϵ can be chosen arbitrarily small, any rate

$$R<(1-q_0)(1-p_{\text{eff}}-1/\beta)$$

is achievable. This is formally stated next.

Theorem 5.5. The capacity of the BEC shuffling-sampling channel with multi-draws satisfies

$$C_{\text{BEC, multi-draw}}\geq(1-q_0)\,(1-p_{\text{eff}}-1/\beta)\,, \tag{5.9}$$

as long as $\gamma=-\beta\log\left(1-\frac{1}{2}(1-p)^2\right)>1$. Here, p_{eff} is the effective erasure probability defined in equation (5.2).

In Figure 5.5, we compare the parameter regime in which the lower bound in Theorem 5.5 holds with the regime in which the upper bound (5.3) holds. We see that the lower bound constraint is strictly weaker than the upper bound constraint, which implies Theorem 5.1.

We conclude by noticing that, for natural choices of the sampling distribution Q, the effective erasure probability p_{eff} can be computed in closed-form. For example, if Q is Poisson(λ), then

$$p_{\text{eff}}=\frac{\sum_{n=1}^{\infty}q_n p^n}{1-q_0}=\frac{E[p^n]-q_0}{1-q_0}=\frac{E[e^{n\ln p}]-q_0}{1-q_0}=\frac{e^{-\lambda(1-p)}-e^{-\lambda}}{1-e^{-\lambda}},$$

leading to the capacity expression

$$C_{\text{BEC, Poisson}(\lambda)} = (1 - e^{-\lambda}) \left(1 - \frac{e^{-\lambda(1-p)} - e^{-\lambda}}{1 - e^{-\lambda}} - \frac{1}{\beta}\right)^+$$
$$= \left(1 - e^{-\lambda(1-p)} - \frac{1 - e^{-\lambda}}{\beta}\right)^+, \tag{5.10}$$

which holds in the blue region of Figure 5.5, and which provides an achievable region (and possibly the capacity) in the green region.

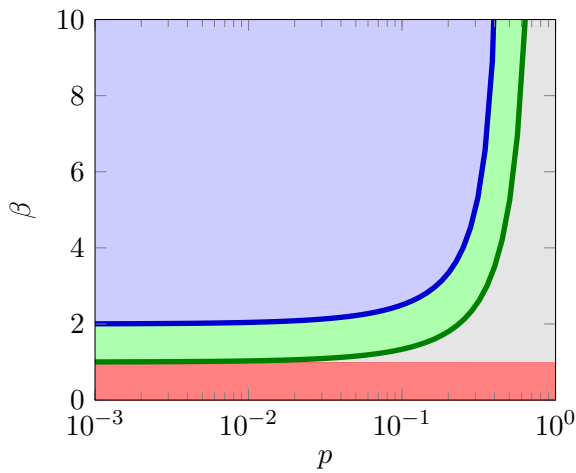

Figure 5.5: The outer bound from (5.3) holds in the blue region, which corresponds to $\beta > 1/(1/2 - p)$. The inner bound from Theorem 5.5 holds above the green line, which corresponds to $\beta > -1/\log(1 - \frac{1}{2}(1-p)^2)$. This characterizes the capacity of the BEC shuffling channel in the blue region as $(1 - q_0)(1 - p_{\text{eff}} - 1/\beta)$. The capacity in the red region (i.e., for $\beta < 1$) is 0 and it is unknown in the gray region.

5.1.2 Is performing clustering and decoding separately optimal?

From a computational efficiency standpoint, it is natural to separate the clustering and decoding tasks. Under this approach, one could use standard clustering algorithms to determine which output sequences originate from the same input sequence. This could then be followed by a consensus step (i.e., a cluster-based error correction), and finally

a decoding of the sequences back to the original message. However, it may be suboptimal to perform the clustering and consensus tasks in a "code-oblivious" way; i.e., without taking advantage of the underlying codebook.

Notice that, while the coding scheme presented in Section 5.1.1 performs the clustering, consensus, and decoding tasks in sequence, the clustering task is not oblivious to the code. This is because the scheme considers *all* valid clustering partitions and, for each one, attempts to decode the message. Therefore, in an indirect way, the code is helping with clustering. In particular, we point out that in the green regime in Figure 5.5, where our achievability scheme holds, the number of valid clustering partitions was upper bounded by $2^{\alpha M}$ for some $\alpha > 0$. Hence, with the aid of the codebook, the decoder is able to identify the correct clustering out of an exponentially large number of valid possibilities.

While this suggests that there may be scenarios where performing clustering and decoding separately is suboptimal, it does not provide any formal evidence. In particular, our upper bound on the number of ways to cluster is clearly loose (but sufficient to establish capacity).

Moreover, it may be possible to cluster a large fraction of the strings correctly in a code-oblivious way, and this may be sufficient for correct decoding of the message. In fact, as we discuss next, Lenz, Siegel, Wachter-Zeh, *et al.* [68] proved that the capacity of the BSC shuffling-sampling channel with multi-draws can be achieved in a regime with a scheme that performs clustering and decoding separately.

5.1.3 The BSC case

Notice that, using the definition of p_{eff} in (5.2), the capacity expression in Theorem 5.1 for the BEC multi-draw channel can be rewritten as

$$(1 - q_0) \left(E[1 - p^N | N \geq 1] - 1/\beta \right), \tag{5.11}$$

and the term $E[1 - p^N | N \geq 1]$ can be understood as the expected capacity of a "multi-draw binary erasure channel," where the decoder observes multiple passes of the input string through a BEC, and the number of passes is given by the conditional distribution of the random variable N given $N \geq 1$.

Given this, it is natural to conjecture that, for different choices of the noisy channel $p(y|x)$, the capacity of the multi-draw noisy shuffling-sampling channel is given by

$$(1 - q_0)\,(E[C_N|N \geq 1] - 1/\beta)\,, \tag{5.12}$$

or equivalently by

$$\sum_{n=1}^{\infty} q_n\, C_n - \frac{1 - q_0}{\beta}. \tag{5.13}$$

While both expressions are equivalent, (5.12) is a more natural generalization of the capacity expressions from Sections 3 and 4, since in the case of Bernoulli(q) sampling, $E[C_N|N \geq 1] = C_1$ is just the capacity of the noisy channel $p(y|x)$.

The capacity expression in (5.12) and (5.13) in fact holds, in addition to the BEC setting, for the BSC(p) case with Poisson sampling. More precisely, in Lenz, Siegel, Wachter-Zeh, *et al.* [65] and Lenz, Siegel, Wachter-Zeh, *et al.* [68], assuming $q_n = e^{-\lambda}\lambda^n/n!$ for $n \geq 0$, the following result is proven.

Theorem 5.6. Provided that $p < \frac{1}{8}$ and $1 - H(4p) - 2/\beta > 0$, the capacity of the BSC shuffling-sampling channel is

$$\begin{aligned} C_{\text{BSC, Poisson}(\lambda)} &= (1 - q_0)\,(E[C_{p,N}|N \geq 1] - 1/\beta) \\ &= \sum_{n=1}^{\infty} q_n\, C_{p,n} - \frac{1 - q_0}{\beta}, \end{aligned} \tag{5.14}$$

where $C_{p,n}$ is the capacity of a BSC with n draws, given by

$$C_n = 1 + \sum_{k=0}^{n} \binom{n}{k} p^k (1-p)^{n-k} \log \frac{1}{1 + p^{n-2k}(1-p)^{2k-n}}. \tag{5.15}$$

The formula (5.15) for the capacity of a BSC with n draws had been previously characterized by Mitzenmacher [76]. In addition to providing evidence that (5.12) and (5.13) may hold for more general channels, the result in Lenz, Siegel, Wachter-Zeh, *et al.* [68] requires the introduction of several new techniques, particularly for the achievability.

Notice that the achievability based on linear codes introduced in Section 5.1.1 does not generalize in a straightforward manner beyond

the erasure setting, since it requires the notion of consistency between output strings in order to build the consistency graph and identify valid clustering partitions. The approach in Lenz, Siegel, Wachter-Zeh, *et al.* [68] instead focuses on a (p, β) parameter regime where a greedy clustering algorithm can be shown to recover most of the clusters correctly with high probability. Notice that this is different from the achievability presented in Section 5.1.1 for the BEC case, which does not formally require a code-oblivious clustering algorithm to recover most of the clusters correctly.

In addition, rather than dealing with solving a linear system of equations as done in Section 5.1.1 for the BEC case, in order to achieve the capacity in Theorem 5.6, Lenz, Siegel, Wachter-Zeh, *et al.* [68] introduce a new notion of typicality between a set of n output strings and an input string.

Notice that, as in the BEC case discussed in Section 4.2.1, the achievability for the BSC case requires a more sophisticated scheme that goes beyond simple index-based coding. Intuitively, if one were to use an index-based scheme, each cluster would need to be decoded independently so that the indices corresponding to each cluster could be obtained. However, the number of sequences per cluster varies from cluster to cluster. Hence, it is unclear what level of error correction needs to be used for each string, since large clusters are easier to decode than small clusters. We derive the rate that is achievable by index-based coding in multi-draw settings in (7.4) in Section 7.1.

5.1.4 General Discrete Memoryless Channels

The capacity of multi-draw DNA storage channels where the noisy channel $p(y|x)$ is an arbitrary discrete memoryless channel recently received a careful treatment by Weinberger and Merhav [116]. The paper provides a new lower bound for general multi-draw DNA storage channels that is based on a random coding argument and holds for all values of $\beta > 1$, unlike the achievability arguments for Theorems 5.5 and 5.6, which only cover specific discrete memoryless channels. The paper also provides a new capacity upper bound in terms of a single-letter expression that holds for all values of β. The upper bound follows the

general approach outlined in Section 4.1.2 and elaborated by Lenz, Siegel, Wachter-Zeh, *et al.* [65], but is based on a more careful notion of distance between the sequences.

Weinberger and Merhav [116] also show that there exists a β_{cr} such that, for $\beta > \beta_{\text{cr}}$, the upper and lower bounds coincide, establishing the capacity of the multi-draw DNA storage channel for this regime. This is similar to the parameter regimes in Theorems 5.1 and 5.6 (which can be equivalently expressed as $\beta > \beta_{\text{cr}}(p)$). In particular, the paper considers the case of *modulo-additive* channels with input alphabet $\mathcal{X} = \{1, \ldots, |\mathcal{X}|\}$ and output alphabet $\mathcal{Y} = \mathcal{X}$, and input-output relationship given by

$$Y = X + Z \bmod |\mathcal{X}|,$$

where Z follows some distribution over $\{1, \ldots, |\mathcal{X}|\}$. The capacity of the modulo-additive multi-draw DNA storage channel is shown to be

$$C_{\text{mod},Q} = \sum_{n=1}^{\infty} q_n \, C_n - \frac{1 - q_0}{\beta}, \tag{5.16}$$

as long as $\beta > \beta_{\text{cr}}$, where β_{cr} can be explicitly computed [116], and C_n is the capacity of the modulo-additive channel with n draws. For the special case $\mathcal{X} = \{0, 1\}$, the modulo-additive channel reduces to the BSC, and the result in Theorem 5.6 is recovered, but with a weaker requirement on p and β. We point out that the capacity formula in (5.16) can be rewritten as

$$(1 - q_0) \left(E[C_N | N \geq 1] - 1/\beta \right),$$

which is the general formula for all capacity expressions discussed in this monograph, providing additional evidence for the conjecture in Section 7.1.

