New developments in coding against insertions and deletions

Ray Li

May 2017

Mellon College of Science
Carnegie Mellon University
Pittsburgh, PA 15213

Submitted in partial fulfillment of the requirements for the Honors Degree Program.

Keywords: Coding theory, Synchronization errors, Combinatorics, Pseudorandomness, Edit distance

Abstract

Unlike the traditional study of error correcting codes that analyzes substitution and erasure errors, error correcting codes for insertion and deletion errors are poorly understood. This thesis summarizes existing work on error correcting codes against insertions and/or deletions, and presents new work on such codes for different ways such insertions/deletions may be caused, such as adversarial (worst-case), random, oblivious, and online insertions/deletions. In doing so, we consider both algorithmic and existential results in these deletion models. With the exception of our first set of results, we focus on deletions only, which we believe captures much of the complexity of this problem.

We first consider the case of *adversarial (worst-case)* insertion and deletion errors. We present results for *efficiently* coding against *insertion and deletion* errors. We do this in three parameter settings, where efficient decoding against *deletions only* was already known. (i) Binary codes with rate approaching 1; (ii) Codes with constant rate for error fraction approaching 1 over fixed alphabet size; and (iii) Constant rate codes over an alphabet of size k for error fraction approaching $1 - 2/(k + \sqrt{k})$.

Continuing with our algorithmic results, we turn to *random deletions*. In the *binary deletion channel (BDC)*, bits are deleted independently with a fixed probability p. The existence of codes against the BDC of rate $(1 - p)/9$, and thus bounded away from 0 for any $p < 1$, was already known. We give an explicit construction with polynomial time encoding and deletion correction algorithms with rate $c_0(1 - p)$ for an absolute constant $c_0 > 0$.

In terms of existential results, one major question in coding against adversarial deletions is determining the *zero-rate threshold of adversarial deletions*, $p_0^{(adv)}$, which we define as the supremum of all p for which there exists a code family with rate bounded away from 0 capable of correcting pn adversarial deletions. Perhaps surprisingly, $p_0^{(adv)}$ is not known, and the best bounds are $\sqrt{2} - 1 \le p_0^{(adv)} \le \frac{1}{2}$.

To better understand adversarial deletions, we turn to models that are in between the adversarial and random models in power, namely oblivious deletions and online deletions. We can analogously define the zero-rate thresholds, $p_0^{(obliv)}$ and $p_0^{(on)}$, for oblivious and online deletions, respectively. In oblivious deletions, an adversary has access to the codebook and message, but chooses bit positions to delete obliviously of the codeword. We prove the existence of binary codes of positive rate that can correct an arbitrary pattern of p fraction of deletions, for any $p < 1$. In other words, $p_0^{(obliv)} = 1$.

For online deletions, where the channel decides whether to delete bit x_i based only on knowledge of bits $x_1 x_2 \ldots x_i$, define the corresponding zero-rate threshold $p_0^{(on)}$ to be the supremum of p for which there exist deterministic codes against an online channel causing pn deletions with low *average* probability of error. That is, the probability that a randomly chosen codeword is decoded incorrectly is small. We prove $p_0^{(adv)} = \frac{1}{2}$ if and only if $p_0^{(on)} = \frac{1}{2}$. In particular, to construct codes approaching the threshold of $1/2$ for adversarial deletions, it suffices to do so for online deletions.

Acknowledgments

I thank Venkat Guruswami for advising me on these projects and for his patience and constant support. I am grateful for Boris Bukh and James Cummings for serving on my committee. I thank the Carnegie Mellon Math Department for their support. I thank Joshua Brakensiek and Boris Bukh for helpful discussions regarding these projects.

Contents

List of Figures

Chapter 1

Introduction

Error correcting codes are objects designed for communication across unreliable or noisy channels. Error correcting codes have many applications including electronic communication, data storage, complexity theory. A fundamental question in the study of error correcting codes asks to determine the trade-off between the redundancy (formally, the rate) and error tolerance in various models of communication.

Traditionally, error correcting codes have been designed to tolerate substitution errors and symbol erasures (where the locations of the erased symbols is known) [22, 39]. However, recently much more work has been done to explore error correcting codes against synchronization errors, such as insertions and deletions. Insertions and deletions look similar to substitutions and erasures: deletions are erasures that corrupt synchronization information, and substitutions can be imitated by a deletion followed by an insertion. However, while our understanding of coding against substitution and erasures is near complete, there are large gaps in our understanding of coding against a combination of both insertions and deletions.

In this thesis, we present a detailed investigation of the deletion coding problem. In particular, we present constructive and impossibility results for coding against deletions in four models of errors: adversarial, random, oblivious, and online. In some cases, we include discussion on decoding against insertions and deletions.

1.1 Error correcting codes

In computing, one often needs to communicate a message (i.e. a string) across a *channel*. However, the channel may be *noisy*, so if we sent that message directly across the channel, it may be received incorrectly at the other end (see Figure 1.1). To deal with this, we can, using an error-correcting code, *encode* our message into a longer message, known as a *codeword*, that contains some redundancy, so that even in the presence of errors we can recover our original message. For example, for example, if our message is a binary string, we might encode as a longer binary string. Figure 1.2 illustrates such an error-correcting code. To encode our message, we duplicate our binary message three times. That way, even in the presence of a single bit-flip error, we can still recover or *decode* the original message by breaking the received message into three parts and choosing the majority.

While the practical motivations naturally lead us to think about error correcting codes in terms of their encoding and decoding algorithms, it is often convenient to think about our problem in terms of the encoded words themselves.

Definition 1.1.1. A *codebook* C of *block length* (or simply *length*) n over an alphabet Σ is a subset of Σ^n. The elements of C are called *codewords*.

In the literature, C is sometimes referred to simply as the *code*. However, the distinction between code and codebook becomes relevant when we reintroduce the algorithmic aspect, where codes are defined not only by their codebook but also their encoding and decoding functions. In such cases, we define a code by the codebook, C, a message set $\mathcal{M}$, an encoding function Enc, and a decoding function $\mathrm{Dec} : \Sigma^* \to \mathcal{M}$. Sometimes, the encoding function $\mathrm{Enc} : \mathcal{M} \times \{0,1\}^s \to C$ uses random bits that are private to the encoder (see, for example, Chapter 5). Typically, all codewords in a given code have the same length. In our example, our code is a binary (i.e. $\Sigma = \{0,1\}$) code $C = \{000000, 010101, 101010, 111111\}$. We could, using a *message set* $\mathcal{M} = \{00, 01, 10, 11\}$, define an encoder by $\mathrm{Enc}(m) = mmm$ and decoder $\mathrm{Dec}(x_1 \dots x_6) = \mathrm{Majority}(x_1, x_3, x_5)\,\mathrm{Majority}(x_2, x_4, x_6)$. In physical applications, codes can be *binary*, or over large alphabets, but we restrict our definitions and discussion in this introduction to binary codes for simplicity.

One can imagine in this problem setting there is a natural trade-off between the redundancy and robustness of our encoding schemes: if we introduce zero redundancy and send our messages as they are, then we cannot correct any errors, and as we introduce more redundancy we should be able to correct more errors. These notions of redundancy and robustness can be formalized as rate and error fraction, respectively.

Definition 1.1.2. The *rate* $\mathcal{R}$ of a code $C \subseteq \Sigma^n$ with message set $\mathcal{M}$ and encoding function $\mathrm{Enc} : \mathcal{M} \times \{0,1\}^r \to C$ is $\mathcal{R} = \frac{\log|\mathcal{M}|}{n \log|\Sigma|}$.

For deterministic codes, when there is no randomness in the encoder, we have $\mathcal{R} = \frac{\log|C|}{n\log|\Sigma|}$. For binary codes, we have $\mathcal{R} = \frac{\log|\mathcal{M}|}{n}$.

Definition 1.1.3. We say a code C with block length n corrects *error fraction* p if, for each codeword $c \in C$, when up to pn errors are applied to c, it is still possible to recover each original codeword.

Loosely speaking, the rate of a code is the fraction of the bits of a codeword that are "not redundant." In our example, the message is $\frac{1}{3}$ the length of the transmitted string, so the rate is $\frac{1}{3}$. This can also be verified with the formula $\log|C|/n = \frac{\log 4}{6} = \frac{1}{3}$. We also have $\frac{1}{6}$ of the transmitted bits are corrupted, so the error fraction tolerated is $\frac{1}{6}$.

We make several remarks about the informality of Definition 1.1.3. Indeed, our thesis seeks to understand these nuances.

1. In our definition of error fraction, we did not specify what an "error" is. In our example, the error is a *bit-flip*, also known as a *substitution*. As we hope to show, the nature of error drastically affects the way we construct and analyze error correcting codes. Throughout this thesis, the nature of error will be implied by the context. This thesis focuses on the setting where errors are insertions and/or deletions.

2. The definition of error fraction does not specify the manner in which the errors are applied. In this thesis, we address when errors are either *adversarial*, *online*, *oblivious*, or *random*. This affects the construction and analysis of codes. When errors are random, we may use the term *error rate*, instead of error fraction, to denote the probability that an error occurs at a given symbol in the code.

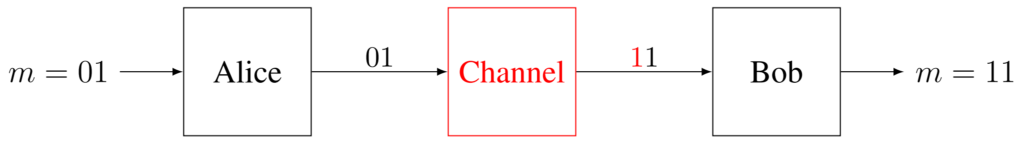


Figure 1.1: Noisy channel

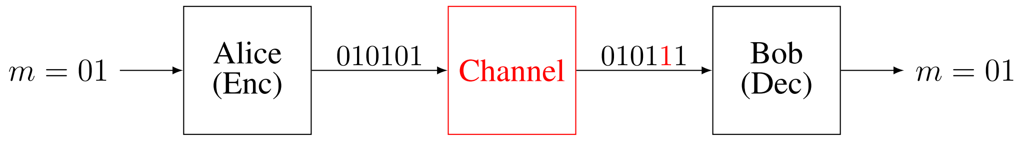


Figure 1.2: Error-correcting code

3. When we say "recover" we either mean recover with probability 1 (in the case of adversarial errors), or recover with probability that approaches 1 asymptotically as a function of n (in the case of oblivious, online, and random errors).

Ideally, our codes have rate bounded away from 0 and tolerate an *error fraction* that is bounded away from 0 as the length n tends to infinity. In this way, we often seek *families* of error correcting codes $C_1, C_2, \dots$ with block lengths $n_1 < n_2 < \cdots$ going to infinity, and $\lim_{k\to\infty} \mathcal{R}(C_k) > 0$. We may informally refer to such a family of codes as a *code with rate bounded away from 0* or a *constant rate code*.

For deterministic codes, rate is defined purely in terms of the codebook, and in this way, many error correcting code problems are purely mathematical or combinatorial (as opposed to algorithmic) problems. With this in mind, we can broadly separate the study of error-correcting codes into the following three types of questions.

1. (Existential) Given $\mathcal{R}, p \in (0, 1)$, do there exist codes with rate $\mathcal{R}$ and error fraction p?
2. (Constructive) Can we efficiently construct codes with rate $\mathcal{R}$ and error fraction p?
3. (Algorithmic) Can we efficiently encode and decode codes with rate $\mathcal{R}$ and error fraction p?

1.2 Types of errors

Shannon pioneered the study of error-correcting codes with his seminal 1948 paper [39]. Since then, tremendous progress has been made in designing error-correcting codes robust to substitution and erasure errors. On the other hand, the closely related insertions and deletion errors, first studied by Levenshtein in the 60s [32], are poorly understood. This thesis focuses on the setting of deletion

Error type	Before	After
Substitution	000	001
Erasure	000	00?
Deletion	000	00
Insertion	000	0010

Figure 1.3: Types of errors in error-correcting codes

and insertion errors.

The differences between these different types of errors is illustrated in Figure 1.3. In particular, erasures provide us the location of the erased bit, while deletion errors do not indicate the location of the deleted bit. Additionally, note that one substitution can be imitated by an insertion followed by a deletion. However, despite the apparent similarities between these types of errors, our understanding of substitutions and erasures is far beyond our understanding of insertions and deletions. Substitutions and erasures preserve the synchronization information (i.e. the position of each bit within the word) while insertions and deletions do not. Much of the major progress for substitutions and erasures decoding has been possible because the synchronization preservation has allowed us to leverage algebraic techniques like Reed-Solomon codes. By contrast, decoding against insertions and deletions is a fundamentally *combinatorial* problem, and without this valuable synchronization information, many of these classical techniques are either inapplicable or messy to apply. To quote from Mitzenmacher's survey [35]: "[C]hannels with synchronization errors, including both insertions and deletions as well as more general timing errors, are simply not adequately understood by current theory. Given the near-complete knowledge we have [for] channels with erasures and errors . . . our lack of understanding about channels with synchronization errors is truly remarkable."

From a practical standpoint, synchronization errors like deletions may become increasingly relevant as they become a greater bottleneck in engineering applications [35]. Understanding synchronization also may be relevant for biological applications such as analyzing DNA, where *tandem duplications* form about 3% of the human genome [1, 29]. They may also be relevant in queuing theory, in particular *timing channels*, where information is encoded in the transmission times of messages [27].

1.3 Noise models

We address the following noise models in this thesis.

1.3.1 Adversarial deletions

- A code C *decodes against* t *(p fraction of) adversarial deletions* (or *worst-case deletions*) if, for every two distinct codewords c and c', it is not possible to apply up to t (pn) deletions to each of c and c' to obtain the same string.
- We shall see in Lemma 2.3.1 that C decoding against t adversarial deletions is equivalent to $\mathrm{LCS}(c, c') \le n - t$ for all distinct $c, c' \in C$, where LCS denotes the length of the *longest common subsequence*.

1.3.2 Random deletions

- The *Binary Deletion Channel with deletion probability* p *(BDC$_p$)* takes as input a string $c = c_1 \dots c_n$, and deletes each bit c_i independently with probability p. The non-deleted bits form a string s of length at most n, which is the output of the channel.
- A codebook C with encoder $\mathrm{Enc} : \{0,1\}^{\mathcal{R}n} \to C$ and decoder $\mathrm{Dec} : \{0,1\}^* \to \{0,1\}^{\mathcal{R}n} \cup$

$\{\perp\}$ *decodes against BDC$_p$* if, for all messages $m \in \{0,1\}^{\mathcal{R}n}$,

$$\mathbf{Pr}[\mathrm{Dec}(\mathbf{BDC}_p(\mathrm{Enc}(m))) \neq m] = o(1) \tag{1.1}$$

where the probability is over the randomness of BDC$_p$, and the $o(1)$ is a term that goes to 0 with n.