5.2 Connections with the trace reconstruction problem

If one considers the multi-draw channel studied in this section but with a single input sequence ($M = 1$), we obtain the problem of reconstructing an original "seed" string from several noisy versions of it. This problem was originally proposed by Levenshtein [69]. The special case where we

observe the output of passing the seed string multiple times through a deletion channel was studied in Batu, Kannan, Khanna, *et al.* [8] under the name of trace reconstruction problem. This problem received considerable attention in the last few years [1], [29], [44], [49], [72], [74], [78], [90], [106], [107], partially due to the fact that DNA sequencing technologies and especially nanopore sequencing technologies suffer from deletion errors [74]. In addition, the trace reconstruction problem has applications in immunogenomics, where one seeks to reconstruct the set of germline genes of an individual by analyzing antibody repertoires. In this setting, each antibody can be viewed as a trace generated by recombining elements from the germline genes [11].

Most of the work on the trace reconstruction problem has focused on characterizing the minimum number of traces (i.e., output sequences) needed for reconstructing the seed string correctly. For example, for a random binary seed string of length n, it is known that $\exp(O(\log^{1/3} n))$ traces are sufficient [44], while for a worst-case binary string, $\exp(\tilde{O}(n^{1/5}))$ traces are sufficient [18].

The trace reconstruction problem has a very direct connection with the problem of retrieving data stored on DNA. As discussed in this section, under most proposed DNA storage prototypes, we observe a random number of copies of each stored DNA molecule. Hence, provided that one can correctly cluster the observed strings based on string of origin, the problem of reconstructing the sequences can be seen as several instances of the trace reconstruction problem. Notice, however, that treating each of these instances as independent trace reconstruction problems may be suboptimal from an information-theoretic standpoint, as discussed in Section 5.1.2. Hence, the DNA storage setting can be more appropriately cast as a "trace reconstruction with multiple seeds," as proposed and studied in Bhardwaj, Pevzner, Rashtchian, *et al.* [11].

A second and arguably more important distinction between the general trace reconstruction problem (with multiple seeds) and the DNA storage setting is the fact that the trace reconstruction problem is typically concerned with the reconstruction of an arbitrary seed string (or one drawn according to a probability distribution). On the other hand, DNA molecules in a storage system are chosen as codewords from a codebook. As such, they can be designed to facilitate the solution

of the trace reconstruction problem that the decoder faces. This key observation led to the proposal of the *coded trace reconstruction* problem by Cheraghchi, Gabrys, Milenkovic, *et al.* [24]. For the coded trace reconstruction with a single seed of length n, the authors propose marker-based code constructions, which rely on the placing of long runs of zeros throughout the seed string. These markers can be identified by the decoder and effectively allow the problem to be converted into many small coded trace reconstruction problems. Based on this idea, it is shown that one can build codes with asymptotic rate 1 for which the number of traces needed for correct decoding is $\exp\left(O((\log\log n)^{2/3})\right)$. This is in stark contrast to the $\exp(\tilde{O}(n^{1/5}))$ traces needed in the case of an arbitrary seed string [18].

We point out that while establishing the exact scaling of the number of traces needed for perfect reconstruction as a function of the seed length n is an important problem, it is less relevant in the short-molecule setting that is the focus of this monograph. Moreover, from a capacity standpoint, the coded trace reconstruction problem can be seen as a generalization of the standard deletion channel (where one observes the output multiple independent times instead of one). See the paper by Haeupler and Mitzenmacher [41] for results on the capacity of a channel where multiple copies of a sequence are passed through a deletion channel. As the capacity of the deletion channel is a long-standing open problem [23], characterizing the capacity of coded trace reconstruction with a constant number of traces should be just as difficult. In Section 7.2 we discuss additional open problems in the context of the deletion channel and trace reconstruction that are motivated by DNA storage.

5.3 Code constructions for multi-draw channels

In addition to the work on the coded trace reconstruction problem discussed in Section 5.2, several works have proposed code constructions for the multi-draw settings that arise in the context of DNA storage systems. Most of these focus on coding-theoretic questions, including the derivation of bounds on the size of codes with specific distance properties, and the development of explicit codes with efficient encoding and decoding.

The notion of *clustering-correcting codes* was proposed by Shinkar, Yaakobi, Lenz, *et al.* [97]. These codes are designed so that each of the input sequences has a data field and an index field, designed to facilitate the clustering of the output sequences in the presence of noise. Clustering is initially performed based on the index fields, but it is shown that the data fields can also be used to improve the clustering quality (and reduce the amount of redundancy needed). Additionally, Lenz, Siegel, Wachter-Zeh, *et al.* [67] studied the multi-draw DNA storage channel from a coding-theoretic perspective. In particular, versions of the Gilbert-Varshamov bound and the sphere-packing bound were derived, providing insight into fundamental limits for explicit code constructions.

5.4 Proof of Lemmas

Lemma 5.2. *Let* $\mathbf{G}$ *be an* $ML \times B$ *matrix with i.i.d.* Bernoulli(1/2) *entries. Fix any* $\delta \in (0,1)$ *and a submatrix* $\mathbf{G}'$ *formed by an arbitrary set of* $(1-\delta)B$ *rows of* $\mathbf{G}$*. Then* $\mathbf{G}'$ *is full rank (over the finite field* $\mathbb{F}_2$*) with probability tending to* 1 *as* $B \to \infty$*.*

Proof. We follow the approach in the lecture notes by Sayir [93]. In order for $\mathbf{G}'$ to be full rank, the $(n+1)$th row must be chosen as a vector that is not in the span of rows $1, \ldots, n$. Note that the space spanned by n linearly independent vectors in $\mathbb{F}_2$ has exactly 2^n distinct elements. If we assume that the first n rows are linearly independent, then the probability that the $(n+1)$th row (which is a B-dimensional vector) is not in the span of the first n rows is $1 - 2^{n-B}$. By induction we see that the probability that all $(1-\delta)B$ rows are linearly independent is

$$\prod_{j=1}^{(1-\delta)B} \left(1 - 2^{-(B-j+1)}\right) = \prod_{i=\delta B+1}^{B} \left(1 - 2^{-i}\right) = \frac{\prod_{i=1}^{B} (1 - 2^{-i})}{\prod_{i=1}^{\delta B} (1 - 2^{-i})}. \tag{5.17}$$

As $B \to \infty$, both the product in the numerator and in the denominator can be verified (e.g., using numerical software) to converge to

$$\prod_{i=1}^{\infty} \left(1 - 2^{-i}\right) = 0.28879...$$

which implies that (5.17) tends to 1 as $B \to \infty$, proving the lemma. Notice that, if $\delta = 0$, the probability does not tend to 1 and instead tends to 0.28879, which is in contrast to the case of a real-valued matrix with random entries from a continuous distribution, where all square submatrices will be full-rank with high probability. □

Lemma 5.3. *The number of incorrect edges Z satisfies*

$$\Pr\left(Z > M^{2-\gamma+\epsilon}\right) \to 0 \tag{5.18}$$

as $M \to \infty$, for any $\epsilon > 0$.

Proof. Consider two output strings y_i^L and y_j^L that are generated from distinct input strings x_i^L and x_j^L. We show that y_i^L and y_j^L are consistent with probability $M^{-\gamma}$.

Let $x_i^L[\ell]$ and $x_j^L[\ell]$, for $\ell = 1, \ldots, L$ be the individual symbols in the sequences. Notice that x_i^L and x_j^L are generated by choosing $\mathbf{t}_i$ and $\mathbf{t}_j$ uniformly at random from $\{0,1\}^B$, and computing $\mathbf{G}'\mathbf{t}_i$ and $\mathbf{G}''\mathbf{t}_j$, where $\mathbf{G}'$ and $\mathbf{G}''$ are each obtained by taking the L rows from $\mathbf{G}$ corresponding to the ith and jth input sequences (note that $\mathbf{G}\mathbf{t}_i$ has length ML).

First we claim that the $2L$ random variables $x_i^L[\ell], x_j^L[\ell]$, $\ell = 1, \ldots, L$, are mutually independent Bernoulli(1/2). Treating all vectors as column vectors, we can write

$$\begin{bmatrix} x_i^L \\ x_j^L \end{bmatrix} = \underbrace{\begin{bmatrix} \mathbf{G}' & 0 \\ 0 & \mathbf{G}'' \end{bmatrix}}_{H} \underbrace{\begin{bmatrix} \mathbf{t}_i \\ \mathbf{t}_j \end{bmatrix}}_{\tilde{\mathbf{t}}}. \tag{5.19}$$

The block diagonal matrix H above has dimension $2L \times 2B$, where $B = ML(1 - q_0 - \epsilon)(1 - p_{\text{eff}} - \epsilon)(1 - \epsilon)$. For M large enough, we have $B > L$, and H is full row-rank, with a null space of dimension $2B - 2L$.

Hence, for any $\mathbf{c} \in \mathbb{F}_2^L$, the number of solutions $\tilde{\mathbf{t}}$ to $\mathbf{c} = H\tilde{\mathbf{t}}$ is 2^{2B-2L} and, if $\tilde{\mathbf{t}}$ is drawn uniformly at random from $\mathbb{F}_2^{2B}$,

$$\Pr(H\tilde{\mathbf{t}} = \mathbf{c}) = \frac{2^{2B-2L}}{2^{2B}} = 2^{-2L}.$$

This implies that the column vector $\begin{bmatrix} x_i^L \\ x_j^L \end{bmatrix}$ is chosen uniformly at random from $\mathbb{F}_2^{2L}$. This in turn implies that the entries of x_i^L and x_j^L are all mutually independent i.i.d. Bernoulli(1/2) random variables.