- The decoder that makes the most sense against BDC would be the *maximum likelihood decoder*. This is the decoder Dec such that $\mathrm{Dec}(s)$ is the message m that maximizes the probability that BDC$_p(\mathrm{Enc}(m)) = s$. However, as the maximum likelihood decoder is difficult to analyze, the current best existential and algorithmic results [12, 17, 35] use slightly less optimal decoding methods in exchange for ease of analysis.

1.3.3 Oblivious deletions

- In the oblivious model of deletions, the channel knows the codebook and the message, but not the codeword, and chooses a pattern of up to pn locations within the codeword to delete.
- A code $(C, \mathrm{Enc}, \mathrm{Dec})$ is correctable against p fraction of *oblivious deletions under a worst-case error criterion* if for every deletion pattern τ and message m, we have

$$\mathop{\mathbf{Pr}}_{r}[\mathrm{Dec}(\tau(\mathrm{Enc}(m,r))) \neq m] = o(1), \tag{1.2}$$

where the probability is over the random bits r that are private to the encoder.

- A deterministic code $(C, \mathrm{Enc}, \mathrm{Dec})$ is correctable against p fraction of *oblivious deletions under an average-case error criterion* if the decoding failure probability of a uniformly random codeword is small. That is, for every deletion pattern τ, we have

$$\mathop{\mathbf{Pr}}_{m\sim U(\mathcal{M})}[\mathrm{Dec}(\tau(\mathrm{Enc}(m))) \neq m] = o(1), \tag{1.3}$$

where the probability is over the uniformly random message.

1.3.4 Online deletions

- For online deletions, the channel decides whether to delete bit x_i based only on knowledge of bits $x_1 x_2 \dots x_i$,
- Formally, an online deletion channel OnAdv consists of n functions $\{\mathrm{OnAdv}_i : i \in [n]\}$ such that $\mathrm{OnAdv}_i : \mathcal{X}^i \to \mathcal{Y}$, where $\mathcal{X} = \{0,1\}$ and $\mathcal{Y} = \{\langle 0\rangle, \langle 1\rangle, \langle\rangle\}$ is a set of strings and $\langle\rangle$ denotes the empty string, satisfies $\mathrm{OnAdv}(x_1, \dots, x_i) \in \{\langle x_i\rangle, \langle\rangle\}$. The resulting string received by the output is the concatenation of the outputs of $\mathrm{OnAdv}_1, \dots, \mathrm{OnAdv}_n$.
- A code $(C, \mathrm{Enc}, \mathrm{Dec})$ is correctable against p fraction of *online deletions under a worst-case error criterion* if, for every online adversary OnAdv and every message m, we have

$$\mathop{\mathbf{Pr}}_{r}[\mathrm{Dec}(\mathrm{OnAdv}(\mathrm{Enc}(m,r))) \neq m] = o(1), \tag{1.4}$$

where the probability is over the random bits r that are private to the encoder.

- A deterministic code $(C, \mathrm{Enc}, \mathrm{Dec})$ is correctable against p fraction of *online deletions under an average-error criterion* if the decoding failure probability of a uniformly at random codeword is small. That is, for every online adversary OnAdv, we have

$$\mathop{\mathbf{Pr}}_{m\sim U(\mathcal{M})}[\mathrm{Dec}(\mathrm{OnAdv}(\mathrm{Enc}(m))) \neq m] = o(1), \tag{1.5}$$

where the probability is over the uniformly random message.

1.3.5 Comments

For oblivious and online deletions, we assume the adversary is deterministic: if a randomized adversary could cause decoding failure with (worst-case or average) probability ϵ for some fixed $\epsilon > 0$, then sampling over the random bits of the adversary implies a *deterministic* adversary that causes decoding failure with probability ϵ. Thus, to code against randomized online adversaries, it suffices to code against deterministic adversaries.

Note that the oblivious and online models are in between the adversarial and random models in terms of the adversary's knowledge. In the adversarial model, the adversary has full knowledge of the codeword. In the random model, the channel only knows the current bit in deciding whether to delete it. In the oblivious model, the channel knows the codebook and the message, but not the codeword. In the online model, at bit i, the channel knows the first i bits of the codeword. For concreteness, we restrict the majority of this thesis to discussing deletions, which captures most (and sometimes all) of the complexity associated with coding against synchronization errors. In this thesis, we survey existing results on the adversarial, random, oblivious, and online models of deletions, addressing the existential, constructive, and algorithmic questions related to each error model.

1.4 Outline of thesis

Chapter 2 establishes some preliminary definitions and notation to be used throughout the thesis. In the following chapters, we in turn address four different error models of insertions and deletions while presenting new results for each. Each chapters begins with an introduction, followed by new results and a survey of relevant literature. The remainder of each chapter states and proves the new result(s). In Chapter 3 and Chapter 4, we establish algorithmic results for adversarial and random deletions, respectively. In Chapter 5, we establish a positive existential result for coding against oblivious deletions, and in Chapter 6, we establish a negative result for coding against online deletions.

A common question in coding theory is determining the *capacity* of a channel, i.e. the maximum rate of a family of constant rate codes for a given error fraction and model of errors. For random *erasures* or random *substitutions*, we have complete characterizations of the capacities [14, 39], and for adversarial erasures and substitutions we have good characterizations. However, for deletions, our knowledge is far less complete. In particular, for some deletion models, such as adversarial deletions, we cannot even determine when the capacity becomes 0, let alone find reasonable bounds on the capacity.

To address this question formally, we introduce the *zero-rate threshold*. Let $p_0^{(adv)}$, the *zero-rate threshold of adversarial deletions*, be the supremum of all p for which there exists a code family with rate bounded away from 0 capable of correcting pn adversarial deletions. No nontrivial code can correct $n/2$ adversarial deletions, where n is the block length of the code, as the adversary can ensure either $0^{n/2}$ or $1^{n/2}$ is received, regardless of the codeword. This shows that $p_0^{(adv)} \le \frac{1}{2}$. A recent construction of deletion-correcting codes [4] shows that $p_0^{(adv)} \ge \sqrt{2}-1$. These are the best known bounds for $p_0^{(adv)}$.

While we do not to improve these bounds for $p_0^{(adv)}$, we do improve on the best known algorithmic results. In Chapter 3, we present efficiently decodable codes against adversarial *insertions and deletions*, where efficient coding against *deletions only* was previously known. We find both families of efficient codes with rate approaching 1 tolerating constant fraction of insertions and deletions, and families of efficient constant rate codes tolerating a fraction of insertions and deletions approaching 1.

We can analogously define a zero-rate threshold for other models of deletions. For random deletions, there are positive rate codes against the binary deletion channel with deletion probability p for any $p < 1$, so the zero-rate threshold $p_0^{(rand)}$ is 1. For p approaching 1, the capacity of the BDC$_p$ is known to decay as $\alpha(1-p)$ for some $\alpha \in [1/9, 0.4143]$ [12, 36]. However, the capacity lower bound of $(1-p)/9$ is only an existential result, and in Chapter 4, we construct *efficient* codes against the BDC$_p$ with rate $\alpha(1-p)$ for a constant α.

In order to better understand the zero-rate threshold for adversarial deletions, $p_0^{(adv)}$, we turn to oblivious and online deletions. Let the *zero-rate threshold for oblivious deletions*, $p_0^{(obliv)}$, be the supremum of all p such that there exist families of codes with rate bounded away from 0 that can correct against p fraction of oblivious deletions. Define the corresponding zero-rate threshold $p_0^{(on)}$ to be the supremum of p for which there exist deterministic codes against pn online deletions under an average error criterion.

In Chapter 5, we prove, perhaps surprisingly, that $p_0^{(obliv)} = 1$. In Chapter 6, we prove $p_0^{(adv)} = \frac{1}{2}$ if and only if $p_0^{(on)} = \frac{1}{2}$. In particular, to construct codes approaching the threshold of $1/2$ for adversarial deletions, it suffices to construct deterministic codes for online deletions. We were not able to prove a similar result relating $p_0^{(adv)}$ and $p_0^{(on,s)}$, the zero-rate threshold for online deletions that allows both deterministic and stochastic codes.

Chapter 2

Preliminaries

2.1 Definitions

General Notation. For a boolean statement P, let $\mathbf{1}[P]$ be 1 if P is true and 0 otherwise.

Throughout this thesis, $\log x$ refers to the base-2 logarithm.

We use interval notation $[a,b] = \{a, a+1, \dots, b\}$ to denote intervals of integers, and we use $[a] = [1,a] = \{1,2,\dots,a\}$.

For a set S and an integer a, let $\binom{S}{a}$ denote the family of subsets of S of size a.

Let $U(S)$ denote the uniform distribution on S. Let $\mathrm{Binomial}(n,p)$ denote the Binomial distribution.

Let $\mathbb{F}_q$ denote the finite field of size q.

Words (Strings). A *word* (or *string*) is a sequence of symbols from some *alphabet* Σ. Let Σ^n denote words of length n and let $\Sigma^* = \cup_{n=0}^{\infty}\Sigma_n$. For clarity, particularly in Chapter 4, we may denote explicit words using angle brackets, like $\langle 01011\rangle$. We denote string concatenation of two words w and w' with ww'. We denote $w^k = ww\cdots w$ where there are k concatenated copies of w. We also denote a concatenation of a sequence of words as $w_1w_2\cdots w_k = \prod_{i=1}^k w_i$. We denote words from binary alphabets with lowercase letters c, s, w and words from non-binary alphabets with capital letters X, Y, Z.

A *subsequence* of a word w is a word obtained by removing some (possibly none) of the symbols in w.

A *subword* or *interval* of a word w is a contiguous subsequence of characters from w. We identify intervals of words with intervals of integers corresponding to the indices of the subsequence. For example, the interval $\{1,2,\dots,|w|\} = [1,|w|]$ is identified with the entire word w.

Let $w' \sqsubseteq w$ denote "w' is a subsequence of w".

Define a *run* of a word w to be a maximal single-symbol subword. That is, a subword w' in w consisting of a single symbol such that any longer subword containing w' has at least two different symbols. Note the runs of a word partition the word. For example, $\langle 110001\rangle$ has 3 runs: one run of 0s and two runs of 1s.

For a string w, let $|w|$ denote the length of the string. Let $\Delta_{i/d}(w_1,w_2)$ denote the *insertion/deletion distance* between w_1 and w_2, i.e. the minimum number of insertions and deletions needed to transform w_1 into w_2. For two words $w_1, w_2 \in C$, let $\mathrm{LCS}(w_1,w_2)$ be the length of the longest common subsequence of w_1 and w_2. Define $\mathrm{LCS}(C) = \max_{w_1,w_2\in C, w_1\neq w_2}\mathrm{LCS}(w_1,w_2)$.

A useful fact is that $\Delta(w_1, w_2) = |w_1| + |w_2| - 2\,\mathrm{LCS}(w_1, w_2)$. For intuition as to why, note that we can apply $|w_1| - \mathrm{LCS}(w_1, w_2)$ deletions to w_1 and $|w_2| - \mathrm{LCS}(w_1, w_2)$ deletions to w_2 to obtain the same string. This fact is closely related to Lemma 2.3.1 below.

Deletion Patterns. A *deletion pattern* is a function τ that removes a fixed subset of symbols from words of a fixed length. Let $\mathcal{D}(n, m)$ denote the set of deletion patterns τ that operate on length n words and apply exactly m deletions. For example $\tau : x_1x_2x_3 \mapsto x_1x_3$ is a member of $\mathcal{D}(3, 1)$. Let $\mathcal{D}(n) = \cup_{m=0}^{n}\mathcal{D}(n, m)$.

We identify each deletion pattern $\tau \in \mathcal{D}(n, m)$ with a size m subset of $[n]$ corresponding to the deleted bits. We often use sets to describe deletion patterns when the length of the codeword is understood. For example $[n]$ refers to the single element of $\mathcal{D}(n, n)$. Accordingly, let $\subseteq$ be a partial order on deletion patterns corresponding to set inclusion, and let $|\tau|$ denote the number of bits deleted by τ. As such, we have $\tau \subseteq \tau'$ implies $|\tau| \le |\tau'|$.

For a word w and $\tau \in \mathcal{D}(|w|)$, let $\tau(w)$ and $w \setminus \tau$ both denote the result of applying τ to w. We use the second notation when we identify sets with deletion patterns, as in the above paragraph where the set elements correspond to the deleted positions.

Graphs. In a (directed or undirected) graph G, let $V(G)$ and $E(G)$ denote the vertex set and edge set of G respectively. For a subset $W \subseteq V(G)$ of the vertices, let $G|_W$ denote the subgraph induced by W. For a vertex $v \in V(G)$, let $\deg_G(v)$ denote the degree of v in G when G is undirected, and let $\mathrm{indeg}_G(v)$, $\mathrm{outdeg}_G(v)$ denote the indegree and outdegree of v, respectively, in G when G is directed. We drop the subscript of G in $\deg$, indeg and outdeg notations when the graph G is understood.

Notation Conventions. Random variables are denoted by A, B, X, Y, Z. Codes are always denoted by the letter C, with possible decorations like C_{in}. Graphs are denoted by G. Deletion patterns are denoted by Greek letters σ, τ. In Chapter 4, σ also denotes outer codeword symbols. We denote sets by $S, \mathcal{J}, \mathcal{L}$ and intervals by I. We denote strings by c, g, r, s, w. We use $a, b, i, j, k, \ell, q, r, s, t, x, y$ as variables or constants. We use $\alpha, \beta, \gamma, \delta, \epsilon, \eta$ as constants. We typically denote finite field sizes by q.

2.2 Concentration bounds.

We use the following forms of Chernoff bound.

Lemma 2.2.1 (Chernoff)**.** *Let $A_1, \dots, A_n$ be independent random variables taking values in $[0, 1]$. Let $A = \sum_{i=1}^{n} A_i$ and $\delta \in [0, 1]$. Then*

$$\mathbf{Pr}[A \le (1-\delta)\,\mathbf{E}[A]] \;\le\; \exp\left(-\delta^2\,\mathbf{E}[A]/2\right). \tag{2.1}$$

Furthermore,

$$\mathbf{Pr}[A \ge (1+\delta)\,\mathbf{E}[A]] \;\le\; \left(\frac{e^{\delta}}{(1+\delta)^{(1+\delta)}}\right)^{\mathbf{E}[A]} \tag{2.2}$$

We also have the following corollary, whose proof is in Appendix A.

Lemma 2.2.2. *Let $0 < \alpha < \beta$. Let $A_1, \dots, A_n$ be independent random variables taking values in $[0, \beta]$ such that, for all i, $\mathbf{E}[A_i] \le \alpha$. For $\gamma \in [\alpha, 2\alpha]$, we have*

$$\mathbf{Pr}\left[\sum_{i=1}^{n} A_i \ge n\gamma\right] \le \exp\left(-\frac{(\gamma-\alpha)^2 n}{3\alpha\beta}\right). \tag{2.3}$$

We also use the submartingale form of Azuma's Inequality.

Lemma 2.2.3 (Azuma)**.** *Let M be a constant. Let $X_1, X_2, \dots$ be a* submartingale. *That is, they satisfy*

$$\mathbf{E}[X_{n+1}|X_1, \dots, X_n] \geq X_n. \tag{2.4}$$

Suppose that $|X_i - X_{i-1}| \leq M$ for all i. Then for all positive reals t, we have

$$\mathbf{Pr}[X_k - X_1 \leq -t] \leq \exp\left(\frac{-t^2}{2M^2(k-1)}\right). \tag{2.5}$$

2.3 Codes

General Notation. A code C of block length n over an alphabet Σ is a subset $C \subseteq \Sigma^n$. The rate $\mathcal{R}$ of C is defined to be $\frac{\log|C|}{n \log|\Sigma|}$. The encoding function of a code is a map $\mathrm{Enc} : [|C|] \to \Sigma^n$ whose image equals C (with messages identified with $[|C|]$ in some canonical way), and the decoding function of a code is a map $\mathrm{Dec} : \Sigma^* \to C$. Alphabet sizes $|\Sigma|$ are denoted by k when relevant.

A code is encodable (decodable) in time $f(n)$ if, for all elements of $[|C|]$ (Σ^*), the map Enc (Dec) can be computed in time $f(n)$. A code is constructible in time $f(n)$ if descriptions of C, Dec, and Enc can be produced in time $f(n)$.

Code Concatenation. Just as in [4, 20, 38], our constructions use the idea of code concatenation: If $C_{\text{out}} \subseteq \Sigma_{\text{out}}^n$ is an "outer code" with encoding function $\mathrm{Enc}_{\text{out}}$, and $C_{\text{in}} \subseteq \Sigma_{\text{in}}^m$ is an "inner code" with encoding function $\mathrm{Enc}_{\text{in}} : \Sigma_{\text{out}} \to \Sigma_{\text{in}}^m$, then the concatenated code $C_{\text{out}} \circ C_{\text{in}} \subseteq \Sigma_{\text{in}}^{nm}$ is a code whose encoding function first applied $\mathrm{Enc}_{\text{out}}$ to the message, and then applied Enc_{in} to each symbol of the resulting outer codeword.

Throughout this thesis, we let n denote the block length of a code, unless we deal with a concatenated code, in which case n denotes the block length of the outer code, m or L denotes the block length of the inner code, and N denotes the block length of the entire code.

In a concatenated code with outer code length n and inner code length L, we can identify a deletion pattern on the entire codeword $\tau \in \mathcal{D}(nL)$ as the "concatenation" of n deletion patterns $\tau_1 \frown \cdots \frown \tau_n$, one for each inner codeword. To be precise, for all $\tau \in \mathcal{D}(nL)$, there exists $\tau_i \in \mathcal{D}(L)$ such that $\tau = \cup_{i=1}^n \{j + (i-1)L : j \in \tau_i\}$, and we denote this by $\tau = \tau_1 \frown \cdots \frown \tau_n$. Using this notation, we refer to τ_i as an *inner code deletion pattern*.

Reed Solomon Codes. Reed Solomon codes are useful codes that reply on strong properties of polynomials. A (canonical) Reed Solomon code over $\mathbb{F}_q$ has block length $n = q$ and has message set consisting of all degree-less-than-k polynomials over $\mathbb{F}_q$, for some fixed k. A polynomial $f \in \mathbb{F}_q[X]$ is encoded by its evaluation at every point in $\mathbb{F}_q$: if we enumerate the elements of $\mathbb{F}_q$ as $\alpha_1, \dots, \alpha_q$, we have $\mathrm{Enc} : f \mapsto f(\alpha_1)f(\alpha_2)\dots f(\alpha_q)$. By properties of polynomials, an (n, k) Reed Solomon code can correct up to $(n-k)/2$ errors and erasures. This decoding can be done efficiently, e.g. by the Welch-Berlekamp algorithm in $O(n^3)$.

List Decoding. List decoding is a powerful technique in coding theory. A normal (unique) decoding algorithm is required to return the exact codeword, but a list decoding algorithm is allowed to return a list of codewords containing the correct codeword.

Reed Solomon codes can be efficiently list decoded. Sudan's list decoding algorithm takes an (n, k) Reed Solomon code under up to $n - \sqrt{2nk}$ errors with list size at most $\ell = \sqrt{2n/k}$. In other words, given the parameters n, t, k and n pairs $(\alpha_i, y_i) \in \mathbb{F}_q^2$, the algorithm finds all degree-at-most-k polynomials $f \in \mathbb{F}[X]$ such that $|\{i | f(\alpha_i) = y_i\}| \geq t$. The algorithm guarantees that the number of such polynomials is at most ℓ and runs in time $O(n^2)$.

Adversarial insertions, deletions, and insertions and deletions. The following lemma, originally due to Levenshtein [32], shows that, existentially, coding against adversarial deletions, adversarial insertions, and adversarial insertions and deletions are equally difficult. We present the lemma with proof for completeness.

Lemma 2.3.1. *Let $C \subseteq [k]^n$, and let $t < n$ be a positive integer. The following are equivalent.*

1. $LCS(C) \leq n - t - 1$.
2. *C can correct up to t adversarial insertions*
3. *C can correct up to t adversarial deletions*
4. *C can correct up to t adversarial insertions and deletions.*

Proof. The implications $4 \implies 2$ and $4 \implies 3$ are trivial. We prove $1 \implies 4$ and $3 \implies 1$. The proof of $2 \implies 1$ is the same idea as the proof of $3 \implies 1$.

Suppose 4 is false. That is, there exist distinct $c, c' \in C$ such that we can apply t insertions and deletions to each of c and c' to obtain the same string s. Color each bit of the strings c and c' blue, and apply i insertions and $t - i$ deletions to c and c' such that each inserted bit is colored black, to obtain two identical but possibly differently colored copies of s, which we refer to as s and s'. Note that s and s' each have $n - t + i$ blue bits and length $n - t + 2i$. Thus, s and s' are both colored blue in at least $2(n - t + i) - (n - t + 2i) = n - t$ of the $n - t + 2i$ positions. These blue bits form a common subsequence of c and c' of length at least $n - t$. This proves $1 \implies 4$.

Now suppose 1 is false. That is, there exists distinct $c, c' \in C$ such that $\text{LCS}(c, c') \geq n - t$. Thus, thus is some common subsequence s of c and c' such that $|s| = n - t$. Hence, an adversary, upon seeing that the code is c, can apply t deletions to c to obtain s, in which case the decoder cannot tell whether the received word is c or c'. We conclude C cannot correct up to t deletions. This proves $3 \implies 1$. The proof of $2 \implies 1$ is similar. $\square$

Chapter 3

Adversarial insertions and deletions

3.1 Introduction

In adversarial deletions, the adversary is allowed to delete up to pn bits with full knowledge of a codeword and codebook. A code C is decodable against pn adversarial deletions if and only if, for any two distinct codewords c and c' in C, it is impossible to apply pn (possibly different) deletions to c and c' and obtain the same result. By Lemma 2.3.1, this is equivalent to the condition that the longest common subsequence between any two codewords of C has less than $(1-p)n$ bits. This condition also ensures that C is capable of correcting any combination of adversarial insertions and deletions totaling pn in number.

In this chapter, we first overview existing results against coding against adversarial deletions and insertions in §3.2. In §3.3, we then summarize our new results from [16] on efficiently coding against adversarial insertions and deletions in high rate and high noise regimes. We then present our constructions. We present our high rate construction in §3.4. Our two high noise constructions follow a general framework given in §3.5. We present our specific results in §3.6 and §3.7.

3.2 Prior work

We first review the best known results on adversarial deletions. When $p \geq 1/2$, the adversary can delete $n/2$ bits that includes either all the 0's or all the 1's in the codeword, resulting in just two possible received sequences. Therefore, it is impossible to correct an adversarial deletion fraction of $1/2$ with rate bounded away from 0, and the zero-rate threshold $p_0^{(adv)}$ is at most $1/2$. Rather remarkably, we do not know if this trivial limit can be approached: are there codes $C \subseteq \{0,1\}^n$ of size $2^{\Omega_\epsilon(n)}$ decodable against $(1/2-\epsilon)n$ deletions for any $\epsilon > 0$? Or is there some p^* bounded away from $1/2$ such that any code $C \subseteq \{0,1\}^n$ that is decodable against $p^* n$ deletions must have size at most $2^{o(n)}$ (in which case, we have $p_0^{(adv)} \leq p^*$)? This was explicitly raised as a key open problem in [33]. Upper bounds on the asymptotic rate function in terms of the deletion fraction were obtained in [28], improving in some respects Levenshtein's bounds [33]. New upper bounds on code size for a fixed number of deletions that improve over [32] were obtained in [8]. A more combinatorially specific reframing of the question might ask, how does the maximum number of deletions that a code C can correct vary as $|C|$ increases? Bukh and Ma [3] provide partial results, establishing that when $|C| = O(\log n / \log\log n)$, the maximum correctable fraction is at most

$n/2 - \frac{1}{\text{poly}\,|C|} n^{1-1/|C|}$.

Turning to constructions of deletion codes, Schulman and Zuckerman [38] construct constant-rate binary codes that are efficiently decodable from a small constant fraction of worst-case insertions and deletions and can also handle a small fraction of transpositions. Kash et al. [25] proved that randomly chosen codes of small enough rate $R > 0$ can correctly decode against pn adversarial deletions when $p \leq 0.17$. Even non-constructively, this remained the best achievability result (in terms of correctable deletion fraction) until recently. Bukh, Guruswami, and Håstad [4] improved this and showed that there are binary codes of rate bounded away from 0 decodable against pn adversarial deletions for any $p < \sqrt{2} - 1$. In other words, they showed the zero-rate threshold, $p_0^{(adv)}$, is at least $\sqrt{2} - 1$. They extend this result to codes over constant-sized alphabets, showing that, over alphabets of size k, there are codes with rate bounded away from 0 decodable against pn adversarial deletions for $p < 1 - \frac{2}{k+\sqrt{k}}$. Furthermore, they gave an efficient construction of such codes along with an efficient deletion correcting algorithm. In the binary case, closing the gap between $\sqrt{2} - 1$ and $\frac{1}{2}$ for $p_0^{(adv)}$ remains a tantalizing open problem.

The above constructions optimize the error fraction p for binary codes at a cost of the rate. We can also ask: can we improve the correctable error fraction for codes over larger alphabets? Additionally, can we optimize the rate of our binary codes? Note that over large alphabets, in particular those which can grow with the code length, one can add the coordinate position as a header to the codeword symbol, and reduce the deletion model to the simpler erasure model, at the expense of a negligible decrease in rate (due to the addition of the header). Thus, in general, coding for synchronization errors over larger alphabets should be easier than over constant or binary alphabets. Over large alphabets, Guruswami and Wang [20] proved that there exist codes over size $\text{poly}(1/\epsilon)$ alphabets and with rate $\Omega(\epsilon^2)$ that can be decoded from a $1 - \epsilon$ fraction of worst-case deletions. They also construct binary codes with rate $1 - \tilde{O}(\sqrt{\epsilon})$ that can be efficiently decoded from a constant fraction ϵ of worst case deletions for sufficiently small ϵ.

We construct binary codes with rate $1 - \tilde{O}(\sqrt{\epsilon})$ that can be efficiently decoded from a constant fraction ϵ of worst case *insertions and deletions* for sufficiently small ϵ, and codes over size $\text{poly}(1/\epsilon)$ alphabets with rate $\Omega(\epsilon^2)$ that can be decoded from a $1 - \epsilon$ fraction of worst case deletions. Haeupler and Shahrasbi [21] recently improved on these results, constructing, for $\delta \in (0, 1)$ and $\epsilon > 0$, binary codes with rate $1 - \delta - \epsilon$ and $\delta > 0$.

3.3 Results summary

Our work primarily builds off recent results by Guruswami, Bukh, Håstad, and Wang [4, 20], which address the construction and efficient decoding of codes for constant fractions of deletions. These works establish three results, providing families of codes with each of the following parameters.

1. Families with rate approaching 1 decoding a constant fraction of deletions
2. Families with constant rate decoding a fraction of deletions approaching 1
3. Families over a fixed alphabet of size k with constant rate and decoding a fraction of deletions approaching $1 - \frac{2}{k+\sqrt{k}}$ (In particular, one gets binary codes for correcting a deletion fraction approaching $\sqrt{2} - 1$.)

Over an alphabet of size k, it is impossible to have a constant rate code that corrects a $1 - \frac{1}{k}$ fraction of deletions. The last result establishes that the maximum correctable fraction of deletions of a constant rate code is $1 - \Theta(\frac{1}{k})$.

Combinatorially, decoding a given number of worst-case insertions and deletions is identical to decoding the same number of worst-case deletions. This is formally stated in Lemma 2.3.1. By 2.3.1, the codes provided in the three constructions must also be capable of decoding both insertions and deletions. The task that remains, and which our work addresses, is to construct codes in the *same parameter settings* that can efficiently correct a combination of insertions *and* deletions.

The regime under which errors are insertions and deletions is closely related to *edit-distance* (also known as Levenshtein distance), which measures errors of a code under insertions, deletions, and substitutions. A substitution can be viewed as a deletion followed by an insertion. Thus, all results established in the insertion and deletion regime, both constructive and algorithmic, hold in the edit-distance regime when the number of errors is cut in half, and therefore in the traditional coding theory setting in which the only errors are substitutions. The edit-distance is a more challenging model, however; while the Gilbert-Varshamov bound gives codes over size k alphabets that can correct up to a fraction of substitutions approaching $\frac{1}{2}(1 - \frac{1}{k})$, the question of whether there exist positive rate codes capable of correcting a deletion fraction approaching $1 - \frac{1}{k}$ is still open.

These are the efficiently decodable code constructions in the deletion-only regime that we are generalizing to the insertion/deletion regime.

1) A binary code family of rate $1 - \tilde{O}(\sqrt{\epsilon})$ that can be efficiently decoded from an ϵ fraction of worst-case deletions, for all ϵ smaller than some absolute constant $\epsilon_0 > 0$. Furthermore, the codes are constructible, encodable, and decodable, in time $N^{\text{poly}(1/\epsilon)}$, where N is the block length. [Theorem 4.1 from [20]]

2) For any $\epsilon > 0$, a code family over an alphabet of size $\text{poly}(1/\epsilon)$ and rate $\Omega(\epsilon^2)$ that can be decoded from a $1 - \epsilon$ fraction of worst-case deletions. Furthermore, this code is constructible, encodable, and decodable in time $N^{\text{poly}(1/\epsilon)}$. [Theorem 3.1 from [20]]

3) For all integers $k \geq 2$ and all $\epsilon > 0$, a code family over alphabet size k of positive rate $r(k, \epsilon) > 0$ that can be decoded from a $1 - \frac{2}{k+\sqrt{k}} - \epsilon$ fraction of worst-case deletions in $O_{k,\epsilon}(N^3 (\log N)^{O(1)})$ time.