Given this fact, the event that y_i^L and y_j^L are consistent is the intersection of L independent events

$$\{x_i^L[\ell] = x_j^L[\ell] \text{ or } x_i^L[\ell] = \varepsilon \text{ or } x_j^L[\ell] = \varepsilon\},$$

for $\ell = 1, \ldots, L$. Each of these events happens with probability $1 - \frac{1}{2}(1-p)^2$, implying that x_i^L and x_j^L are consistent with probability

$$(1 - \tfrac{1}{2}(1-p)^2)^L = 2^{-\gamma \log M} = M^{-\gamma}.$$

Finally, we notice that the expected number of output sequences is $ME[N_1]$, and the expected number of pairs of output strings is at most $M^2E[N_1]^2$. Hence, the expected number of incorrect edges satisfies

$$E[Z] \leq M^2 E[N_1]^2 M^{-\gamma} = E[N_1]^2 M^{2-\gamma}.$$

Finally, using Markov's inequality, we have that

$$\Pr\left(Z > M^{2-\gamma+\epsilon}\right) \leq \frac{E[Z]}{M^{2-\gamma+\epsilon}} \leq \frac{E[N_1]^2 M^{2-\gamma}}{M^{2-\gamma+\epsilon}} = E[N_1]^2 M^{-\epsilon},$$

which tends to 0 as $M \to \infty$ for any $\epsilon > 0$. □

6

Coding and Computational Aspects

The results on the noisy shuffling-sampling channel in Sections 3, 4 and 5 provide insights into optimal ways to encode data for DNA based storage (for example, the fact that an index-based scheme is optimal in the single-draw setting). However, when designing coding schemes for real DNA storage systems, several additional factors need to be taken into consideration. These include the specific noise profiles of the technologies being used and the computational complexity of the algorithms employed in encoding and decoding. In this section, we discuss practical coding schemes for DNA data storage, as well as their limitations and computational aspects.

The first two demonstrations of DNA storage systems [27], [38] are based on indexing individual sequences. Both systems did not use modern error correcting codes to protect the information: the scheme by Church, Gao, and Kosuri [27] partitions the data to be stored into parts, adds an index to each part, and maps the parts to DNA sequences. The scheme by Goldman, Bertone, Chen, *et al.* [38] also first partitions the information into parts, but ensures that neighboring parts overlap, which introduces redundancy. Next, the scheme by Goldman, Bertone, Chen, *et al.* [38] adds an index to each part, and finally maps the parts

into DNA sequences. Since no principled error-correction mechanisms are used, both systems cannot efficiently deal with errors and thus the two demonstrations of DNA storage systems relied on accurate synthesis and on sequencing with large coverage. Both systems could recover the majority of the encoded information through an accurate experimental setup, but could not recover every single bit correctly.

Starting with Grass, Heckel, Puddu, *et al.* [39], all subsequent designs of DNA data storage systems used a combination of outer and inner error-correcting codes to protect the information against noise (for example, Blawat, Gaedke, Hütter, *et al.* [12], Bornholt, Lopez, Carmean, *et al.* [14], Chandak, Tatwawadi, Lau, *et al.* [17], Erlich and Zielinski [33], and Organick, Ang, Chen, *et al.* [80], and others). This approach, discussed in Section 6.1, works well if the errors within the sequences are relatively few and mostly substitution errors.

If the error probabilities within sequences are large and dominated by deletions and insertions, then it is necessary to combine the information of many noisy copies of each DNA sequence to recover the information. A natural approach to do this is to first cluster the sequences, second to reconstruct a sequence close to an original sequence from each cluster, and third to perform subsequent error correcting steps; we discuss this approach along with the associated computational costs in Section 6.2.

6.1 Inner-outer coding scheme

In Section 4.1 we presented an index-based coding scheme based on combining an inner code and an outer code (see Figure 4.1). While that scheme enabled us to establish an achievable rate (and the capacity) for the BSC shuffling-sampling channel, it relied on unspecified capacity-achieving codes for the BSC and capacity-achieving codes for an erasure channel over an alphabet of arbitrary size. Here, we describe an inner-outer coding scheme that uses existing practical codes, and that works well if individual sequences have few errors. The encoding and decoding steps, illustrated in Figure 6.1, are explained next.

Encoding: The data to be stored is given as a sequence of bits. We first map the data to B information blocks, denoted by $\mathbf{M}_b$, $b = 1, \ldots, B$,

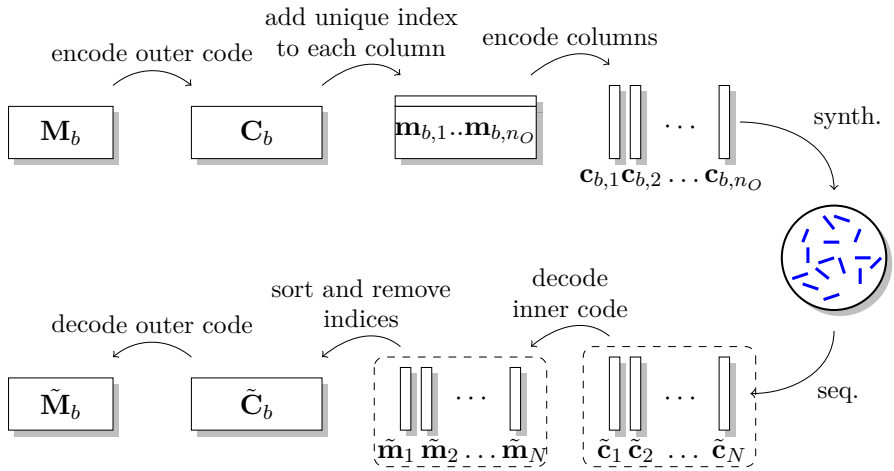

Figure 6.1: The inner-outer coding scheme breaks information into blocks $\mathbf{M}_b$, encodes each row of the blocks with an outer code that can correct erasures and substitution errors, adds a unique index to each column, and finally encodes each column with an inner code. The inner code enables correcting errors within the sequences, and the outer code recovers sequences that are not sequenced, and correct errors made by the inner decoding step.

each consisting of $m \times k_O$ symbols in a finite field. The size of the finite field depends on the code used as explained below, and the parameter m depends on the desired final length of the sequences. The original data is a file stored in bits, and mapping the data to information blocks with elements in a finite-field simply corresponds to converting a number from one base to another.

Each row of each information block is encoded with a code we call *outer code*. This yields the outer-encoded block of codewords $\mathbf{C}_b$, each consisting of $m \times n_O$ symbols, where n_O is the length of the outer code. The outer code needs to be able to correct substitution errors and erasures; an example of such a code is a Reed-Solomon code. We discuss choices for the outer code below.

Each of the n_O columns of the information block will correspond to a DNA sequence. Since the DNA sequences are unordered, we add a unique index to each of the columns of each of the information blocks. For example, the ith column in the bth information block receives the

index $n_O \cdot b + i$. Note that the outer code can correct damaged or lost DNA sequences, as explained below.

Next, each column of an information block along with the index, denoted by $\mathbf{m}_{b,1}, \ldots, \mathbf{m}_{b,n_O}$, is viewed as a vector of length k_I with symbols in an appropriate finite field, but not necessarily the same field as that used for the outer code. Each of the indexed vectors is encoded with an *inner code* yielding the codewords $\mathbf{c}_{b,1}, \ldots, \mathbf{c}_{b,n_O}$, each consisting of $n_I > k_I$ symbols.

Finally, each inner codeword $\mathbf{c}_{b,j}$ is mapped to a string with symbols in $\{\mathsf{A}, \mathsf{C}, \mathsf{G}, \mathsf{T}\}$, corresponding to the four nucleotides. Those strings are then synthesized in DNA.

Decoding: In the decoding stage, we start with N DNA sequences, obtained from sequencing. Those sequences are mapped to the received inner codewords $\tilde{\mathbf{c}}_1, \tilde{\mathbf{c}}_2, \ldots, \tilde{\mathbf{c}}_N$, where each codeword $\tilde{\mathbf{c}}_j$ consists of n_I symbols.

Those inner codewords are then decoded independently one by one which yields the information vectors $\tilde{\mathbf{m}}_1, \tilde{\mathbf{m}}_2, \ldots, \tilde{\mathbf{m}}_N$. This step corrects errors within each of the sequences. We then sort the vectors $\tilde{\mathbf{m}}_j$ by their index, and construct the received outer codewords $\tilde{\mathbf{C}}_b$, which consists of $m \times n_O$ symbols. Note that for each column of the outer codeword $\tilde{\mathbf{C}}_b$ there might be several candidates $\tilde{\mathbf{m}}_j$. Out of those candidates we select the one which has the smallest number of errors, if this information is available to us from the inner decoder. If there are several candidates with the same number of decoding errors, we choose the one which occurs most frequently. If there are several that occur equally frequently, we simply choose one of the candidates at random.

Columns of $\tilde{\mathbf{C}}_b$ for which we do not have a candidate $\tilde{\mathbf{m}}_i$ are marked as erasures. Columns of $\tilde{\mathbf{C}}_b$ can be erasures, if the respective sequence has not been sequenced, and substitutions, if the candidate obtained in the previous inner-decoding step contains errors. Finally, the rows of $\tilde{\mathbf{C}}_b$ are decoded individually with the outer decoder to obtain the received information blocks $\tilde{\mathbf{M}}_b$ consisting of $m \times n_O$ many symbols. The received information blocks are then mapped back to a sequence of bits, corresponding to the recovered information.

Choices for the outer code

The outer code needs to be able to correct substitution errors as well as erasures. Erasures arise from sequences that are lost, and substitution errors occur from errors within sequences that are not corrected by the inner code.

Grass, Heckel, Puddu, *et al.* [39] proposed to use Reed-Solomon codes as an outer code, and Organick, Ang, Chen, *et al.* [80] also used Reed-Solomon codes as an outer code. Reed-Solomon codes are algebraic codes that can correct e erasures and s substitutions provided that $e + 2s \leq n - k$, where n is the number of symbols of the codeword, and k is the number of information symbols of the codeword. Reed-Solomon codes are optimal in that they achieve the maximal obtainable minimum distance between two codewords and thus the maximal number of erasures and substitutions that can be corrected. Therefore, Reed-Solomon codes are a natural choice for the outer code in DNA storage. However, Reed-Solomon codes of large block length (i.e., large n) are computationally expensive to decode, and in DNA data storage large block length are desirable.

Chandak, Tatwawadi, Lau, *et al.* [17] proposed to use LDPC codes as an outer code to correct both erasures (lost sequences) as well as substitution errors in sequences. Contrary to Reed-Solomon codes, LDPC codes of long blocklength can be decoded very efficiently. Thus, designs based on LDPC can be an excellent choice for DNA storage system.

Erlich and Zielinski [33] proposed to use Fountain codes (see MacKay [71] for a brief introduction to Fountain codes) as an outer code. Fountain codes are codes for channels with erasures, designed for a setup where a file is transmitted in multiple small packets, and each of those packets is either received without error or is not received. In MacKay's words, Fountain codes "spray" packets at the decoder without requiring knowledge of whether the packets are received. Once sufficiently many packets are received (specifically, if slightly more than k_O packets are received where k_O is the number of original information packets), the data can be recovered.

The key feature of Fountain codes is its ratelessness: the transmitter does not require knowledge of how many packets have been received. DNA storage is not a natural setup for rateless codes, because we have to decide at storage time how much redundancy we need to add and generate the number of sequences to be stored accordingly. In addition, Fountain codes require $k_O + O(\sqrt{k_O}\log^2(k_O/\delta))$ symbols to recover a message with k_O symbols with probability $1-\delta$.