Our work constructs the following three families of codes.

1. alphabet size: 2, rate: $1 - \tilde{O}(\sqrt{\epsilon})$, insertion/deletion fraction: ϵ, decoding time: $N^{\text{poly}(1/\epsilon)}$. (Thm. 3.4.2)
2. alphabet size: $\text{poly}(1/\epsilon)$, rate: $\Omega(\epsilon^5)$, insertion/deletion fraction: $1 - \epsilon$, decoding time: $N^{\text{poly}(1/\epsilon)}$. (Thm. 3.6.3)
3. alphabet size: $k \geq 2$, rate: $(\epsilon/k)^{\text{poly}(1/\epsilon)}$, insertion/deletion fraction: $1 - \frac{2}{k+\sqrt{k}} - \epsilon$, decoding time: $O_{k,\epsilon}(N^3 \text{ poly} \log(N))$. (Thm. 3.7.2)

Remark. Theorem 3.6.3 gives constant rate codes that decode from a $1 - \epsilon$ fraction of insertions/deletions. This also follows as a corollary from Theorem 3.7.2. However, the rate of the construction in Theorem 3.7.2 is $(\epsilon/k)^{\text{poly}(1/\epsilon)}$, which is far worse than $\text{poly}(\epsilon)$. The main point of 3.7.2 is to highlight the near-tight trade-off between alphabet size and insertion/deletion fraction.

Remark. At the expense of slightly worse parameters, the construction and decoding complexities in Theorems 3.4.2 and 3.6.3 can be improved to $\text{poly}(N) \cdot (\log N)^{\text{poly}(1/\epsilon)}$. See Theorems 3.4.7 and 3.6.4.

Theorems 3.6.3 and 3.7.2 use the powerful idea of list decoding, exemplified in [4]. A normal decoding algorithm is required to return the exact codeword, but a list decoding algorithm is allowed to return a list of codewords containing the correct codeword. The codes for both theorems are decoded by first applying a list decoding algorithm, and then noting that if the easier list decoding is guaranteed to succeed (that is, returns a list containing the correct codeword), one can simply pass through the resulting list and choose the unique codeword that has sufficiently small distance from the received word. The codeword will be unique because the codes constructed are provably decodable under the required number of insertion/deletions according to the results in [4, 20].

The extent of difference between the insertion/deletion decoding algorithms and their deletion-only analogues varies depending on the parameter setting. For a $1 - \frac{2}{k+\sqrt{k}} - \epsilon$ fraction of insertions/deletions, the decoding algorithm uses the same list decoding approach as the deletion-only decoding algorithm in [4]. For a $1 - \epsilon$ fraction of insertions/deletions, we adopt the list decoding approach that in fact simplifies the construction presented in [20]. For achieving a rate of $1 - \epsilon$, we use the same code as in [20] with different parameters, but considerably more bookkeeping is done to provide a provably correct decoding algorithm. In particular, both Theorem 4.1 from [20] and Theorem 3.4.2 place chunks of 0s between inner codewords. However, while identifying buffers in the received word in the deletion-only case merely requires identifying long runs of 0s, identifying buffers in the insertion/deletion case requires identifying strings of fixed length with sufficiently small fraction of 1s.

3.4 High rate

Lemma 3.4.1 (Proposition 2.5 of [20]). *Let $\delta, \beta \in (0,1)$. Then, for every m, there exists a code $C \subseteq \{0,1\}^m$ of rate $R = 1 - 2h(\delta) - O(\log(\delta m)/m) - 2^{-\Omega(\beta m)/m}$ such that*

- *for every string $s \in C$, every interval of length βm in s, contains at least $\beta m/10$ 1's,*
- *C can be corrected from a δ fraction of worst-case deletions, and*
- *C can be found, encoded, and decoded in time $2^{O(m)}$.*

Theorem 3.4.2. *There exists a constant $\epsilon_0 > 0$ such that the following holds. Let $0 < \epsilon < \epsilon_0$. There is an explicit binary code $C \subseteq \{0,1\}^N$ with rate $1 - \tilde{O}(\sqrt{\epsilon})$ that is decodable from an ϵ fraction of insertions/deletions in $N^{\text{poly}(1/\epsilon)}$ time. Furthermore, C can be constructed and encoded in time $N^{\text{poly}(1/\epsilon)}$.*

Proof. With hindsight, let $\epsilon_0 = \frac{1}{121^2}$, and let $0 < \epsilon < \epsilon_0$. Consider the concatenated construction with the outer code being a Reed-Solomon code that can correct a $60\sqrt{\epsilon}$ fraction of errors and erasures. For each $1 \le i \le n$, we replace the ith coordinate c_i with the pair (i, c_i); to ensure that this doesn't affect the rate much, we take the RS code to be over $\mathbb{F}_{q^h}$, where $n = q$ is the block length and $h = 1/\epsilon$. We encode each outer symbol pair in the inner code, defined as follows.

The inner code is a good binary insertion/deletion code C_1 of block length m decoding a $\delta = 40\sqrt{\epsilon} < \frac{1}{2}$ fraction of insertions and deletions, such that every interval of length $\delta m/16$ in a codeword has at least $1/10$ fraction of 1s. This code can be found using Lemma 3.4.1. We also assume each codeword begins and ends with a 1.

Now take our concatenated Reed-Solomon code of block length mn, and between each pair of adjacent inner codewords of C_1, insert a *chunk* of δm 0s. This gives us our final code C with block length $N = nm(1 + \delta)$.

Lemma 3.4.3. *The rate of C is $1 - \tilde{O}(\sqrt{\epsilon})$.*

Proof. The rate of the outer RS code is $(1 - 120\sqrt{\epsilon})\frac{h}{h+1}$, and the rate of the inner code can be taken to be $1 - 2h(\delta) - o(1)$ by Lemma 3.4.1. Adding in the buffers reduces the rate by a factor of $\frac{1}{1+\delta}$. Combining these with our choice of δ gives us a total rate for C of $1 - \tilde{O}(\sqrt{\epsilon})$. □

Lemma 3.4.4. *The code C can be decoded from an ϵ fraction of insertions and deletions in time $N^{\mathrm{poly}(1/\epsilon)}$.*

Consider the following algorithm that runs in time $N^{\mathrm{poly}(1/\epsilon)}$ for decoding the received word:

1) Scan from the left of the received word. Every time we encounter a substring of length exactly δm with at most $\frac{1}{160}$ fraction of 1s (or $\delta m/160$ 1s), mark it as a *decoding buffer*. Then, continue scanning from the end of the buffer and repeat. This guarantees no two buffers overlap. This takes time $\mathrm{poly}(N)$.

2) Start with an empty set L. The buffers divide the received word into strings which we call *decoding windows*. For each decoding window, apply the decoder from Lemma 3.4.1 to recover a pair (i, r_i). If we succeed, add this pair to L. This takes $N^{\mathrm{poly}(1/\epsilon)}$ time.

3) If for any i, L contains multiple pairs with first coordinate i, remove all such pairs from L. L thus contains at most one pair (i, r_i) for each index i. Then apply the RS decoding algorithm to the string r whose ith coordinate is r_i if $(i, r_i) \in L$ and erased otherwise. This takes time $\mathrm{poly}(N)$.

Remark. In the deletion only case, the decoding buffers are runs of at least $\delta m/2$ contiguous zeros. Runs of consecutive zeros are obviously a poor choice for decoding buffers in the presence of insertions, as we can destroy any buffer with a constant number of insertions.

Note that the total number of insertions/deletions we can make is at most $(1+\delta)mn\epsilon < 2\epsilon mn$.

Suppose our received codeword is $s = u_1 y_1 u_2 \cdots u_{n'}$, where $y_1, \ldots, y_{n'-1}$ are the identified decoding buffers and $u_1, \ldots, u_{n'}$ are the decoding windows. Then consider a canonical mapping from characters of c to characters of s where u_i is mapped to by a substring t_i of c, y_i is mapped to by a string x_i, so that $c = t_1 x_1 \cdots t_{n'}$ and $\Delta(c, s) = \sum_{i=1}^{n'} \Delta(u_i, t_i) + \sum_{i=1}^{n'-1} \Delta(y_i, x_i)$.

With our canonical mapping, we can identify $\mathrm{LCS}(c, s)$ many characters in s with characters in c. Intuitively, these are the characters that are uncorrupted when we transform c into s using insertions and deletions. Call a received buffer y_i in s a *good decoding buffer* (or *good buffer* for short) if at least $\frac{3}{4}\delta m$ of its characters are identified with characters from a single chunk of δm 0s in c. Call a decoding buffer *bad* otherwise. Call a chunk of δm 0s in c *good* if at least $\frac{3}{4}\delta m$ of its zeros map to characters in single decoding buffer. Note that there is a natural bijection between good chunks in c and good decoding buffers in s.

Lemma 3.4.5. *The number of bad decoding buffers of s is at most $8\sqrt{\epsilon}n$.*

Proof. Suppose we have a bad buffer y_i. It either contains characters from at least two different chunks of δm 0s in c or contains at most $\frac{3\delta m}{4}$ characters from a single chunk.

In the first case, x_i must contain characters in two different chunks so its length must be at least m, so y_i must have been obtained from at least $m - \delta m > \delta m > 40\sqrt{\epsilon}m$ deletions from x_i.

In the second case, if x_i has length at most $\frac{7\delta m}{8}$ then the insertion/deletion distance between x_i and y_i is at least $\frac{\delta m}{8} = 5\sqrt{\epsilon}m$. Otherwise, x_i has at least $\frac{\delta m}{8}$ characters in some inner codeword of c, so x_i has at least $\frac{\delta m}{80}$ 1s, so we need at least $\frac{\delta m}{80} - \frac{\delta m}{160} = \frac{1}{4}\sqrt{\epsilon}m$ deletions to obtain y_i from x_i.

By a simple counting argument, the total number of bad buffers we can have is at most $\frac{2\epsilon mn}{\frac{1}{4}\sqrt{\epsilon}m} = 8\sqrt{\epsilon}n$. □

Lemma 3.4.6. *The number of good decoding buffers of s is at least $(1-8\sqrt{\epsilon})n$.*

Proof. It suffices to prove the number of good chunks of c is at least $(1-8\sqrt{\epsilon})n$. If a chunk is not mapped to a good buffer, at least one of the following is true.

1. The chunk is "deleted" by inserting enough 1s.
2. Part of the chunk is mapped to a bad buffer that contains characters from $t-1 \geq 1$ other chunks.
3. Part of the chunk is mapped to a bad buffer that contains no characters from other chunks.

In the first case, we need at least $\frac{\delta m}{160} = \frac{1}{4}\sqrt{\epsilon}m$ insertions to delete the chunk. In the second case, creating the bad buffer costs at least $(t-1)(m-\delta m) \geq \frac{t\delta m}{2}$ deletions, which is at least $20\sqrt{\epsilon}m$ deletions per chunk. In the third case, creating the bad buffer costs at least $\frac{1}{4}\sqrt{\epsilon}m$ edits by the argument in Lemma 3.4.5. Thus, we have at most $\frac{2\epsilon mn}{\frac{1}{4}\sqrt{\epsilon}m} = 8\sqrt{\epsilon}n$ bad chunks, so we have at least $(1-8\sqrt{\epsilon})n$ good chunks, as desired. □

Since there are at least $(1-8\sqrt{\epsilon})n$ good decoding buffers and at most $8\sqrt{\epsilon}n$ bad decoding buffers, there must be at least $(1-16\sqrt{\epsilon})n$ pairs of consecutive good decoding buffers. For any pair of consecutive good decoding buffers y_{j-1}, y_j in s, the corresponding two good chunks of δm 0s in c are consecutive unless there is at least one bad chunk in between the two good chunks, which happens for at most $8\sqrt{\epsilon}n$ pairs. Thus, there are at least $(1-24\sqrt{\epsilon})n$ pairs of consecutive good decoding buffers in s such that the corresponding good chunks of 0s in c are also consecutive.

Now suppose w is an inner codeword between two good chunks with corresponding consecutive good decoding buffers, y_{j-1}, y_j. The corresponding decoding window between the decoding buffers is u_j, mapped to from t_j, a substring of c. We claim that most such w are decoded correctly.

For all but $2\frac{2\epsilon mn}{\delta m/8} + 2\frac{2\epsilon mn}{\delta m/8} + \frac{2\epsilon mn}{\delta m/4} < 2\sqrt{\epsilon}n$ choices of j, we have $\Delta(x_{j-1}, y_{j-1}) \leq \frac{\delta m}{8}$, $\Delta(x_j, y_j) \leq \frac{\delta m}{8}$, and $\Delta(t_j, u_j) \leq \frac{\delta m}{4}$. When we have an inner codeword w and an index j such that all these are true, we have $|x_{j-1}|, |x_j| \leq \frac{9\delta m}{8}$, and each of x_{j-1}, x_j shares at least $\frac{3\delta m}{4}$ characters with one of the chunks of δm 0s neighboring w. It follows that x_{j-1}, x_j each contain at most $\frac{3\delta m}{8}$ characters of w. Additionally, by the definition of a good chunk, u_j contains at most $\frac{\delta m}{4}$ characters in each of the chunks neighboring w. Thus, we have $\Delta(w, t_j) \leq \frac{3\delta m}{4}$, in which case, $\Delta(w, t_j) \leq \Delta(w, t_j) + \Delta(t_j, u_j) \leq \delta m$. Thus, for at least $(1-24\sqrt{\epsilon})n - 2\sqrt{\epsilon}n = (1-26\sqrt{\epsilon})n$ inner words w, there exists $j \in \{1, \ldots, n'\}$ such that $\Delta(w, u_j) \leq \delta m$.

Therefore, our algorithm detects at least $(1-26\sqrt{\epsilon})n$ correct pairs (i, r_i). Since our algorithm detects at most $(1+8\sqrt{\epsilon})n$ pairs total, we have at most $34\sqrt{\epsilon}n$ incorrect pairs. Thus, after removing conflicts, we have at least $(1-60\sqrt{\epsilon})n$ correct values, so our Reed Solomon decoder will succeed. □

Remark. Our decoding algorithm succeeds as long as the inner code can correct up to a δ fraction of insertions/deletions and consists of codewords such that every interval of length $\delta m/16$ has at least $1/10$ fraction of 1s. The time complexity of Theorem 3.4.2 can be improved using a more efficient inner code, at the cost of reduction in rate.

Because of the addition of buffers, the code of Theorem 3.4.2 may not be dense enough to use as an inner code. The inner code needs to have $1/10$ fraction of 1s for every interval of length

$\delta m/16$. However, we can modify the construction of the inner concatenated code so that the inner codewords of the inner code in Theorem 3.4.2 have at least $1/5$ fraction of 1s in every interval of length $\delta m/16$. This guarantees that the inner codewords of our two level construction have sufficiently high densities of 1s. This is summarized in the following theorem.

Theorem 3.4.7. *There exists a constant $\epsilon_0 > 0$ such that the following holds. Let $\epsilon_0 > \epsilon > 0$. There is an explicit binary code $C \subseteq \{0,1\}^N$ that is decodable from an ϵ fraction of insertions/deletions with rate $1 - \tilde{O}(\sqrt[4]{\epsilon})$ in time $\text{poly}(N) \cdot (\log N)^{\text{poly}(1/\epsilon)}$.*

3.5 High noise

Because our decoding algorithms for the $1-\epsilon$ and $1 - \frac{2}{k+\sqrt{k}} - \epsilon$ insertion/deletion constructions use the same list decoding technique, we abstract out the technical part of the decoding algorithm with the following theorem.

Theorem 3.5.1. *Let C be a code over alphabet of size k and length $N = nm$ obtained by concatenating a Reed-Solomon C_{out} of length n with an inner code C_{in} of length m. Suppose C_{out} has rate r and is over $\mathbb{F}_q$ with $n = q$. Suppose $C_{in} : [n] \times \mathbb{F}_q \to [k]^m$ can correct a $1-\delta$ fraction of insertions and deletions in $O(t(n))$ for some function t. Then, provided C is (combinatorially) decodable under up to $1 - \delta - 4r^{1/4}$ fraction of insertions and deletions, it is in fact decodable in time $O(N^3 \cdot (t(N) + \text{polylog}\, N))$.*

Proof. Let $\gamma = 4r^{1/4}$. Consider the following algorithm, which takes as input a string s that is the result of changing a codeword c under a fraction $\leq (1-\delta-\gamma)$ of insertions/deletions.

1) $\mathcal{J} \leftarrow \emptyset$.

2) For each $0 \leq j \leq \lceil \frac{2n}{\gamma} \rceil$ $1 \leq j' \leq \lceil \frac{4}{\gamma} \rceil$, do the following.

a) Let $\sigma_{j,j'}$ denote the substring from indices $\frac{\gamma m}{2} j$ to $\frac{\gamma m}{2}(j+j')$.

b) By brute force search over $\mathbb{F}_q \times \mathbb{F}_q$, find all pairs (α, β) such that $\Delta(\text{Enc}_{C_{in}}((\alpha,\beta)), \sigma_{j,j'}) \leq (1-\delta)m$. If exactly one such pair (α,β) exists, then add (α,β) to $\mathcal{J}$.

3) Find the list, call it $\mathcal{L}$, of all polynomials $p \in \mathbb{F}_q[X]$ of degree less than rn such that $|\{(\alpha, p(\alpha)) | \alpha \in \mathbb{F}_q\} \cap \mathcal{J}| \geq \frac{\gamma n}{2}$.

4) Find the unique polynomial in $\mathcal{L}$, if any, such that the insertion/deletion distance between its encoding under C and s is at most $(1-\gamma-\delta)N$.

CORRECTNESS. Break the codeword $c \in [k]^{nm}$ of the concatenated code C into n inner blocks, with the ith block $b_i \in [k]^m$ corresponding to the inner encoding of the ith symbol $(\alpha_i, f(\alpha_i))$ of the outer Reed-Solomon code. For some fixed canonical way of forming s out of c, let s_i be the block formed out of b_i, so that $s_1, \dots, s_n$ partition the string s. Call an index i *good* if it can be obtained from b_i by at most $(1-\delta-\frac{\gamma}{2})m$ insertions or deletions, and *bad* otherwise. The number of bad indices is at most $\frac{(1-\delta-\gamma)mn}{(1-\delta-\gamma/2)m} \leq (1-\frac{\gamma}{2})n$, so the number of good indices is at least $\frac{\gamma n}{2}$.

For any good index a, there exists some $\sigma_{j,j'}$ such that s_a is a substring of $\sigma_{j,j'}$ and $0 < |\sigma_{j,j'}| - |s_a| < \frac{\gamma m}{2}$. Since a is good, the insertion/deletion distance between b_a and s_a is at most $(1-\delta-\gamma/2)m$, and the insertion/deletion distance between s_a and $\sigma_{j,j'}$ is less than $\gamma m/2$, so the insertion/deletion distance between b_a and $\sigma_{j,j'}$ is at most $(1-\delta)m$. Since C_{in} can handle up to $(1-\delta)m$ insertions and deletions, it follows that b_a is the unique codeword of C_{in} such that $\Delta(\text{Enc}_{C_{in}}(b_a), \sigma_{j,j'}) \leq (1-\delta)m$. Since b_a is the encoding of $(\alpha_a, f(\alpha_a))$ under C_{in}, we conclude that for any good index a, the pair $(\alpha_a, f(\alpha_a))$ will be included in $\mathcal{J}$. In particular, $\mathcal{J}$ will have at least $\gamma n/2$ such pairs, so the correct f will be in $\mathcal{L}$.

We now check that step 3 of the algorithm will succeed. We have $|\mathcal{J}| \leq \frac{2n}{\gamma} \cdot \frac{4}{\gamma} = \frac{8n}{\gamma^2}$, and Sudan's list decoding algorithm will give a list of degree-less-than-rn polynomials over $\mathbb{F}_q$ such that $(\alpha, p(\alpha)) \in \mathcal{J}$ for more than $\sqrt{2(rn)|\mathcal{J}|}$ values of $\alpha \in \mathbb{F}_q$ [40]. Furthermore, this list will have at most $\sqrt{2|\mathcal{J}|/(rn)}$ elements. For our choice of γ, we have $\gamma n/2 > \sqrt{\frac{16rn^2}{\gamma^2}} \geq \sqrt{2(rn)|\mathcal{J}|}$, so the list decoding will succeed.

By above, there will be at least one polynomial in $\mathcal{L}$ such that the longest common subsequence of its encoding with s has length at least $(\gamma + \delta)m$, namely the correct polynomial f. Since we assumed C can decode up to a $1 - \delta - \gamma$ fraction of insertions/deletions, all other polynomials in $\mathcal{L}$ will have longest common subsequence with s smaller than $(\gamma + \delta)m$. Thus our algorithm returns the correct f.

RUNTIME. We have $O(n) \leq O(N)$ intervals $\sigma_{j,j'}$ to check, and each one brute forces over n^2 terms of $\mathbb{F}_q \times \mathbb{F}_q$. Encoding takes time $O(t(n)) \leq O(t(N))$ by assumption and computing the longest common subsequence takes $O(m^2) = O(\log^2 N)$ time, so in total the second step of the algorithm takes $O(N^3(t(N) + \log^2 N))$ time. Since $|\mathcal{J}| \leq O(N)$ for sufficiently large N, the Reed-Solomon list decoding algorithm can be performed in time $O(N^2)$, see for instance [37]. There are a constant number of polynomials to check at the end, and each one takes $O(N^2)$ time using the longest common subsequence algorithm. Thus, the overall runtime of the algorithm is $O(N^3(t(N) + \operatorname{polylog} N))$. □

3.6 High noise: Decoding against $1 - \epsilon$ insertions/deletions

Lemma 3.6.1. *Suppose we have a code C which is the concatenation of an outer code C_{out} of length n with an inner code C_{in} of length m. Suppose further that for some $\Delta, \delta \in (0, 1)$, we have* $\mathrm{LCS}(C_{out}) \leq \Delta n, \mathrm{LCS}(C_{in}) \leq \delta m$. *Then* $\mathrm{LCS}(C) \leq (\Delta + 2\delta)nm$.

Lemma 3.6.2 ($\theta = 1/3$ case of Corollary 2.6 of [20]). *Let $1/2 > \epsilon > 0$, and k be a positive integer. For every m, there exists a code $C \subseteq [k]^m$ of rate $R = \epsilon/3$ that can correct a $1 - \epsilon$ fraction of insertions/deletions in time $k^{O(m)}$, provided $k \geq 64/\epsilon^3$.*

Theorem 3.6.3. *For any $\epsilon > 0$, there exists a family of codes over an alphabet of size* $\operatorname{poly}(1/\epsilon)$ *and rate $\Omega(\epsilon^5)$ that can be efficiently decoded from a $1 - \epsilon$ fraction of insertions/deletions. Furthermore, this code is constructible, encodable, and decodable in time $N^{\operatorname{poly}(1/\epsilon)}$.*

Proof. Let $n = q, m = 24 \log q/\epsilon$, and $k = O(1/\epsilon^3)$. By Lemma 3.6.2, we can construct by brute force a code $C_1 : n \times \mathbb{F}_q \to [k]^m$ that can be decoded from $1 - \epsilon/4$ fraction of worst-case insertions and deletions. We can concatenate C_1 with an outer Reed-Solomon code of rate $(\epsilon/8)^4$.

The rate of the inner code is $\Omega(\epsilon)$, and the rate of the outer code is $\Omega(\epsilon^4)$, so the total rate is $\Omega(\epsilon^5)$.

By Lemma 3.6.1, $\mathrm{LCS}(C) \leq (\epsilon/8)^4 + 2(\epsilon/4) < \epsilon$, so C is capable of decoding up to $1 - \epsilon$ fraction of insertions and deletions. Encoding in C_1 is done by brute force in time $N^{\operatorname{poly}(1/\epsilon)}$, so by Theorem 3.5.1, C is capable of decoding up to $1 - \epsilon/4 - 4((\epsilon/8)^4)^{1/4} > 1 - \epsilon$ fraction of worst-case insertions and deletions in time $O(N^3(N^{\operatorname{poly}(1/\epsilon)} + \operatorname{poly}\log N)) = N^{\operatorname{poly}(1/\epsilon)}$, as desired. □

Remark. Our construction only requires that the inner code can be decoded from $1 - \epsilon/4$ fraction of worst-case insertions and deletions. By using the concatenated code of Theorem 3.6.3 as the inner code of the same construction (thus giving us two levels of concatenation), we can reduce the time complexity significantly, at the cost of a polynomial reduction in other parameters of the code, as summarized below.

Theorem 3.6.4. *For any $\epsilon > 0$, there exists a family of constant rate codes over an alphabet of size $\mathrm{poly}(1/\epsilon)$ and rate $\Omega(\epsilon^9)$ that can be decoded from a $1-\epsilon$ fraction of insertions/deletions. Furthermore, this code is constructible, encodable, and decodable in time $\mathrm{poly}(N) \cdot (\log N)^{\mathrm{poly}(1/\epsilon)}$.*

3.7 High noise: Decoding against $1 - \frac{2}{k+\sqrt{k}} - \epsilon$ insertions/deletions

First, we summarize an existence result from [4].

Lemma 3.7.1 (Theorem 18 of [4])**.** *Fix an integer $k \geq 2$ and $\gamma > 0$. Then there are infinitely many N for which there is a concatenated Reed Solomon code $C \subseteq [k]^N$ that has outer rate at least $\gamma/2$, has total rate at least $(\gamma/k)^{O(\gamma^{-3})}$, is decodable under $1 - \frac{2}{k+\sqrt{k}} - \gamma$ fraction of insertions and deletions, has an inner code decodable under $1 - \frac{2}{k+\sqrt{k}} - \gamma/4$ insertions and deletions, and is constructible in time $O(N \log^2 N)$.*

Theorem 3.7.2. *Fix an integer $k \geq 2$ and $\epsilon > 0$. For infinitely many and sufficiently large N, there is an explicit code $C \subseteq \{0,1\}^N$ with rate $r(k,\epsilon) = (\epsilon/k)^{O(\epsilon^{-12})}$ over a size k alphabet that can be decoded from a $1 - \frac{2}{k+\sqrt{k}} - \epsilon$ fraction of worst-case insertions and deletions in time $O_{k,\epsilon}(N^3 \,\mathrm{polylog}(N))$. Furthermore, this code is constructible in time $O_{k,\epsilon}(N \log^2 N)$.*

Proof. Consider the codes C given by Lemma 3.7.1 with $\gamma = 2(\epsilon/5)^4$. C has outer rate at least $\gamma/2 = (\epsilon/5)^4$ and total rate at least $(\gamma/k)^{O(\gamma^{-3})}$. Furthermore, C can decode up to $1 - \frac{2}{k+\sqrt{k}} - \gamma$ fraction of insertions/deletions, and the inner code of C can decode $1 - \frac{2}{k+\sqrt{k}} - \gamma/4$ fraction of insertions/deletions. Thus, by Theorem 3.5.1, C can efficiently decode up to $1 - \frac{2}{k+\sqrt{k}} - \gamma/4 - 4(\gamma/2)^{1/4} > 1 - \frac{2}{k+\sqrt{k}} - \epsilon$ fraction of insertions/deletions. $\square$

Chapter 4

Random deletions

4.1 Introduction

We consider the problem of designing error-correcting codes for reliable and efficient communication on the binary deletion channel. The *binary deletion channel* (BDC) *deletes* each transmitted bit independently with probability p, for some $p \in (0, 1)$ which we call the *deletion probability*. Crucially, the location of the deleted bits are *not* known at the decoder, who receives a *subsequence* of the original transmitted sequence. The loss of synchronization in symbol locations makes the noise model of deletions challenging to cope with. As one indication of this, we still do not know the channel capacity of the binary deletion channel. Quoting from the first page of Mitzenmacher's survey [35]: "Currently, we have no closed-form expression for the capacity, nor do we have an efficient algorithmic means to numerically compute this capacity." This is in sharp contrast with the noise model of bit erasures, where each bit is independently replaced by a '?' with probability p (the binary erasure channel (BEC)), or of bit errors, where each bit is flipped independently with probability p (the binary symmetric channel (BSC)). The capacity of the BEC and BSC equal $1-p$ and $1-h(p)$ respectively, and we know codes of polynomial complexity with rate approaching the capacity in each case.

In §4.2, we survey the existing literature on the binary deletion channel and similar models of synchronization errors. We then present a result from [18] in §4.3 and a proof in §4.4. We then conclude in §4.5 with a discussion of possible alternatives to coding against the BDC and potential limitations of such approaches.

4.2 Prior work

The capacity of the binary deletion channel is clearly at most $1-p$, the capacity of the simpler binary erasure channel. Diggavi and Grossglauser [11] establish that the capacity of the deletion channel for $p \leq \frac{1}{2}$ is at least $1-h(p)$. Kalai, Mitzenmacher, and Sudan [23] proved this lower bound is tight as $p \to 0$, and Kanoria and Montanari [24] determined a series expansion that can be used to determine the capacity exactly. Turning to large p, Rahmati and Duman [36] prove that the capacity is at most $0.4143(1-p)$ for $p \geq 0.65$. Drinea and Mitzenmacher [12, 13] proved that the capacity of the BDC is at least $(1-p)/9$, which is within a constant factor of the upper bound. In particular, the capacity is positive for every $p < 1$, which is perhaps surprising. The asymptotic behavior of the capacity of the BDC at both extremes of $p \to 0$ and $p \to 1$ is thus known.