While Foutain codes use slightly more symbols than an optimal erasure code (such as a Reed-Solomon code which only requires k_O symbols for recovering a message of length k_O), the extra cost is negligible once the codeword becomes long (i.e., becomes negligible in k_O). However, properly designed Fountain codes are computationally efficient to decode. For example, Luby-transform codes are an efficient choice for Fountain codes, and are decodable at cost $O(k_O \log(k_O))$ with the Luby-Transform. The main shortcoming of Fountain codes for DNA storage is that they cannot correct substitution errors in an obvious way, which makes them a sub-optimal choice for a DNA storage setup, as in a DNA storage channel it is difficult to guarantee that all sequences passed to the outer decoder are error free.

Choices for the inner code

Ideally, the inner code would be able to correct substitution, deletion, and insertion errors, as all those errors occur during storage. However, currently there are no short codes (say, of length 50-200) that can correct all types of errors simultaneously with little redundancy.

Erlich and Zielinski [33], Grass, Heckel, Puddu, *et al.* [39], and Organick, Ang, Chen, *et al.* [80] used Reed-Solomon codes as an inner code. The sequences in those setups are are relatively short (60-200 base pairs long), and since short Reed-Solomon codes are computationally efficient for short codelengths they are a good choice for setups where substitution errors dominate.

Reed-Solomon codes and other standard codes (like Reed-Muller codes) are originally not designed to correct deletions or insertions, albeit linear codes (including Reed-Solomon codes) of large length can correct some deletion and insertion errors [28]. Being able to correct

deletions and insertions would be desirable as those errors can occur during synthesis, storage, and sequencing.

Channels that introduce deletions and insertions as well as corresponding coding schemes are much less understood than erasure and substitution channels and codes. The following is known about deletions and insertions: Mitzenmacher and Drinea [75] established that the capacity of a deletion channel, i.e., a channel that deletes each bit with probability p, is lower-bounded by $(1-p)/9$ for all p. For $p \to 0$, the capacity behaves like $1 - O(p\log(p))$ as established by Kanoria and Montanari [50]. This establishes that at least in the asymptotic regime, the capacity of a deletion channel is no more than nine times lower than that of an erasure channel which erases each bit with probability p, and for small p, the capacity is similar to that of a binary symmetric channel. The exact capacity of the deletion channel is still not known to date. For adversarial deletions and insertions, Haeupler and Shahrasbi [42] gave efficient insertion-deletion correcting codes for large alphabets, and Cheng, Guruswami, Haeupler, *et al.* [22] built on this work by building insertion-deletions correction codes that work with smaller alphabets. See Haeupler and Shahrasbi [43] for a recent overview on codes for deletions and insertions. To give a few examples, the papers by Brakensiek, Guruswami, and Zbarsky [15], Gabrys and Sala [36], Guruswami and Hastad [40], and Sima and Bruck [100] provide codes for correcting a constant number of deletions.

6.2 Coding for noisy setups

The inner-outer encoding-decoding scheme does not exploit the fact that we typically have multiple noisy copies of each sequence. If the sequences are relatively error-free or only contain few substitution errors, exploiting multiple copies is not necessary. However, if the sequences contain many substitution, insertion, and deletion errors, then the inner-outer encoding-decoding scheme does not work simply because no codes exist that enable correcting sufficiently many deletion, insertion, and substitution errors in short codewords for correctly decoding sufficiently many of the sequences.

In this section, we discuss a relatively straightforward decoding approach for very noisy sequences, which first clusters the sequences and second performs multiple alignment on the clusters in order to extract a single candidate sequence from each cluster. This candidate sequence has significantly fewer or no errors and can be passed through the previously discussed inner-outer encoding-decoding scheme to recover the information.

The idea of clustering reads before decoding and an efficient clustering scheme has been proposed by Rashtchian, Makarychev, Racz, *et al.* [85]. Antkowiak, Lietard, Darestani, *et al.* [7] built a DNA storage system with low-cost photolithographic synthesis which introduces large error rates in the sequences, with the clustering-multiple-alignment scheme described next.

Introducing randomization: Clustering millions of DNA sequences and recovering candidate sequences from the clusters is in general (i.e., for arbitrary sequences) intractable. The problem of recovering a sequence from multiple noisy copies of that sequences, each perturbed by random insertions and deletions, previously discussed in Section 5.2, is known as trace reconstruction.

Clustering and trace reconstruction become computationally and theoretically feasible if the fragments are random. Holden, Pemantle, and Peres [48] shows that for random sequences, the trace reconstruction problem can be solved based on only $O(\log^{1/3} L)$ traces/fragments, where L is the length of the sequences.

In DNA storage, it is possible to design the sequences so that they are pseudorandom for efficient computational clustering and trace-reconstruction. Specifically, we can simply multiply the information in the sequences with a pseudorandom sequence, so that all fragments appear random. That guarantees that distinct molecules are far from each other, and that individual sequences look like a pseudorandom sequence. This randomization step converts a difficult clustering problem instance to an "easy" instance, since now the true cluster centers are almost orthogonal to each other, and it also converts a potentially difficult trace reconstruction problem into an easy one.

Clustering reads: The goal of the clustering step is to efficiently cluster millions of short reads. Clustering algorithms are based on a notion of distance between DNA sequences or strings. The natural measure of distance between two sequences—given that the perturbations are deletions, insertions, and substitutions—is the edit distance. The edit distance between two sequences is the minimal number of deletions, insertions, and substitutions required to transform one sequence into the other. The edit distance between two sequences is expensive to compute.

We next describe an efficient method for clustering the noisy reads. The clustering method is based on locality sensitive hashing (LSH), and is inspired by an algorithm proposed for clustering web documents [45]. LSH relies on a cheaper-to-compute proxy for the edit distance, computed using the so-called Min-Hashing (MH) method.

To compute a proxy for the edit distance, we first split each sequence into overlapping subsequences of length k, called k-mers (also called shingles of length k or q-grams). For example, for $k = 2$, the sequence ACGT becomes the set $\{\mathsf{AC}, \mathsf{CG}, \mathsf{GT}\}$. Now, each sequence is represented by a set of k-mers and similarity between the two sets can be measured by the Jaccard similarity of two sets (the Jaccard similarity is defined as the size of the intersection divided by the size of the union of the sample sets). The clustering method we propose does not compute the Jaccard distance for all pairs of sequences, since this is computationally infeasible. Instead, we use locality sensitive hashing to find similar sequences. Locality sensitive hashing first generates a signature for each sequence, with the Min-Hashing method. Those sequences are likely to be equal if two sequences are similar or the same. Specifically, we perform the following steps:

1. Extract pairs of similar sequences: We first generate sets of pairs of sequences by finding all pairs of sequences that have the same signature.

2. Filter pairs: As a next step, we go through all the pairs and after performing a local pairwise alignment on each pair, we drop a pair from the set if the number of matched characters falls below a threshold.

3. Generate clusters from similar pairs: Finally, we generate clusters from the pairs based on sorting the pairs.

Organick, Ang, Chen, *et al.* [80] used a similar clustering-based approach as described above to recover data from nanopore reads, which have large insertion and deletion and error rates. Antkowiak, Lietard, Darestani, *et al.* [7] used the scheme described above to recover data in a system that uses noisy synthesis.

Reconstructing a single sequence from a set of noisy copies: After we produced clusters consisting of noisy copies of an original sequence, our goal is to reconstruct the original sequence. This is known as a trace reconstruction problem.

One approach to reconstruct a sequence is to first align the sequences within a cluster and second perform majority voting to extract a single sequence from the cluster. For aligning the clusters, a number of off-the-shelf algorithms such as the MUSCLE algorithm [31] are available.

Recent works have proposed efficient codes trace reconstruction, i.e., reconstruction of a sequence from multiple noisy copies. Specifically, Cheraghchi, Gabrys, Milenkovic, *et al.* [24] proposed marker-based code constructions with logarithmic complexity in the number of traces, and Srinivasavaradhan, Gopi, Pfister, *et al.* [108] proposed an efficient reconstruction algorithm with complexity that is linear in the number of traces. Lenz, Maarouf, Welter, *et al.* [64] proposed a concatenated coding scheme, and Chrisnata, Kiah, and Yaakobi [26] and Gabrys and Yaakobi [37] provide codes for correcting deletions from muliple noisy copies. Finally, Sabary, Yaakobi, and Yucovich [89] studies the correction of two insertions and deletions again from multiple reads.

Alternative approaches: Antkowiak, Lietard, Darestani, *et al.* [7] used the clustering algorithm described above and trace reconstruction by alignment followed by majority voting to store data reliably in a very noisy setup where the synthesis introduces large insertion and deletion error rates. While intuitive, it is unlikely that the clustering approach is information-theoretically close to optimal, as discussed in Section 5.1.2.

7

Extensions and Open Problems

In this monograph we took steps towards understanding the fundamental limits of DNA-based storage systems. Our main focus were noisy shuffling-sampling channels, which are a rich class of models for DNA storage systems that capture the fact that molecules are stored in an unordered fashion, are usually short, and are corrupted by individual base errors. By considering special cases of noisy shuffling-sampling channels, we built an increasingly more realistic sequence of models and characterized their capacity. We presented and discussed several recently proposed techniques, both in terms of achievability and converse, that enabled us to characterize the information-theoretic limits of noisy shuffling-sampling channels and gain insights for the design of practical systems that perform close to what is fundamentally possible.

7.1 Generalizing capacity results

Several generalizations of the capacity expressions in this monograph are possible. Note that the expression

$$(1 - q_0)\,(E[C_N|N \geq 1] - 1/\beta)^+ \tag{7.1}$$

discussed in Section 5.1.3, recovers the capacity expressions of all shuffling-sampling channels we studied as special cases. Here, C_N is the capacity of the respective noisy channel $p(y|x)$ with N draws. For the BSC shuffling-sampling channel studied in Section 4.1, for example, this follows from the fact that, for Bernoulli(q) sampling, $E[C_N|N \geq 1] = C_1$ is simply the capacity of the BSC that corrupts each individual sequence. Therefore, Theorems 3.2, 4.1, 4.3, 5.1, and 5.6 and the generalization by Weinberger and Merhav [116] discussed in Section 5.1.4 can all be stated using the capacity expression in (7.1).

Different alphabet sizes: While a quaternary alphabet $\Sigma = \{\mathsf{A}, \mathsf{C}, \mathsf{G}, \mathsf{T}\}$ is the relevant one for DNA-based storage systems, all results in this monograph were stated for binary alphabets for convenience. Here we discuss their generalization to an arbitrary alphabet Σ.

It is straightforward to see that the index-based scheme that achieves capacity in the error-free setting of Section 3 can be extended to a general alphabet Σ. In that case, $\log_{|\Sigma|} M$ symbols from each length-L sequence are needed for a unique index and we can store

$$\log|\Sigma| \left(L - \log_{|\Sigma|} M\right) = L(\log|\Sigma| - 1/\beta)$$

data bits per sequence. Using those bits for an outer code that handles the q_0 fraction of missing sequences (as done in Section 3.2), we achieve a rate of $(1 - q_0)(\log|\Sigma| - 1/\beta)$.

In the case of noisy shuffling-sampling channels, the index-based approach can be similarly generalized. Specifically, if C_{noisy} is the capacity of a channel $p(y|x)$ with Σ as input alphabet, then the rate

$$(1 - q_0)(C_{\text{noisy}} - 1/\beta)^+ \tag{7.2}$$

is achievable in the case of Bernoulli($1 - q_0$) sampling. Notice that $|\Sigma|$ does not appear in (7.2) as it is captured by C_{noisy}, which can now be larger than 1. Also notice that, in this case the capacity can be positive even for values of $\beta < 1$.

For natural extensions of the BEC and BSC to larger alphabets, the converse argument described in Section 4.1.1 can be extended in a natural way, establishing (7.2) as the capacity, at least for a certain regime of channel parameters.

A unified capacity expression: Since (7.1) also holds for general alphabet sizes, it is natural to state the following conjecture.

Conjecture 1. The capacity of a noisy shuffling-sampling channel (illustrated in Figure 2.2) with sampling distribution $N \sim (q_0, q_1, \dots)$ and discrete memoryless noisy channel $p(y|x)$ is given by

$$(1 - q_0)\left(E[C_N | N \geq 1] - 1/\beta\right)^+,$$

where C_n is the capacity of the noisy channel $p(y|x)$ with n draws.

We point out that, even in the special cases of the noisy shuffling-sampling channel where the capacity expression in Conjecture 1 holds, the result was only shown in specific parameter regimes. Hence, even in the case of the BEC and BSC, the conjecture has not been fully verified.

Optimality of index-based coding and independent decoding: As we discussed in Section 4.3, for single-draw channels where each sequence is either observed at the output once (with probability $1 - q$) or not observed at all (with probability q), an index-based coding scheme (that adds a unique index to each sequence) achieves rate

$$(1 - q_0)(C_{\text{noisy}} - 1/\beta). \tag{7.3}$$

The decoding is straightforward: each output sequence is independently decoded, and the indices are then used to order the sequences and recover the message. The rate above is then achieved by adding an outer code on top of the index-based coding to handle missing output sequences and the vanishingly small fraction of output sequences that are decoded in error, as discussed in Shomorony and Heckel [98].

In the multi-draw setting, however, a naive use of index-based coding is suboptimal. The reason for this is that, since the number of sequences in each cluster is random, the indices must be encoded with enough redundancy to be decodable even in clusters of size one. Therefore, only the data bits take advantage of the multi-draw setting. More concretely, in each length-L sequence, we need $(\log M)/C_1$ bits for the index, where C_1 is the capacity of the noisy channel (with one draw). We are left with

$M(L - (\log M)/C_1)$ bits for data. The data bits can take advantage of the multi-draw nature of the channel, and can be encoded with rate

$$E[C_N] = (1 - q_0)E[C_N|N \geq 1] + q_0 \cdot 0 = (1 - q_0)E[C_N|N \geq 1],$$

leading to an achievable rate of

$$\begin{aligned} &\frac{M(L - (\log M)/C_1)(1 - q_0)E[C_N|N \geq 1]}{ML} \\ &\quad = (1 - q_0)\left(E[C_N|N \geq 1] - \frac{\gamma}{\beta}\right), \end{aligned} \tag{7.4}$$

where $\gamma = E[C_N|N \geq 1]/C_1$. Since $C_i > C_1$ for $i > 1$ for any reasonable channel, $\gamma > 1$, showing that the index-based scheme is suboptimal. Whenever $\beta \gg \gamma$, simple index-based schemes perform close to optimal.

Despite its suboptimality in the multi-draw setting, index-based coding schemes are still interesting in practice for their simplicity. Moreover, based on the results of this monograph, we conjecture that they are still optimal in the single-draw setting.

Conjecture 2. For noisy shuffling-channels with Bernoulli$(1 - q)$ sampling, index-based coding is capacity-optimal.

Notice that, if Conjecture 1 holds, the fact that index-based schemes achieve the rate in (7.3) would imply that Conjecture 2 holds as well.

7.2 Insertions, deletions, and trace reconstruction

In all settings studied in this monograph, the noisy channel $p(y|x)$ is assumed to be a discrete memoryless channel. Single-base substitutions are the prevalent error source of most current DNA storage systems [47], which rely on *low-error* synthesis and sequencing technologies that are relatively expensive and limited in speed. A key idea towards developing the next-generation of DNA storage systems is to employ *high-error*, but cheaper and faster synthesis and sequencing technologies such as light-directed maskless synthesis and nanopore sequencing. Such systems induce a significant amount of insertion and deletion errors. Thus, an important area of further investigation is to understand the capacity of channels that introduce deletions and insertions as well.

From the point of view of characterizing the capacity, this poses significant challenges, since the capacity of (non-memoryless) channels with insertions and deletions is a long-standing open problem [25]. Nevertheless, it may be possible to establish the impact of adding shuffling and sampling on the capacity, without characterizing the capacity of the noisy channel. Given the results in Section 4, it is reasonable to conjecture the following.

Conjecture 3. Consider a noisy shuffling channel where the input sequences are shuffled and independently passed through a insertion-deletion channel. The capacity of this channel is

$$(C_{\text{indel}} - 1/\beta)^+,$$

where C_{indel} is the (unknown) capacity of the insertion-deletion channel.

Notice that, as discussed in Section 7.1, index-based schemes achieve the rate in Conjecture 3, and the challenge is in proving a matching outer bound. We point out that, in deriving the outer bounds in the BEC and BSC cases, it was important to know the capacity of C_{BEC} and C_{BSC} explicitly (see Section 4.2.2 for a discussion). Therefore, it is unlikely that the tools presented in this monograph can be used to establish Conjecture 3.

Another set of interesting open questions arises from studying the trace reconstruction problem from a capacity standpoint. Most of the work on trace reconstruction focuses on characterizing the number of traces needed for perfect reconstruction, as discussed in Section 5.2. The recent work on coded trace reconstruction [24] also studies the question of the amount of redundancy that needs to be added to the sequences in order to allow perfect reconstruction. However, the focus of this work is on the regime where the number of traces is growing, which makes the channel capacity be 1.

From a capacity standpoint, it would be interesting to study the regime where the number of traces T is constant. Since the case $T = 1$ reduces to a standard deletion channel, whose capacity is unknown, it is unlikely that the case $T > 1$ is any easier. However, it may still be possible to understand the impact of T on the capacity. For example, in the case of a BEC(p) with T "traces", it is straightforward to see

that the capacity is $1 - p^T$, as one can generate a consensus sequence with effective erasure rate p^T. In order to understand the impact of the number of traces in the capacity of the deletion channel, one could generalize existing bounds for the case of general T. Since several known bounds on the capacity of the deletion channel have the form $\alpha - \beta p$ for constants $\alpha, \beta > 0$, it is reasonable to conjecture that bounds of the form $\alpha - \beta p^T$ can be derived for the T-trace setting.

7.3 Storing data on very short molecules

One of the challenges in making DNA storage practically viable is that the cost of synthesizing relatively long DNA molecules with few errors is very expensive. For example, using the architecture proposed in Erlich and Zielinski [33], where the length of the synthesized molecules is around 150 nucleotides, the estimated cost of synthesizing 1GB of data was $3.27 million [33, Supplementary Material].

Synthesis costs are significantly lower if one focuses on short molecules and admits higher error rates [16], [55]. Hence, it is of interest to study the fundamental limits of DNA-based storage systems (and noisy shuffling-sampling channels) in the "very short" sequence regime. When the synthesized molecules are very short, placing a unique index may use up a significant fraction of the synthesized DNA. Moreover, in this regime it may make sense for the multi-set of stored DNA molecules to contain multiple copies of the same sequence and, in principle, one can utilize the frequencies of different molecules in the pool as a way to encode information.

Histograms as codewords and the Poisson channel: Consider once again the error-free sampling-shuffling channel discussed in Section 3.2. Due to the shuffling nature of the channel, the order of the sequences x_i^L in a given codeword $[x_1^L, \ldots, x_M^L]$ does not matter, and each codeword can be equivalently represented as a histogram of size 2^L, as illustrated in Figure 7.1. The effect of the sampling-shuffling channel can then be seen as simply changing the frequency of each of the molecules in the multiset; i.e., corrupting each entry in the histogram independently.

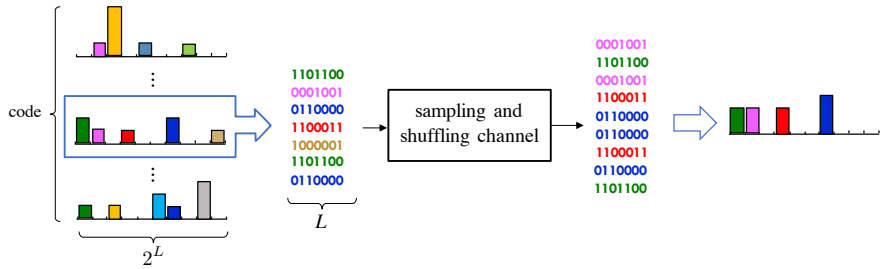

Figure 7.1: When several copies of each molecule are present in the multi-set of input sequences, it is reasonable to view the end-to-end channel as operating on histograms of size 2^L. The sampling-shuffling channel modifies each entry of the histogram according to the sampling distribution Q.

To formalize this representation, we can redefine the input to the channel to be a vector $[x[t] : t \in \{0,1\}^L]$ with entries in $\mathbb{Z}_+$ indexed by binary strings of length L (i.e., an input histogram, as illustrated in Figure 7.1). If we assume that the sampling distribution Q is Poisson(λ), then the channel can be described as

$$x[t] \in \mathbb{Z}_+ \quad \longrightarrow \quad y[t] \sim \text{Poisson}(\lambda x[t]), \tag{7.5}$$

for $t \in \{0,1\}^L$. Since we are only allowed to write M molecules, we have the constraint

$$\sum_{t\in\{0,1\}^L} x[t] \le M \;\Leftrightarrow\; \frac{1}{2^L} \sum_{t\in\{0,1\}^L} x[t] \le M2^{-L} = 2^{L(1/\beta-1)}. \tag{7.6}$$

This channel can be seen as a *discrete-time Poisson channel*, previously studied in the context of optical communications [60], [61], [95], [114]. However, unlike previously studied Poisson channels, the input is constrained to be integer-valued. Moreover, the *average power constraint* in (7.6) changes with the block length 2^L. This channel provides us with a good model to study the shuffling channel in the "very short" molecule regime, as we discuss next.