This work is concerned with *constructive* results for coding for the binary deletion channel. That is, we seek codes that can be constructed, encoded, and decoded from deletions caused by the BDC, in polynomial time. Recently, there has been good progress on codes for *adversarial* deletions, including constructive results. Here the model is that the channel can delete an arbitrary subset of pn bits in the n-bit codeword. A code capable of correcting pn worst-case deletions can clearly also correct deletions caused by a BDC with deletion probability $(p - \epsilon)$ with high probability, so one can infer results for the BDC from some results for worst-case deletions. For small p, Guruswami and Wang [20] constructed binary codes of rate $1 - O(\sqrt{p})$ to efficiently correct a p fraction worst-case deletions. So this also gives codes of rate approaching 1 for the BDC when $p \to 0$. For larger p, Kash et al. [25] proved that randomly chosen codes of small enough rate $R > 0$ can correctly decode against pn adversarial deletions when $p \leq 0.17$. Even non-constructively, this remained the best achievability result in terms of correctable deletion fraction until the recent work of Bukh, Guruswami, and Håstad [4] who constructed codes of positive rate efficiently decodable against pn adversarial deletions for any $p < \sqrt{2} - 1$. For adversarial deletions, it is impossible to correct a deletion fraction of $1/2$, whereas the capacity of the BDC is positive for all $p < 1$. So solving the problem for the much harder worst-case deletions is not a viable approach to construct positive rate codes for the BDC for $p > 1/2$.

To the best of our knowledge, explicit efficiently decodable code constructions were not available for the binary deletion channel for arbitrary $p < 1$. We present such a construction in this work. Our rate is worse than the $(1-p)/9$ achieved non-constructively, but has asymptotically the same dependence on p for $p \to 1$.

One work that considers efficient recovery against random deletions is by Yazdi and Dolecek [42]. In their setting, two parties Alice and Bob are connected by a two-way communication channel. Alice has a string X, Bob has string Y obtained by passing X through a binary deletion channel with deletion probability $p \ll 1$, and Bob must recover X. They produce a polynomial-time synchronization scheme that transmits a total of $O(pn \log(1/p))$ bits and allows Bob to recover X with probability exponentially approaching 1.

For other models of random synchronization errors, Kirsch and Drinea [26] prove information capacity lower bounds for channels with i.i.d deletions and duplications. Fertonani et al. [15] prove capacity bounds for binary channels with i.i.d insertions, deletions, and substitutions.

For deletion channels over non-binary alphabets, Rahmati and Duman [36] prove a capacity upper bound of $C_2(p) + (1-p)\log(|\Sigma|/2)$, where $C_2(p)$ denotes the capacity of the binary deletion channel with deletion probability p, when the alphabet size $|\Sigma|$ is even. In particular, using the best known bound for $C_2(p)$ of $C_2(p) \leq 0.4143(1-p)$, the upper bound is $(1-p)(\log|\Sigma| - 0.5857)$.

4.3 New result on efficiently coding against random deletions

We consider the problem of constructing explicit codes with efficient decoding for the binary deletion channel. Here our result is the following. Our rate is worse than the $(1-p)/9$ achieved non-constructively, but has asymptotically the same dependence on p for $p \to 1$.

Theorem 4.3.1. *Let $p \in (0,1)$. There is an explicit a family of binary codes that (1) has rate $(1-p)/110$, (2) is constructible in polynomial time, (3) encodable in time $O(N)$, and (3) decodable with high probability on the binary deletion channel with deletion probability p in time $O(N^2)$.*

(Here N is the block length of the code)

Our construction concatenates a high rate outer code over a large alphabet that is efficiently decodable against a small fraction of *adversarial* insertions and deletions, with a good inner binary code. For the outer code, we can use the recent construction of [21]. To construct the inner code, we first choose a binary code correcting a small fraction of adversarial deletions. By concentration bounds, duplicating bits of a codeword in a disciplined manner is effective against the random deletion channel, so we, for some constant B, duplicate every bit of the binary code $B/(1-p)$ times. We further ensure our initial binary code has only runs of length 1 and 2 to maximize the effectiveness of duplication. We add small buffers of 0s between inner codewords to facilitate decoding.

One might wonder whether it would be possible to use Drinea and Mitzenmacher's existential result [12, 13] of a $(1-p)/9$ capacity lower bound as a black box inner code to achieve a better rate together with efficient decodability. We discuss this approach in §4.5 and elaborate on what makes such a construction difficult to implement.

4.4 Construction

We present a family of constant rate codes that decodes with high probability on a binary deletion channel with deletion fraction p (BDC$_p$). These codes have rate $c_0(1-p)$ for an absolute positive constant c_0, which is within a constant of the upper bound $(1-p)$, which even holds for the erasure channel. By Drinea and Mitzenmacher [12] the maximum known rate of a non-efficiently correctable binary deletion channel code is $(1-p)/9$.

The construction is based on the intuition that deterministic codes are better than random codes for the deletion channel. Indeed, for adversarial deletions, length n random codes correct at most $0.22n$ deletions [25], while explicitly constructed codes can correct close to $(\sqrt{2}-1)n$ deletions [4].

We begin by borrowing a result from [20].

Lemma 4.4.1 (Corollary of Lemma 2.3 of [20])**.** *For every binary string $c \in \{0,1\}^m$, there are at most $\delta m\binom{m}{(1-\delta)m}^2$ strings $c' \in \{0,1\}^m$ such that it is possible to apply δm deletions to c and c' and obtain the same result.*

The next lemma gives codes against a small fraction of adversarial deletions with an additional run-length constraint on the codewords.

Lemma 4.4.2. *Let $\delta > 0$. There exists a length m binary code of rate $\mathcal{R} = 0.6942 - 2h(\delta) - O(\log(\delta m)/m)$ correcting a δ fraction of adversarial insertions and deletions such that each codeword contains only runs of size 1 and 2. Furthermore this code is constructible in time $\tilde{O}(2^{(0.6942+\mathcal{R})m})$.*

Proof. It is easy to show that the number of codewords with only runs of 1 and 2 is F_m, the mth Fibonacci number, and it is well known that $F_m = \varphi^m + o(1) \approx 2^{0.6942m}$ where φ is the golden ratio. Now we construct the code by choosing it greedily. Each codeword is confusable with at most $\delta m\binom{m}{(1-\delta)m}^2$ other codewords, so we can choose at least

$$\frac{2^{0.6942m}}{\delta m\binom{m}{(1-\delta)m}^2} = 2^{m(0.6942-2h(\delta)-O(\log(\delta m)/m))} \tag{4.1}$$

codewords.

We can find all words of length m whose run lengths are only 1 and 2 by recursion in time $O(F_m) = O(2^{0.6942m})$. Running the greedy algorithm, we need to, for at most $F_m \cdot 2^{\mathcal{R}m}$ pairs of such words, determine whether the pair is confusable (we only need to check confusability of a candidate word with words already added to the code). Checking confusability of two words under adversarial deletions reduces to checking whether the longest common subsequence is at least $(1-\delta)m$, which can be done in time $O(m^2)$. This gives an overall runtime of $O(m^2 \cdot F_m \cdot 2^{\mathcal{R}m}) = \tilde{O}(2^{(0.6942+\mathcal{R})m})$. □

Corollary 4.4.3. *There exists a constant m_0^* such that for all $m \geq m_0^*$, there exists a length m binary code of rate $\mathcal{R}_{in} = 0.558$ correcting a $\delta_{in} = 0.0083$ fraction of adversarial insertions and deletions such that each codeword contains runs of size 1 and 2 only and each codeword starts and ends with a 1. Furthermore this code is constructible in time $O(2^m)$.*

Our construction utilizes the following result as a black box for efficiently coding against an arbitrary fraction of insertions and deletions with rate approaching capacity.

Theorem 4.4.4 (Theorem 1.1 of [21])**.** *For any $0 \leq \delta < 1$ and $\epsilon > 0$, there exists a code C over alphabet Σ with $|\Sigma| = \text{poly}(1/\epsilon)$ with block length n, rate $1-\delta-\epsilon$, and is efficiently decodable from δn insertions and deletions. The code can be constructed in time $\text{poly}(n)$, encoded in time $O(n)$, and decoded in time $O(n^2)$.*

We apply Theorem 4.4.4 for small δ, so we also could use the high rate binary code construction of [16].

We now turn to our code construction for Theorem 4.3.1.

The code. Let

$$B = 60, \quad B^* = 1.4\bar{3}B = 86, \quad \eta = \frac{1}{1000}, \quad \delta_{out} = \frac{1}{1000}. \tag{4.2}$$

Let

$$m_0 = \max(\alpha \log(1/\delta_{out})/\eta, m_0^*) \tag{4.3}$$

where α is a sufficiently large constant and m_0^* is given by Corollary 4.4.3. Let $\epsilon_{out} > 0$ be small enough such that the alphabet Σ, given by Theorem 4.4.4 with $\epsilon = \epsilon_{out}$ and δ_{out}, satisfies $|\Sigma| \geq m_0$, and let C_{out} be the corresponding code.

Let $C_{in} : |\Sigma| \to \{0,1\}^m$ be the code given by Corollary 4.4.3, and let $\mathcal{R}_{in}, \delta_{in}$, and $m = \frac{1}{\mathcal{R}_{in}} \log|\Sigma| = O(\log(1/\epsilon))$ be the rate, deletion fraction, and block length of the code, respectively ($\mathcal{R}_{in}$ and δ_{in} are given by Corollary 4.4.3). Each codeword of C_{in} has runs of length 1 and 2 only, and each codeword starts and ends with a 1. This code is constructed greedily.

Our code is a modified concatenated code. We encode our message as follows.

- *Outer Code.* First, encode the message into the outer code, C_{out}, to obtain a word $c^{(out)} = \sigma_1 \dots \sigma_n$.
- *Concatenation with Inner Code.* Encode each outer codeword symbol $\sigma_i \in \Sigma$ by the inner code C_{in}.
- *Buffer.* Insert a buffer of ηm 0s between adjacent inner codewords. Let the resulting word be $c^{(cat)}$. Let $c_i^{(in)} = C_{in}(\sigma_i)$ denote the encoded inner codewords of $c^{(cat)}$.
- *Duplication.* After concatenating the codes and inserting the buffers, replace each character (including characters in the buffers) with $\lceil B/(1-p) \rceil$ copies of itself to obtain a word

of length $N := Bnm/(1-p)$. Let the resulting word be c, and the corresponding inner codewords be $\{c_i^{(dup)}\}$.

Rate. The rate of the outer code is $1 - \delta_{out} - \epsilon_{out}$, the rate of the inner code is $\mathcal{R}_{in}$, the buffer and duplications multiply the rate by $\frac{1}{1+\eta}$ and $(1-p)/B$ respectively. This gives a total rate that is slightly greater than $(1-p)/110$.

Notation. Let s denote the received word after the codeword c is passed through the deletion channel. Note that (i) every bit of c can be identified with a bit in $c^{(cat)}$, and (ii) each bit in the received word s can be identified with a bit in c. Thus, we can define relations $f^{(dup)} : c^{(cat)} \to c$, and $f^{(del)} : c \to s$ (that is, relations on the indices of the strings). These are not functions because some bits may be mapped to multiple (for $f^{(dup)}$) or zero (for $f^{(del)}$) bits. Specifically, $f^{(del)}$ and $f^{(dup)}$ are the inverses of total functions. In this way, composing these relations (i.e. composing their inverse functions) if necessary, we can speak about the *image* and *pre-image* of bits or subwords of one of $c^{(cat)}$, c, and s under these relations. For example, during the Duplication step of encoding, a bit $\langle b_j \rangle$ of $c^{(cat)}$ is replaced with $B/(1-p)$ copies of itself, so the corresponding string $\langle b_j \rangle^{B/(1-p)}$ in c forms the *image* of $\langle b_j \rangle$ under $f^{(dup)}$, and conversely the *pre-image* of the duplicated string $\langle b_j \rangle^{B/(1-p)}$ is that bit $\langle b_j \rangle$.

Decoding algorithm.

- *Decoding Buffer.* First identify all runs of 0s in the received word with length at least $B\eta m/2$. These are our *decoding buffers* that divide the word into *decoding windows*, which we identify with subwords of s.
- *Deduplication.* Divide each decoding window into runs. For each run, if it has strictly more than B^* copies of a bit, replace it with as two copies of that bit, otherwise replace it with one copy. For example, $\langle 0 \rangle^{2B}$ gets replaced with $\langle 00 \rangle$ while $\langle 0 \rangle^{B}$ gets replaced with $\langle 0 \rangle$. For each decoding window, concatenate these runs of length 1 and 2 in their original order in the decoding window to produce a *deduplicated* decoding window.
- *Inner Decoding.* For each deduplicated decoding window, decode an outer symbol $\sigma \in \Sigma_{out}$ from each decoding window by running the brute force deletion correction algorithm for C_{in}. That is, for each deduplicated decoding window $s_*^{(in)}$, find by brute force a codeword $c_*^{(in)}$ in C_{in} that such that $\Delta_{i/d}(c_*^{(in)}, s_*^{(in)}) \le \delta_{in} m$. If $c_*^{(in)}$ is not unique or does not exist, do not decode an outer symbol σ from this decoding window. Concatenate the decoded symbols σ in the order in which their corresponding decoding windows appear in the received word s to obtain a word $s^{(out)}$.
- *Outer Decoding.* Decode the message $\mathfrak{m}$ from $s^{(out)}$ using the decoding algorithm of C_{out} in Theorem 4.4.4.

For purposes of analysis, label as $s_i^{(dup)}$ the decoding window whose pre-image under $f^{(del)}$ contains indices in $c_i^{(dup)}$. If this decoding window is not unique (that is, the image of $c_i^{(dup)}$ contains bits in multiple decoding windows), then assign $s_i^{(dup)}$ arbitrarily. Note this labeling may mean some decoding windows are unlabeled, and also that some decoding windows may have multiple labels. In our analysis, we show both occurrences are rare. For a decoding window $s_i^{(dup)}$, denote the result of $s_i^{(dup)}$ after Deduplication to be $s_i^{(in)}$.