Encoding information in concentrations: Notice that, for all settings considered, when $\beta < 1$, the capacity is zero. Intuitively, this is because there are only $2^L = M^\beta < M$ distinct binary strings of length L. Hence, in this regime, one is forced to use the same string many times in

the input, and the capacity is zero. However, motivated by the fact that replicating DNA via PCR is a relatively common and inexpensive biotechnology technique [4], [82], [118], it is interesting to ask whether it is possible to encode information in the frequencies, or concentrations, of very short molecules.

To gain insight into how to optimally encode information in frequencies one can draw from the literature on the Poisson channel. Lapidoth and Moser [60] provide the following asymptotic characterization of the capacity of a Poisson channel.

Theorem 7.1. Consider the discrete-time Poisson channel

$$x[t] \geq 0 \quad \longrightarrow \quad y[t] \sim \text{Poisson}(x[t]),$$

with average power constraint $\mathbb{E}\,[x[t]] \leq P$. The capacity $C(P)$ satisfies

$$\lim_{P\to\infty} \left(C(P) - \frac{1}{2}\log P \right) = 0.$$

Notice that the average power power constraint in (7.6) is $P = 2^{L(1/\beta-1)}$. When $\beta < 1$, we have $P \to \infty$, and the approximation $C(P) \approx \frac{1}{2}\log(P)$ suggested by Theorem 7.1 should hold. Since for (7.5) we have 2^L channel uses, Theorem 7.1 suggests that the maximum achievable rate in bits per nucleotide should be roughly

$$\frac{2^L}{LM}\frac{1}{2}\log(P) = \frac{M^\beta L(1-\beta)}{2ML\beta} = \frac{1-\beta}{2\beta}M^{\beta-1}.$$

This approximation suggests that in the short-molecule regime $\beta < 1$, the number of bits per nucleotide that can be reliably stored scales as $M^{\beta-1}$. While this storage rate goes to zero, it suggests that the number of data bits stored can grow as $M^\beta L$, motivating the following conjecture.

Conjecture 4. Consider an error-free shuffling-sampling channel with $\beta \in (0,1)$. If $\{\mathcal{C}_M\}$ with $M \to \infty$ is a sequence of codes with vanishing error probabilities, then

$$\limsup_{M\to\infty} \frac{\log|\mathcal{C}_M|}{M^\beta L} \leq \frac{1-\beta}{2\beta}. \tag{7.7}$$

Moreover, there exist codes $\{\mathcal{C}_M\}$ that satisfy (7.7) with equality.

Formally establishing this result will require understanding the non-standard Poisson channel in (7.5), where the input is constrained to be integer-valued and the average power constraint changes with the block length.

Once this error-free setting is established, one can seek generalizations to noisy cases, similar to what was done in Section 4, but in the short-molecule regime. Notice that the noise, say of a BSC, causes sequences from one entry of the histogram to jump to a different entry of the histogram, and the channel is no longer memoryless, further complicating the setting.

7.4 New technologies and paradigms

DNA-based storage is an emerging idea and the system-level details depend on the current state of the art in terms of DNA synthesis, storage, and sequencing. As these technologies evolve, one may need to extend or adapt the theoretical framework proposed in this monograph to different settings.

Storing data on different macromolecules: In addition to DNA, synthetic polymers have been considered as a medium for data storage [35], [83]. In this setting, one can encode binary data using polyphosphodiesters in such a way that the two bits 0 and 1 are represented by molecules of different masses that are stitched together into a long string. The data retrieval is performed using tandem mass spectrometry. At a high level, many copies of the long polymer are broken down to pieces of random sizes, and the tandem mass spectrometer provides estimates of the *masses* of the fragments. Assuming that the masses of the 0 and 1 molecules are not multiples of each other, from the mass of each fragment one can obtain its *composition*; i.e., the number of zeros and ones in it.

Based on this setting, Acharya, Das, Milenkovic, *et al.* [2] introduced the problem of binary string reconstruction from its *substring composition multiset*. For example, the substring composition multiset of the string 1001 is $\{1, 0, 0, 1, 1^1 0^1, 0^2, 1^1 0^1, 1^1 0^2, 1^1 0^2\}$. Follow-up work considered the case where the composition multiset may be subject to

errors and the case where one observes only prefixes and suffixes of the string [35], [83].

From the shuffling-sampling channel standpoint taken in this monograph, an interesting research direction would be to consider a setting where the mass spectrometer randomly picks N substrings from the input string and reports their sequence composition. Unlike in the existing literature, the goal would be to study the capacity of the resulting channel (as a function of the sampling distribution and potential noisy channel).

New synthesis technique: New DNA synthesis and sequencing techniques also lead to new questions on the fundamental limits of the respective systems. A concrete example is DNA-based storage via combinatorial assembly commercialized by the startup CATALOG and described in Roquet, Bhatia, Flickinger, *et al.* [87].

Most existing storage systems rely on array-based synthesis where DNA sequences are synthesized in parallel nucleotide-by-nucleotide. Roquet, Bhatia, Flickinger, *et al.* [87] proposed a combinatorial assembly approach, which generates DNA sequences by assembling presynthesized shorter sequences called components. A sequence consists of assembling T components. Each component is a DNA sequence, consisting of a beginning, center, and end part. For each of the T positions of the sequence, Q candidate sequences exists. For example, say a sequence consists of $T = 3$ components, and for each position, there are $Q = 2$ candidate components, yielding the components $x_{1,a}, x_{1,b}, x_{2,a}, x_{2,b}, x_{3,a}$, and $x_{3,b}$. The beginning and ends of the components are chosen to be complementary such that when, for example, the three components $x_{1,a}, x_{2,b}, x_{3,a}$ are mixed together, they form the sequence $[x_{1,a}, x_{2,b}, x_{3,a}]$.

This approach has the advantage that many sequences can be generated relatively fast, but at the cost of generating a pool of long sequences where each sequence only contains little information. For example, Roquet, Bhatia, Flickinger, *et al.* [87] stored 26 kB with this technology on 97920 sequences, which means each sequence only contains about 2 bits. In contrast, the aforementioned systems based on array-based synthesis store more than 100 bits per sequence, which

means that, to read the same amount of information, we have to read 50 times less sequences with the traditional array-based storage systems. However, characterizing the capacity of combinatorial assembly and other new technologies is an interesting problem with the potential of improving DNA storage systems based on them.

Acknowledgements

The work of Ilan Shomorony was supported in part by the National Science Foundation (NSF) under grant CCF-2007597.

Reinhard Heckel would like to thank Robert Grass for countless helpful discussions on modeling aspects of DNA storage channels and DNA storage in general. Reinhard Heckel has received partial funding for this project from the Institute of Advanced Studies at the Technical University of Munich and from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme, Grant agreement No. 964995.

References

[1] M. Abroshan, R. Venkataramanan, L. Dolecek, and A. G. i Fàbregas, "Coding for deletion channels with multiple traces," in *2019 IEEE International Symposium on Information Theory (ISIT)*, IEEE, pp. 1372–1376, 2019.

[2] J. Acharya, H. Das, O. Milenkovic, A. Orlitsky, and S. Pan, "String reconstruction from substring compositions," *SIAM Journal on Discrete Mathematics*, vol. 29, no. 3, 2015, pp. 1340–1371.

[3] R. Ahlswede and A. Kaspi, "Optimal coding strategies for certain permuting channels," *IEEE transactions on information theory*, vol. 33, no. 3, 1987, pp. 310–314.

[4] D. Aird, M. G. Ross, W.-S. Chen, M. Danielsson, T. Fennell, C. Russ, D. B. Jaffe, C. Nusbaum, and A. Gnirke, "Analyzing and minimizing pcr amplification bias in illumina sequencing libraries," *Genome biology*, vol. 12, no. 2, 2011, R18.

[5] L. Anavy, I. Vaknin, O. Atar, R. Amit, and Z. Yakhini, "Data storage in dna with fewer synthesis cycles using composite dna letters," *Nature Biotechnology*, vol. 37, no. 10, Oct. 2019, pp. 1229–1236.

[6] P. Anderson, R. Black, A. Cerkauskaite, A. Chatzieleftheriou, J. Clegg, C. Dainty, R. Diaconu, R. Drevinskas, A. Donnelly, A. L. Gaunt, *et al.*, "Glass: A new media for a new era?" In *10th {USENIX} Workshop on Hot Topics in Storage and File Systems (HotStorage 18)*, 2018.

[7] P. L. Antkowiak, J. Lietard, M. Z. Darestani, 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 Communications*, 2020.

[8] T. Batu, S. Kannan, S. Khanna, and A. McGregor, "Reconstructing strings from random traces," *roceedings of the 15th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2004, pp. 910–918.

[9] E. B. Baum, "Building an associative memory vastly larger than the brain," *Science*, vol. 268, no. 5210, 1995, pp. 583–585.

[10] T. W. Benjamin, *Coding for a Noisy Channel with Permutation Errors*. Cornell University, 1975.

[11] V. Bhardwaj, P. A. Pevzner, C. Rashtchian, and Y. Safonova, "Trace reconstruction problems in computational biology," *IEEE Transactions on Information Theory*, 2020.

[12] 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," *Procedia Computer Science*, International Conference on Computational Science 2016, ICCS 2016, 6-8 June 2016, San Diego, California, USA, vol. 80, no. Supplement C, 2016, pp. 1011–1022.

[13] N. A. Bokulich, S. Subramanian, J. J. Faith, D. Gevers, J. I. Gordon, R. Knight, D. A. Mills, and J. G. Caporaso, "Quality-filtering vastly improves diversity estimates from illumina amplicon sequencing," *Nature methods*, vol. 10, no. 1, 2013, pp. 57–59.

[14] J. Bornholt, R. Lopez, D. M. Carmean, L. Ceze, G. Seelig, and K. Strauss, "A DNA-Based Archival Storage System," in *Proc. of ASPLOS*, pp. 637–649, New York, NY, USA: ACM, 2016. (accessed on 01/14/2017).

[15] J. Brakensiek, V. Guruswami, and S. Zbarsky, "Efficient low-redundancy codes for correcting multiple deletions," *IEEE Transactions on Information Theory*, vol. 64, no. 5, May 2018, pp. 3403–3410.

[16] M. H. Caruthers, *A brief review of DNA and rna chemical synthesis*, 2011.

[17] 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 *2019 57th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, IEEE, pp. 147–156, 2019.

[18] Z. Chase, "New lower bounds for trace reconstruction," in *Annales de l'Institut Henri Poincaré, Probabilités et Statistiques*, Institut Henri Poincaré, vol. 57, pp. 627–643, 2021.

[19] Y. M. Chee, H. M. Kiah, and T. T. Nguyen, "Linear-time encoders for codes correcting a single edit for DNA-based data storage," in *2019 IEEE International Symposium on Information Theory (ISIT)*, IEEE, pp. 772–776, 2019.

[20] Y. M. Chee, H. M. Kiah, and H. Wei, "Efficient and explicit balanced primer codes," in *2019 IEEE International Symposium on Information Theory (ISIT)*, 2019.

[21] Y.-J. Chen, C. N. Takahashi, L. Organick, C. Bee, S. D. Ang, P. Weiss, B. Peck, G. Seelig, L. Ceze, and K. Strauss, "Quantifying molecular bias in dna data storage," *Nature Communications*, vol. 11, no. 1, 2020, p. 3264.

[22] K. Cheng, V. Guruswami, B. Haeupler, and X. Li, "Efficient linear and affine codes for correcting insertions/deletions," in *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA)*, ser. Proceedings, pp. 1–20, 2021.

[23] M. Cheraghchi, "Capacity upper bounds for deletion-type channels," *Journal of the ACM (JACM)*, vol. 66, no. 2, 2019, pp. 1–79.

[24] M. Cheraghchi, R. Gabrys, O. Milenkovic, and J. Ribeiro, "Coded trace reconstruction," *IEEE Transactions on Information Theory*, 2020.

[25] M. Cheraghchi and J. Ribeiro, "Sharp analytical capacity upper bounds for sticky and related channels," *IEEE Transactions on Information Theory*, 2019.

[26] J. Chrisnata, H. M. Kiah, and E. Yaakobi, "Optimal reconstruction codes for deletion channels," in *2020 International Symposium on Information Theory and Its Applications (ISITA)*, IEEE, pp. 279–283, 2020.

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

[28] R. Con, A. Shpilka, and I. Tamo, "Linear and reed solomon codes against adversarial insertions and deletions," *arXiv:2107.05699 [cs, math]*, 2021.

[29] A. De, R. O'Donnell, and R. A. Servedio, "Optimal mean-based algorithms for trace reconstruction," in *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, pp. 1047–1056, 2017.

[30] E. L. van Dijk, Y. Jaszczyszyn, D. Naquin, and C. Thermes, "The third revolution in sequencing technology," *Trends in Genetics*, vol. 34, no. 9, 2018, pp. 666–681.

[31] R. C. Edgar, "Muscle: Multiple sequence alignment with high accuracy and high throughput," *Nucleic Acids Research*, vol. 32, no. 5, 2004, pp. 1792–1797.

[32] P. Elias, "Coding for two noisy channels," in *Information Theory, 3rd London Symposium, London, England, Sept. 1955*, 1955.

[33] Y. Erlich and D. Zielinski, "DNA fountain enables a robust and efficient storage architecture," *Science*, vol. 355, no. 6328, 2017, pp. 950–954.

[34] R. Gabrys, H. M. Kiah, and O. Milenkovic, "Asymmetric Lee distance codes: New bounds and constructions," in *2015 IEEE Information Theory Workshop (ITW)*, pp. 1–5, 2015.

[35] R. Gabrys, S. Pattabiraman, and O. Milenkovic, "Mass error-correction codes for polymer-based data storage," in *2020 IEEE International Symposium on Information Theory (ISIT)*, IEEE, pp. 25–30, 2020.

[36] R. Gabrys and F. Sala, "Codes correcting two deletions," *IEEE Transactions on Information Theory*, vol. 65, no. 2, 2019, pp. 965–974.

[37] R. Gabrys and E. Yaakobi, "Sequence reconstruction over the deletion channel," *IEEE Transactions on Information Theory*, vol. 64, no. 4, 2018, pp. 2924–2931.

[38] 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, 2013, pp. 77–80.

[39] R. 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," *Angewandte Chemie International Edition*, vol. 54, no. 8, 2015, pp. 2552–2555.

[40] V. Guruswami and J. Hastad, "Explicit two-deletion codes with redundancy matching the existential bound," *IEEE Transactions on Information Theory*, vol. 67, no. 10, 2021, pp. 6384–6394.

[41] B. Haeupler and M. Mitzenmacher, "Repeated deletion channels," in *2014 IEEE Information Theory Workshop*, pp. 152–156, Nov. 2014.

[42] B. Haeupler and A. Shahrasbi, "Synchronization strings: Codes for insertions and deletions approaching the singleton bound," in *ACM SIGACT Symposium on Theory of Computing*, pp. 33–46, 2017.

[43] B. Haeupler and A. Shahrasbi, "Synchronization strings and codes for insertions and deletions—a survey," *IEEE Transactions on Information Theory*, vol. 67, no. 6, 2021, pp. 3190–3206.

[44] L. Hartung, N. Holden, and Y. Peres, "Trace reconstruction with varying deletion probabilities," in *2018 Proceedings of the Fifteenth Workshop on Analytic Algorithmics and Combinatorics (ANALCO)*, SIAM, pp. 54–61, 2018.

[45] T. Haveliwala, A. Gionis, and P. Indyk, "Scalable techniques for clustering the web," 2000.

[46] R. Heckel and R. Grass, *Instructions to encode the first Biohackers episode from DNA*, Mar. 2021, URL: https://github.com/reinhardh/dna_rs_coding.

[47] R. Heckel, G. Mikutis, and R. N. Grass, "A characterization of the dna data storage channel," *Scientific Reports*, vol. 9, no. 1, 2019, pp. 1–12.

[48] N. Holden, R. Pemantle, and Y. Peres, "Subpolynomial trace reconstruction for random strings and arbitrary deletion probability," in *Conference On Learning Theory*, 2018.

[49] T. Holenstein, M. Mitzenmacher, R. Panigrahy, and U. Wieder, "Trace reconstruction with constant deletion probability and related results," in *Proceedings of the nineteenth annual ACM-SIAM symposium on Discrete algorithms*, Citeseer, pp. 389–398, 2008.

[50] Y. Kanoria and A. Montanari, "Optimal coding for the binary deletion channel with small deletion probability," *IEEE Transactions on Information Theory*, vol. 59, no. 10, 2013, pp. 6192–6219.

[51] H. M. Kiah, G. J. Puleo, and O. Milenkovic, "Codes for DNA sequence profiles," *IEEE Trans. on Information Theory*, vol. 62, no. 6, 2016, pp. 3125–3146.

[52] H. M. Kiah, T. T. Nguyen, and E. Yaakobi, "Coding for sequence reconstruction for single edits," in *2020 IEEE International Symposium on Information Theory (ISIT)*, IEEE, pp. 676–681, 2020.

[53] H. M. Kiah, A. Vardy, and H. Yao, "Efficient bee identification," in *IEEE International Symposium on Information Theory (ISIT)*, pp. 1943–1948, Jul. 2021.

[54] J. Koch, S. Gantenbein, K. Masania, W. J. Stark, Y. Erlich, and R. N. Grass, "A dna-of-things storage architecture to create materials with embedded memory," *Nature Biotechnology*, vol. 38, no. 11, Jan. 2020, pp. 39–43.

[55] S. Kosuri and G. M. Church, "Large-scale de novo DNA synthesis: Technologies and applications," *Nature methods*, vol. 11, no. 5, 2014, p. 499.

[56] M. Kovačević, "Runlength-limited sequences and shift-correcting codes: Asymptotic analysis," *IEEE Transactions on Information Theory*, 2019.

[57] M. Kovačević and V. Y. Tan, "Asymptotically optimal codes correcting fixed-length duplication errors in DNA storage systems," *IEEE Communications Letters*, vol. 22, no. 11, 2018, pp. 2194–2197.

[58] M. Kovačević and V. Y. Tan, "Codes in the space of multisets—coding for permutation channels with impairments," *IEEE Transactions on Information Theory*, vol. 64, no. 7, 2018, pp. 5156–5169.

[59] E. S. Lander and M. S. Waterman, "Genomic mapping by fingerprinting random clones: A mathematical analysis," *Genomics*, vol. 2, no. 3, 1988, pp. 231–239.

[60] A. Lapidoth and S. M. Moser, "On the capacity of the discrete-time poisson channel," *IEEE Transactions on Information Theory*, vol. 55, no. 1, 2008, pp. 303–322.

[61] A. Lapidoth, J. H. Shapiro, V. Venkatesan, and L. Wang, "The discrete-time poisson channel at low input powers," *IEEE Transactions on Information Theory*, vol. 57, no. 6, 2011, pp. 3260–3272.

[62] C. Laure, D. Karamessini, O. Milenkovic, L. Charles, and J.-F. Lutz, "Coding in 2d: Using intentional dispersity to enhance the information capacity of sequence-coded polymer barcodes," *Angewandte Chemie International Edition*, vol. 55, no. 36, 2016, pp. 10 722–10 725.

[63] H. H. Lee, R. Kalhor, N. Goela, J. Bolot, and G. M. Church, "Terminator-free template-independent enzymatic dna synthesis for digital information storage," *Nature Communications*, vol. 10, no. 1, Jun. 2019, p. 2383.

[64] 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 *IEEE Information Theory Workshop (ITW)*, pp. 1–5, 2021.

[65] A. Lenz, P. Siegel, A. Wachter-Zeh, and E. Yaakobi, "An upper bound on the capacity of the DNA storage channel," in *IEEE Information Theory Workshop*, 2019.

[66] A. Lenz, P. H. Siegel, A. Wachter-Zeh, and E. Yaakobi, "Anchor-based correction of substitutions in indexed sets," in *2019 IEEE International Symposium on Information Theory (ISIT)*, IEEE, 2019.

[67] A. Lenz, P. H. Siegel, A. Wachter-Zeh, and E. Yaakobi, "Coding over sets for dna storage," *IEEE Transactions on Information Theory*, vol. 66, no. 4, 2019, pp. 2331–2351.

[68] A. Lenz, P. H. Siegel, A. Wachter-Zeh, and E. Yaakohi, "Achieving the capacity of the dna storage channel," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, pp. 8846–8850, 2020.

[69] V. Levenshtein, "Reconstruction of objects from a minimum number of distorted patterns," *Doklady Mathematics*, vol. 55, no. 3, 1997, pp. 417–420.

[70] K. Levick, R. Heckel, and I. Shomorony, "Achieving the capacity of a dna storage channel with linear coding schemes," *56^{th} Annual Conference on Information Sciences and Systems (CISS), IEEE*, 2022.

[71] D. J. C. MacKay, "Fountain codes," *IEE Proceedings - Communications*, vol. 152, no. 6, 2005, pp. 1062–1068.

[72] A. Magner, J. Duda, W. Szpankowski, and A. Grama, "Fundamental bounds for sequence reconstruction from nanopore sequencers," *IEEE Transactions on Molecular, Biological and Multi-Scale Communications*, vol. 2, no. 1, 2016, pp. 92–106.

[73] A. Makur, "Information capacity of bsc and bec permutation channels," in *2018 56th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, IEEE, pp. 1112–1119, 2018.