The following diagram depicts the encoding and decoding steps. The pair $(\{c_i^{(in)}\}_i, c^{(cat)})$ indicates that, at that step of encoding, we have produced the word $c^{(cat)}$, and the sequence $\{c_i^{(in)}\}_i$

are the “inner codewords” of $c^{(cat)}$ (that is, the words in between what would be identified by the decoder as decoding buffers). The pair $(\{c_i^{(dup)}\}_i, c)$ is used similarly.

$$\begin{tikzcd}
\mathfrak{m} \arrow[r, "C_{out}"] & c^{(out)} \arrow[r, "C_{in},Buf"] & \left(\left\{c_i^{(in)}\right\}_i, c^{(cat)}\right) \arrow[r, "Dup"] & \left(\left\{c_i^{(dup)}\right\}_i, c\right) \arrow[dll, bend left, red, "\text{BDC}"] & & \\
 & s \arrow[r, "DeBuf"] & \left\{s_i^{(dup)}\right\}_i \arrow[r, "DeDup"] & \left\{s_i^{(in)}\right\}_i \arrow[r, "\mathrm{Dec}_{in}"] & s^{(out)} \arrow[r, "\mathrm{Dec}_{out}"] & \mathfrak{m}
\end{tikzcd}$$

Runtime. The outer code is constructible in $\mathrm{poly}(n)$ time and the inner code is constructible in time $O(2^m) = \mathrm{poly}(1/\epsilon)$, which is a constant, so the total construction time is $\mathrm{poly}(N)$.

Encoding in the outer code is linear time, each of the n inner encodings is constant time, and adding the buffers and applying duplications each can be done in linear time. The overall encoding time is thus $O(N)$.

The Buffer step of the decoding takes linear time. The Deduplication step of each inner codeword takes constant time, so the entire step takes linear time. For each inner codeword, Inner Decoding takes time $O(m^2 2^m) = \mathrm{poly}(1/\epsilon)$ by brute force search over the 2^m possible codewords: checking each of the 2^m codewords is a longest common subsequence computation and thus takes time $O(m^2)$, giving a total decoding time of $O(m^2 2^m)$ for each inner codeword. We need to run this inner decoding $O(n)$ times, so the entire Inner Decoding step takes linear time. The Outer Decoding step takes $O(n^2)$ time by Theorem 4.4.4. Thus the total decoding time is $O(N^2)$.

Correctness. Note that, if an inner codeword is decoded incorrectly, then one of the following holds.

1. (*Spurious Buffer*) A spurious decoding buffer is identified in the corrupted codeword during the Buffer step.
2. (*Deleted Buffer*) A decoding buffer neighboring the codeword is deleted.
3. (*Inner Decoding Failure*) Running the Deduplication and Inner Decoding steps on $s_i^{(dup)}$ computes the inner codeword incorrectly.

We show that, with high probability, the number of occurrences of each of these events is small.

The last case is the most nontrivial, so we deal with it first, assuming the codeword contains no spurious decoding buffers and the neighboring decoding buffers are not deleted. In particular, there is an i such that our decoding window $s_i^{(dup)}$ whose pre-image under $f^{(del)}$ only contains bits in $c_i^{(dup)}$ (because no deleted buffer) and no bits in the image of $c_i^{(dup)}$ appear in any other decoding window (because no spurious buffer).

Recall that the inner code C_{in} can correct against $\delta_{in} = 0.0083$ fraction of adversarial insertions and deletions. Suppose an inner codeword $c_i^{(in)} = r_1 \dots r_k \in C_{in}$ has k runs r_j each of length 1 or 2, so that $m/2 \le k \le m$.

Definition 4.4.5. A subword of α identical bits in the received word s is

- *type-0* if $\alpha = 0$
- *type-1* if $\alpha \in [1, B^*]$
- *type-2* if $\alpha \in [B^* + 1, \infty)$.

By abuse of notation, we say that a length 1 or 2 run r_j of the inner codeword $c_i^{(in)}$ has *type-t_j* if the image of r_j in s under $f^{(del)} \circ f^{(dup)}$ forms a type-t_j subword.

Let $t_1, \dots, t_k$ be the types of the runs $r_1, \dots, r_k$, respectively. The image of a run of length r_j under $f^{(del)} \circ f^{(dup)}$ has length distributed as $\mathrm{Binomial}(B|r_j|/(1-p), 1-p)$. Let $\delta = 0.4\bar{3}$ be such that $B^* = (1+\delta)B$. By the Chernoff bounds in Lemma 2.2.1, the probability that a run r_j of length 1 is type-2 is

$$\mathop{\mathbf{Pr}}_{Z \sim \mathrm{Binomial}(B/(1-p),1-p)}[Z > B^*] < \left(e^{\delta}/(1+\delta)^{1+\delta}\right)^B < 0.0071. \tag{4.4}$$

Similarly, the probability that a run r_j of length-2 is type-1 is at most

$$\mathop{\mathbf{Pr}}_{Z \sim \mathrm{Binomial}(2B/(1-p),1-p)}[Z \le B^*] < e^{-((1-\delta)/2)^2 B} < 0.0081. \tag{4.5}$$

The probability any run is type-0 is at most $\mathbf{Pr}_{Z \sim \mathrm{Binomial}(B/(1-p),1-p)}[Z = 0] < e^{-B} < 10^{-10}$.

We now have established that, for runs r_j in $c_i^{(in)}$, the probability that the number of bits in the image of r_j in s under $f^{(del)} \circ f^{(dup)}$ is "incorrect" (between 1 and B^* for length 1 runs, and greater than B^* for length 2 runs), is at most 0.0081, which is less than δ_{in}. If the only kinds of errors in the Local Decoding step were runs of c of length 1 becoming runs of length 2 and runs of length 2 become runs of length 1, then we have that, by concentration bounds, with probability $1 - 2^{-\Omega(m)}$, the number of insertions deletions needed to transform $s_i^{(in)}$ back into $c_i^{(in)}$ is at most $\delta_{in} m$, in which case $s_i^{(in)}$ gets decoding to the correct outer symbol using C_{in}.

However, we must also account for the fact that some runs r_j of $c_i^{(in)}$ may become deleted completely after duplication and passing through the deletion channel. That is, the image of r_j in s under $f^{(del)} \circ f^{(dup)}$ is empty, or, in other words, r_j is type-0. In this case the two neighboring runs r_{j-1} and r_{j+1} appear merged together in the Deduplication step of decoding. For example, if a run of 1s was deleted completely after duplication and deletion, its neighboring runs of 0s would be interpreted by the decoder as a single run. Fortunately, as we saw, the probability that a run is type-0 is extremely small ($< 10^{-9}$), and we show each type-0 run only increases $\Delta_{i/d}(c_i^{(in)}, s_i^{(in)})$ by a constant. We show this constant is at most 6.

Let Y_j be a random variable that is 0 if $|r_j| = t_j$, 1 if $\{|r_j|, t_j\} = \{1, 2\}$, and 6 if $t_j = 0$. We claim $\sum_{j=1}^{k} Y_j$ is an upper bound on $\Delta_{i/d}(c_i^{(in)}, s_i^{(in)})$. To see this, first note that if $t_j \neq 0$ for all i, then the number of runs of $c_i^{(in)}$ and $s_i^{(in)}$ are equal, so we can transform $c_i^{(in)}$ into $s_i^{(in)}$ by adding a bit to each length-1 type-2 run of $c_i^{(in)}$ and deleting a bit from each length-2 type-1 run of $s_i^{(in)}$.

Now, if some number, ℓ, of the t_j are 0, then at most 2ℓ of the runs in $c_i^{(in)}$ become merged with some other run (or a neighboring decoding buffer) after duplication and deletion. Each set of consecutive runs $r_j, r_{j+2}, \dots, r_{j+2j'}$ that are merged after duplication and deletion gets replaced with 1 or 2 copies of the corresponding bit. For example, if $r_1 = \langle 11 \rangle, r_2 = \langle 0 \rangle, r_3 = \langle 11 \rangle$, and if after duplication and deletion, $2B$ bits remain in each, and r_2 is type-0, then the image of $r_1 r_2 r_3$ under $f^{(del)} \circ f^{(dup)}$ is $\langle 1 \rangle^{4B}$, which gets decoded as $\langle 11 \rangle$ in the Deduplication step because $\langle 1 \rangle^{4B}$ is type-2. To account for the type-0 runs in transforming $c_i^{(in)}$ into $s_i^{(in)}$, we (i) delete at most two bits from each of the ℓ type-0 runs in $c_i^{(in)}$ and (ii) delete at most two bits for each of at most 2ℓ merged runs in $c_i^{(in)}$. The total number of additional insertions and deletions required to account for type-0 runs of c is thus at most 6ℓ, so we need at most 6 insertions and deletions to account for each type-0 run.

Our analysis covers the case when some bits in the image of $c_i^{(in)}$ under $f^{(del)} \circ f^{(dup)}$ are interpreted as part of a decoding buffer. Recall that inner codewords start and end with a 1, so that $r_1 \in \{\langle 1 \rangle, \langle 11 \rangle\}$ for every inner codeword. If, for example, $t_1 = 0$, that is, the image under $f^{(del)} \circ f^{(dup)}$ of the first run of 1s, r_1, is the empty string, then the bits of r_2 are interpreted as part of the decoding buffer. In this case too, our analysis tells us that the type-0 run r_1 increases $\Delta_{i/d}(c_i^{(in)}, s_i^{(in)})$ by at most 6.

We conclude $\sum_{j=1}^{k} Y_j$ is an upper bound for $\Delta_{i/d}(c_i^{(in)}, s_i^{(in)})$.

Note that if r_j has length 1, then by (4.4) we have

$$\mathbf{E}[Y_j] \;=\; 1 \cdot \mathbf{Pr}[r_j \text{ is type-2}] + 6 \cdot \mathbf{Pr}[r_j \text{ is type-0}] \;<\; 1 \cdot 0.0071 + 6 \cdot 10^{-9} \;<\; 0.0082. \tag{4.6}$$

Similarly, if r_j has length 2, then by (4.5) we have

$$\mathbf{E}[Y_j] \;=\; 1 \cdot \mathbf{Pr}[r_j \text{ is type-1}] + 6 \cdot \mathbf{Pr}[r_j \text{ is type-0}] \;<\; 1 \cdot 0.0081 + 6 \cdot 10^{-9} \;<\; 0.0082. \tag{4.7}$$

Thus $\mathbf{E}[Y_j] < 0.0082$ for all i. We know the word $s_i^{(in)}$ is decoded incorrectly (i.e. is not decoded as σ_i) in the Inner Decoding step only if $\Delta_{i/d}(c_i^{(in)}, s_i^{(in)}) > \delta_{in} m$. The Y_j are independent, so Lemma 2.2.2 gives

$$\begin{aligned}
\mathbf{Pr}[s_i^{(in)} \text{ decoded incorrectly}] \;&\le\; \mathbf{Pr}[Y_1 + Y_2 + \cdots + Y_k \ge \delta_{in} m] \\
&\le\; \mathbf{Pr}[Y_1 + Y_2 + \cdots + Y_k \ge \delta_{in} k] \\
&\le\; \exp\left(-\frac{(\delta_{in} - 0.0082)^2 k}{3 \cdot 6 \cdot \delta_{in}}\right) \\
&\le\; \exp\left(-\Omega(m)\right)
\end{aligned} \tag{4.8}$$

where the last inequality is given by $k \ge m/2$. Since our $m \ge \Omega(\log(1/\delta_{out}))$ is sufficiently large, we have the probability $s_i^{(in)}$ is decoded incorrectly is at most $\delta_{out}/10$. If we let $Y_j^{(i)}$ denote the Y_j corresponding to inner codeword $c_i^{(in)}$, the events E_i given by $\sum_j Y_j^{(i)} \ge \delta_{in} m$ are independent. By concentration bounds on the events E_i, we conclude the probability that there are at least $\delta_{out} n/9$ incorrectly decoded inner codewords is $2^{-\Omega(n)}$.

Our aim is to show that the number of spurious buffers, deleted buffers, and inner decoding failures is small with high probability. So far, we have shown that, with high probability, assuming a codeword is not already affected by spurious buffers and neighboring deleted buffers, the number of inner decoding failures is small. We now turn to showing the number of spurious buffers is likely to be small.

A spurious buffer appears inside an inner codeword if many consecutive runs of 1s are type-0. A spurious buffer requires at least one of the following: (i) a codeword contains a sequence of at least $\eta m/5$ consecutive type-0 runs of 1s, (ii) a codeword contains a sequence of $\ell \le \eta m/5$ consecutive type-0 runs of 1s, such that, for the $\ell + 1$ consecutive runs of 0s neighboring these type-0 runs of 1s, their image under $f^{(del)} \circ f^{(dup)}$ has at least $0.5\eta m$ 0s. We show both happen with low probability within a codeword.

A set of ℓ consecutive type-0 runs of 1s occurs with probability at most $10^{-10\ell}$. Thus the probability an inner codeword has a sequence of $\eta m/5$ consecutive type-0 runs of 1s is at most $m^2 \cdot 10^{-10\eta m/5} = \exp(-\Omega(\eta m))$. Now assume that in an inner codeword, each set of consecutive type-0 runs of 1s has size at most $\eta m/5$. Each set of ℓ consecutive type-0 runs of 1s merges $\ell + 1$

consecutive runs of 0s in c, so that they appear as a single longer run in s. The sum of the length of these $\ell+1$ runs is some number ℓ^* that is at most $2\ell+2$. The number of bits in the image of these runs of $c_i^{(in)}$ under $f^{(del)} \circ f^{(dup)}$ is distributed as $\text{Binomial}(\ell^* B/(1-p), 1-p)$. This has expectation $\ell^* B \leq 0.41 B\eta m$, so by concentration bounds, the probability this run of s has length at least $0.5 B\eta m$, i.e. is interpreted as a decoding buffer, is at most $\exp(-\Omega(\eta m))$. Hence, conditioned on each set of consecutive type-0 runs of 1s having size at most $\eta m/5$, the probability of having no spurious buffers in a codeword is at least $1-\exp(-\Omega(\eta m))$. Thus the overall probability there are no spurious buffers a given inner codeword is at least $(1-\exp(-\Omega(\eta m))(1-\exp(-\Omega(\eta m))) = 1-\exp(-\Omega(\eta m))$. Since each inner codeword contains at most m candidate spurious buffers (one for each type-0 run of 1s), the expected number of spurious buffers in an inner codeword is thus at most $m \cdot \exp(-\Omega(\eta m))$. By our choice of $m \geq \Omega(\log(1/\delta_{out})/\eta)$, this is at most $\delta_{out}/10$. The occurrence of conditions (i) and (ii) above are independent between buffers. The total number of spurious buffers thus is bounded by the sum of n independent random variables each with expectation at most $\delta_{out}/10$. By concentration bounds, the probability that there are at least $\delta_{out} n/9$ spurious buffers is $2^{-\Omega(n)}$.