[74] W. Mao, S. N. Diggavi, and S. Kannan, "Models and information-theoretic bounds for nanopore sequencing," *IEEE Transactions on Information Theory*, vol. 64, no. 4, 2018, pp. 3216–3236.

[75] M. Mitzenmacher and E. Drinea, "A simple lower bound for the capacity of the deletion channel," *IEEE Transactions on Information Theory*, vol. 52, no. 10, 2006, pp. 4657–4660.

[76] M. Mitzenmacher, "On the theory and practice of data recovery with multiple versions," in *2006 IEEE International Symposium on Information Theory*, IEEE, pp. 982–986, 2006.

[77] A. S. Motahari, G. Bresler, and N. David, "Information theory of DNA shotgun sequencing," *IEEE Transactions on Information Theory*, vol. 59, no. 10, 2013, pp. 6273–6289.

[78] F. Nazarov and Y. Peres, "Trace reconstruction with exp (o(n1/3)) samples," in *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, pp. 1042–1046, 2017.

[79] M. S. Neiman, "Some fundamental issues of microminiaturization," *Radiotekhnika*, vol. 1, no. 1, 1964, pp. 3–12.

[80] L. Organick, S. D. Ang, Y.-J. Chen, R. Lopez, S. Yekhanin, K. Makarychev, M. Z. Racz, G. Kamath, P. Gopalan, B. Nguyen, and et al., "Random access in large-scale dna data storage," *Nature Biotechnology*, 2018.

[81] L. Organick, B. H. Nguyen, R. McAmis, W. D. Chen, A. X. Kohll, S. D. Ang, R. N. Grass, L. Ceze, and K. Strauss, "An empirical comparison of preservation methods for synthetic dna data storage," *Small Methods*, vol. 5, no. 5, 2021, p. 2 001 094.

[82] S. Pabinger, S. Rödiger, A. Kriegner, K. Vierlinger, and A. Weinhäusel, "A survey of tools for the analysis of quantitative pcr (qpcr) data," *Biomolecular Detection and Quantification*, vol. 1, no. 1, 2014, pp. 23–33.

[83] S. Pattabiraman, R. Gabrys, and O. Milenkovic, "Reconstruction and error-correction codes for polymer-based data storage," in *2019 IEEE Information Theory Workshop (ITW)*, IEEE, pp. 1–5, 2019.

[84] K. R. Pomraning, K. M. Smith, E. L. Bredeweg, L. R. Connolly, P. A. Phatale, and M. Freitag, "Library preparation and data analysis packages for rapid genome sequencing," in *Fungal Secondary Metabolism*, Springer, 2012, pp. 1–22.

[85] C. Rashtchian, K. Makarychev, M. Racz, S. Ang, D. Jevdjic, S. Yekhanin, L. Ceze, and K. Strauss, "Clustering billions of reads for dna data storage," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[86] A. N. Ravi, A. Vahid, and I. Shomorony, "Capacity of the torn paper channel with lost pieces," in *IEEE International Symposium on Information Theory (ISIT)*, 2021.

[87] N. Roquet, S. P. Bhatia, S. A. Flickinger, S. Mihm, M. W. Norsworthy, D. Leake, and H. Park, "Dna-based data storage via combinatorial assembly," *bioRxiv*, 2021.

[88] O. Sabary, Y. Orlev, R. Shafir, L. Anavy, E. Yaakobi, and Z. Yakhini, "Solqc: Synthetic oligo library quality control tool," *Bioinformatics*, vol. 37, no. 5, 2021, pp. 720–722.

[89] O. Sabary, E. Yaakobi, and A. Yucovich, "The error probability of maximum-likelihood decoding over two deletion/insertion channels," in *2020 IEEE International Symposium on Information Theory (ISIT)*, pp. 763–768, 2020.

[90] O. Sabary, A. Yucovich, G. Shapira, and E. Yaakobi, "Reconstruction algorithms for dna-storage systems," *bioRxiv*, 2020.

[91] T. H. Saey, "Story one: Ancient horse's DNA fills in picture of equine evolution: A 700,000-year-old fossil proves astoundingly well preserved," *Science News*, vol. 184, no. 2, 2013, pp. 5–6.

[92] F. Sala, R. Gabrys, C. Schoeny, and L. Dolecek, "Exact reconstruction from insertions in synchronization codes," *IEEE Transactions on Information Theory*, 2017.

[93] J. Sayir, *Lecture notes for advanced communications and coding: Binary linear codes over the erasure channel*, 2014, URL: https://www-sigproc.eng.cam.ac.uk/foswiki/pub/Main/4F5/cod1.pdf.

[94] M. Schirmer, R. D'Amore, U. Z. Ijaz, N. Hall, and C. Quince, "Illumina error profiles: Resolving fine-scale variation in metagenomic sequencing data," *BMC Bioinformatics*, vol. 17, Mar. 2016, p. 125.

[95] S. Shamai and A. Lapidoth, "Bounds on the capacity of a spectrally constrained poisson channel," *IEEE Transactions on Information Theory*, vol. 39, no. 1, 1993, pp. 19–29.

[96] S. Shin, R. Heckel, and I. Shomorony, "Capacity of the erasure shuffling channel," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, pp. 8841–8845, 2020.

[97] T. Shinkar, E. Yaakobi, A. Lenz, and A. Wachter-Zeh, "Clustering-correcting codes," in *2019 IEEE International Symposium on Information Theory (ISIT)*, IEEE, 2019.

[98] I. Shomorony and R. Heckel, "Dna-based storage: Models and fundamental limits," *IEEE Transactions on Information Theory*, vol. 67, no. 6, 2021, pp. 3675–3689.

[99] I. Shomorony and A. Vahid, "Communicating over the torn-paper channel," in *IEEE Global Communications Conference (IEEE GLOBECOM)*, 2021.

[100] J. Sima and J. Bruck, "On optimal k-deletion correcting codes," *IEEE Transactions on Information Theory*, vol. 67, no. 6, Jun. 2021, pp. 3360–3375.

[101] J. Sima, N. Raviv, and J. Bruck, "On coding over sliced information," in *2019 IEEE International Symposium on Information Theory (ISIT)*, IEEE, pp. 767–771, 2019.

[102] J. Sima, N. Raviv, and J. Bruck, "Robust indexing - optimal codes for dna storage," in *2020 IEEE International Symposium on Information Theory (ISIT)*, pp. 717–722, Jun. 2020.

[103] S. Singh-Gasson, R. D. Green, Y. Yue, C. Nelson, F. Blattner, M. R. Sussman, and F. Cerrina, "Maskless fabrication of light-directed oligonucleotide microarrays using a digital micromirror array," *Nature Biotechnology*, vol. 17, no. 1010, Oct. 1999, pp. 974–978.

[104] W. Song, K. Cai, and K. A. S. Immink, "Sequence-subset distance and coding for error control in dna-based data storage," *IEEE Transactions on Information Theory*, vol. 66, no. 10, 2020, pp. 6048–6065.

[105] W. Song, K. Cai, and K. A. Schouhamer Immink, "Sequence-subset distance and coding for error control in dna-based data storage," *IEEE Transactions on Information Theory*, vol. 66, no. 10, Oct. 2020, pp. 6048–6065.

[106] S. R. Srinivasavaradhan, M. Du, S. Diggavi, and C. Fragouli, "Symbolwise map for multiple deletion channels," in *2019 IEEE International Symposium on Information Theory (ISIT)*, IEEE, pp. 181–185, 2019.

[107] S. R. Srinivasavaradhan, M. Du, S. Diggavi, and C. Fragouli, "On maximum likelihood reconstruction over multiple deletion channels," in *2018 IEEE International Symposium on Information Theory (ISIT)*, IEEE, pp. 436–440, 2018.

[108] S. R. Srinivasavaradhan, S. Gopi, H. D. Pfister, and S. Yekhanin, "Trellis bma: Coded trace reconstruction on ids channels for dna storage," 2021.

[109] S. K. Tabatabaei, B. Wang, N. B. M. Athreya, B. Enghiad, A. G. Hernandez, C. J. Fields, J.-P. Leburton, D. Soloveichik, H. Zhao, and O. Milenkovic, "Dna punch cards for storing data on native dna sequences via enzymatic nicking," *Nature Communications*, vol. 11, no. 1, Apr. 2020, p. 1742.

[110] R. Tamir and N. Merhav, "Error exponents in the bee identification problem," *arXiv preprint arXiv:2011.09799*, 2020.

[111] A. Tandon, V. Y. Tan, and L. R. Varshney, "The bee-identification problem: Bounds on the error exponent," *IEEE Transactions on Communications*, 2019.

[112] A. Tandon, V. Y. Tan, and L. R. Varshney, "The bee-identification error exponent with absentee bees," *IEEE Transactions on Information Theory*, vol. 66, no. 12, 2020, pp. 7602–7614.

[113] T. van der Valk, P. Pečnerová, D. Díez-del-Molino, A. Bergström, J. Oppenheimer, S. Hartmann, G. Xenikoudakis, J. A. Thomas, M. Dehasque, E. Sağlıcan, and et al., "Million-year-old dna sheds light on the genomic history of mammoths," *Nature*, vol. 591, no. 7849, Mar. 2021, pp. 265–269.

[114] S. Verdú, "Poisson communication theory," *International Technion Communication Day in Honor of Israel Bar-David*, vol. 66, 1999.

[115] H. Wei and M. Schwartz, "Improved coding over sets for dna-based data storage," 2021, pp. 1–1.

[116] N. Weinberger and N. Merhav, "The DNA storage channel: Capacity and error probability," *arXiv preprint arXiv:2109.12549*, 2021.

[117] J. L. Weirather, M. de Cesare, Y. Wang, P. Piazza, V. Sebastiano, X.-J. Wang, D. Buck, and K. F. Au, "Comprehensive comparison of pacific biosciences and oxford nanopore technologies and their applications to transcriptome analysis," *F1000Research*, vol. 6, 2017.

[118] G. Wong, I. Wong, K. Chan, Y. Hsieh, and S. Wong, "A rapid and low-cost pcr thermal cycler for low resource settings," *PLoS one*, vol. 10, no. 7, 2015, e0131701.

[119] H. T. Yazdi, Y. Yuan, J. Ma, H. Zhao, and O. Milenkovic, "A rewritable, random-access DNA-based storage system," *Scientific Reports*, vol. 5, 2015.

[120] H. T. Yazdi, R. Gabrys, and O. Milenkovic, "Portable and error-free DNA-based data storage," *Scientific Reports*, vol. 7, no. 1, 2017.

[121] S. S. Yim, R. M. McBee, A. M. Song, Y. Huang, R. U. Sheth, and H. H. Wang, "Robust direct digital-to-biological data storage in living cells," *Nature Chemical Biology*, vol. 17, no. 3, Mar. 2021, pp. 246–253.