A deleted buffer occurs only when the image of the ηm 0s in a buffer under $f^{(del)} \circ f^{(dup)}$ is at most $B\eta m/2$. The number of such bits is distributed as $\text{Binomial}(B\eta m/(1-p), 1-p)$. Thus, each buffer is deleted with probability $\exp(-B\eta m) < \delta_{out}/10$ by our choice of $m \geq \Omega(\log(1/\delta_{out})/\eta)$. The events of a buffer receiving too many deletions are independent across buffers. By concentration bounds, the probability that there are at least $\delta_{out} n/9$ deleted buffers is thus $2^{-\Omega(n)}$.

Each inner decoding failure, spurious buffer, and deleted buffer increases $\Delta_{i/d}(c_i^{(out)}, s_i^{(out)})$ by at most 3: inner decoding failure causes up to 1 insertion and 1 deletion; spurious buffer causes up to 1 deletion and 2 insertions; and deleted buffer causes up to 2 deletions and 1 insertion. Our message is decoded incorrect if $\Delta_{i/d}(c_i^{(out)}, s_i^{(out)}) > \delta_{out} n$. Thus, there is a decoding error in the outer code only if at least one of (i) the number of incorrectly decoded inner codewords, (ii) the number of spurious buffers, or (iii) the number of deleted buffers is at least $\delta_{out} n/9$. However, by the above arguments, each is greater than $\delta_{out} n/9$ with probability $2^{-\Omega(n)}$, so there is a decoding error with probability $2^{-\Omega(n)}$. This concludes the proof of Theorem 4.3.1.

4.5 Possible alternative constructions

As mentioned in the introduction, Drinea and Mitzenmacher [12, 13] proved that the capacity of the BDC$_p$ is at least $(1-p)/9$. However, their proof is nonconstructive and they do not provide an efficient decoding algorithm.

One might think it is possible to use Drinea and Mitzenmacher's construction as a black box. We could follow the approach in this thesis, concatenating an outer code given by [21] with the rate $(1-p)/9$ random-deletion-correcting code as a black box inner code. The complexity of the Drinea and Mitzenmacher's so-called *jigsaw decoding* is not apparent from [13]. However, the inner code has constant length, so construction, encoding, and decoding would be constant time. Thus, the efficiency of the inner code would not affect the asymptotic runtime.

The main issue with this approach is that, while the inner code can tolerate random deletions with probability p, inner codeword bits are *not* deleted in the concatenated construction according to a BDC$_p$; the 0 bits closer to the buffers between the inner codewords are deleted with higher probability because they might be "merged" with a buffer. For example, if an inner codeword is $\langle 101111 \rangle$, then because the codeword is surrounded by buffers of 0s, deleting the leftmost 1 effectively deletes two bits because the 0 is interpreted as part of the buffer. While this may not be a significant issue because the distributions of deletions in this deletion process and BDC$_p$ are quite similar, much more care would be needed to prove correctness.

Our construction does not run into this issue, because our transmitted codewords tend to have *many* 1s on the ends of the inner codewords. In particular, each inner codeword of C_{in} has 1s on the ends, so after the Duplication step each inner codeword has $B/(1-p)$ or $2B/(1-p)$ 1s on the ends. The 1s on the boundary of the inner codeword will all be deleted with probability $\approx \exp(-B)$, which is small. Thus, in our construction, it is far more unlikely that bits are merged with the neighboring decoding buffer, than if we were to use a general inner code construction. Furthermore, we believe our construction based on bit duplication of a worst-case deletion correcting code is conceptually simpler than appealing to an existential code.

As a remark, we presented a construction with rate $(1-p)/110$, but using a randomized encoding we can improve the constant from 1/110 to 1/60. We can modify our construction so that, during the Duplication step of decoding, instead of replacing each bit of $c^{(cat)}$ with a fix number $B/(1-p)$ copies of itself, we instead replaced each bit independently with $\text{Poisson}(B/(1-p))$ copies of itself. Then the image of a run r_j under duplication and deletion is distributed as $\text{Poisson}(B)$, which is independent of p. Because we don't have a dependence on p, we can tighten our bounding in (4.4) and (4.5). To obtain $(1-p)/60$, we can take $B = 28.12$ and set $B^* = 40$, where B^* is the threshold after which runs are decoded as two bits instead of one bit in the Deduplication step. The disadvantage of this approach is that we require our encoding to be randomized, whereas the construction presented above uses deterministic encoding.

Chapter 5

Oblivious deletions

5.1 Introduction

We now turn to a natural model that bridges between the adversarial and random deletion models, namely *oblivious deletions*. Here we assume that an arbitrary subset of pn locations of the codeword can be deleted, but these positions must be picked without knowledge of the codeword. The oblivious model is well-motivated in settings where the noise may be mercurial and caused by hard to model physical phenomena, but not by an adversary.

If the code is deterministic, tackling oblivious deletions is equivalent to recovering from worst-case deletions. We allow the encoder to be randomized, and require that for every message m and every deletion pattern τ, most encodings of m can be decoded from the deletion pattern τ. The randomness used at the encoding is private to the encoder and is not needed at the decoder, which we require to be deterministic. Note that the oblivious model generalizes random deletions, as deleting each bit independently with probability p is oblivious to the actual codeword, and with high probability one has $\approx pn$ deletions. Of course, any code which is decodable against pn adversarial deletions is decodable also against pn oblivious deletions, even without any randomization in the encoding. To our knowledge, before [17], there were no known results on coding against oblivious deletions.

In this chapter, we discuss a new result, Theorem 5.2.1, in [17], on oblivious deletions, in §5.2. We then review the relevant literature in §5.3. We provide a outline of our construction and proof in §5.4, and we give a full proof in §5.5.

5.2 New results on codes against oblivious deletions

Perhaps surprisingly, we prove that in the oblivious model, the limit of $p \leq 1/2$ does not apply, and in fact one can correct a deletion fraction p approaching 1. This generalizes the result that one can correct a fraction $p \to 1$ of random deletions.

Theorem 5.2.1 (Main). *For every $p < 1$, there exists $\mathcal{R} > 0$ and a code family with a randomized encoder* $\mathsf{Enc} : \{0,1\}^{\mathcal{R}n} \to \{0,1\}^n$ *and (deterministic) decoder* $\mathsf{Dec} : \{0,1\}^{(1-p)n} \to \{0,1\}^{\mathcal{R}n} \cup \{\perp\}$ *such that for all deletion patterns τ with pn deletions and all messages $m \in \{0,1\}^{\mathcal{R}n}$,*

$$\mathbf{Pr}[\mathsf{Dec}(\tau(\mathsf{Enc}(m))) \neq m] \leq o(1) ,$$

where the probability is over the randomness of the encoder (which is private to the encoder and not known to the decoder).

The above result is implied by deterministic codes C decodable from arbitrary pn deletions under average-error criterion; i.e., there is a decoding function Dec : $\{0,1\}^{(1-p)n} \to C$ such that for every deletion pattern τ with pn deletions, for most codewords $c \in C$, $\mathsf{Dec}(\tau(c)) = c$. We stress that the above is an *existential* result, and the codes guaranteed by the above theorem are not explicitly specified. The decoding algorithm amounts to looking for a codeword which contains the received bit string as a subsequence, and outputting it if there is a unique such codeword.

5.3 Related work

The model of oblivious errors (such as bit flips) has has been studied in the information-theory literature as a particular case of arbitrarily varying channels with state constraints [31] (see the related work section of [19] for more background on this connection). In particular, for the case of bit flips, the capacity against the model of pn oblivious bit flips (for $p \le 1/2$) equals $1 - h(p)$, matching the Shannon capacity of the binary symmetric channel that flips each bit independently with probability p. (This special case was re-proved in [30] by a different simpler random coding argument compared to the original works [6, 7].) Similarly, the capacity against the model of pn oblivious erasures is $1 - p$, matching the Shannon capacity of the binary erasure channel. Explicit codes of rate approaching $1 - h(p)$ to correct pn oblivious bit flips (in the sense of Theorem 5.2.1, with randomized encoding) were given in [19]. This work also considered computationally bounded noise models, such as channels with bounded memory or with small circuits, and gave optimal rate codes for *list* decoding against those models. These models are more general than oblivious errors, but still not as pessimistic as adversarial noise.

Notice that in the case of both erasures and errors, the capacity in the oblivious and random models were the same. It is not clear if this is the case for deletions. The rate of the codes we guarantee in Theorem 5.2.1 for $p \to 1$ are much worse than the $\Omega(1-p)$ lower bound known for random deletions.

5.4 Outline of construction and proof

Our first, naive attempt at this problem is to choose a random subset of $2^{\mathcal{R}N}$ codewords in $\{0,1\}^N$. This technique, however, does not work in the same way it does for oblivious bit-flips. See Appendix C for a discussion on the difficulties of this approach.

Instead of proving Theorem 5.2.1 directly, we prove a related theorem for decoding in the average case. First, a definition.

Definition 5.4.1. We say a (non-stochastic) binary code C with rate $\mathcal{R}$ and length N *decodes* pN *deletions in the average case* if for any deletion pattern τ deleting pN bits, we have

$$|\{x \in C : \exists y \in C \text{ s. t. } x \neq y \text{ and } \tau(x) \le y\}| \;\le\; o_N(|C|). \tag{5.1}$$

Theorem 5.4.2 (Average Case Deletions)**.** *Let $p \in (0,1)$. There exists a constant $\mathcal{R}$ such that there exists infinitely many N for which there is a rate $\mathcal{R}$ code $C \subseteq \{0,1\}^N$ that decodes pN deletions in average case.*

It is standard to show that oblivious and average-case decoding are equivalent. In particular Theorem 5.4.2 implies Theorem 5.2.1. For completeness we provide a proof of this implication in Appendix B. In fact the capacity for decoding in the average case is the same as the capacity for the oblivious channel. Roughly, if we have a length N code C with rate $\mathcal{R}$ that decodes against pN deletions in the average case, then we can group the codewords into sets of size $2^{0.01\mathcal{R}n}$. Then we associate every message with a set of codewords and encode the message by randomly choosing a codeword from its set. Our decoding function simply takes a received word s and looks for a codeword c such that $s \sqsubseteq c$, i.e. c is a superstring of s. If there is exactly one such c, output the associated codeword, otherwise output $\perp$. Using the fact that C decodes pN deletions in the average case, we can show that for all deletion patterns τ, only a few codewords do not decode correctly via unique decoding in our new stochastic code.

We now outline our construction for Theorem 5.4.2. Our construction for the average deletion code uses the "clean construction" construction from [4] with appropriately selected parameters. The idea is to choose a concatenated code such that the inner code widely varies in the number of runs between codewords. Specifically, we choose sufficiently large constants R and K and set our inner code to have length $L = 2R^K$. For $i = 1, 2, \ldots, K$, set our inner codewords to be

$$g_i = \left(0^{R^{i-1}}1^{R^{i-1}}\right)^{L/(2R^{i-1})} \tag{5.2}$$

where 0^k and 1^k denote strings of k 0s and 1s, respectively. In this way, the number of runs between any two codewords differs by a factor of at least R. Our outer code is a subset of $[K]^n$, so that the total code length is $N = nL$, and we concatenate the code via a function $\psi : [K]^* \to \{0,1\}^*$ that replaces a symbol $i \in [K]$ with the string $g_i \in \{0,1\}^L$. The outer code is chosen via a random process, detailed in the following paragraphs; the process throws out a small subset of "bad" elements of $[K]^n$ and chooses a constant rate code by including each remaining element independently with some small fixed probability. Our decoding function is *unique decoding*. That is, given a received word s, we find a codeword c such that $s \sqsubseteq c$. If such a codeword is unique, our decoder returns that output. Otherwise, the decoder returns $\perp$.

The following example illustrates why the varying run length is powerful even for correct more than $0.5N$ deletions: Suppose $n = 100$, $p = 0.9$, $R \gg 20$, our received word is $s = g_1^{10}$ (that is, 10 copies of g_1 concatenated together) and our code contains the codeword $c = g_2^{100}$. Then s is a subsequence of c if and only if we can identify each of the 10 g_1s with non-overlapping bits of c. However, since g_1 contains over 20 times as many runs as g_2, each g_1 must be identified with a subsequence of c spanning at least 20 inner codewords, i.e. copies of g_2. This means the subsequence s roughly must span at least 200 inner codewords, but c only has 100 inner codewords, contradiction. While this imbalanced run-count behavior is a key to our argument, it is worth noting that the behavior is asymmetric. In particular, while it takes R copies of g_2 to produce g_1 as a subsequence, we only need two copies of g_1 to produce g_2 as a subsequence.

To analyze this code, we leverage the run-count behavior of the inner codewords. This contrasts with the adversarial setting, where the run-count property of the same code is featured less centrally in the proof of correctness [4]. We show that for any deletion pattern τ with up to pN deletions, two random codewords X and Y are "confusable" with exponentially small probability in N. To be precise, we have for all τ,

$$\underset{X,Y\sim U([K]^n)}{\mathbf{Pr}}[\tau(\psi(X)) \sqsubseteq \psi(Y)] \;<\; 2^{-\Omega(n)}. \tag{5.3}$$

The idea for this proof can be illustrated in the case that τ deletes only entire inner codewords. If τ deletes pn of the n inner codewords and does not touch the remaining codewords, then $\tau(\psi(X))$ has the same distribution over length $(1-p)N$ binary strings as $\psi(X')$ where $X' \sim U([K]^{(1-p)n})$. Thus we would like to show

$$\mathop{\mathbf{Pr}}_{\substack{X' \sim U([K]^{(1-p)n}) \\ Y \sim U([K]^n)}} [\psi(X') \sqsubseteq \psi(Y)] \; < \; 2^{-\Omega(n)}. \tag{5.4}$$

Consider trying to find $\psi(X')$ as a substring of $\psi(Y)$ where $X' = X'_1 X'_2 \dots X'_{(1-p)n}$ This is possible if and only if we match the bits of X' to bits of Y greedily. However, each inner codeword $g_{X'_i}$ spans a large number (R) of inner codewords of Y unless the greedy matching encounters at least one Y_j such that $Y_j \le X'_i$, i.e. a higher frequency inner codeword (if $Y_j < X'_i$, we may need to encounter two such higher frequency inner codewords, but two is at least one). Thus, if $X'_i = 1$ for some i, then the number of inner codewords spanned by $g_{X'_i}$ is approximately distributed as $\text{Geometric}(1/K)$. In general, conditioned on $X'_i = k$, the number of inner codewords spanned by $g_{X'_i}$ is approximately distributed as $\text{Geometric}(k/K)$. Thus, the number of inner codewords of Y spanned by a single inner codeword X'_i is

$$\frac{1}{K} \cdot \Theta\left(\frac{K}{1}\right) + \frac{1}{K} \cdot \Theta\left(\frac{K}{2}\right) + \cdots + \frac{1}{K} \cdot \Theta\left(\frac{K}{K}\right) \; = \; \Theta(\log K) \tag{5.5}$$

If we choose K so that $\Theta(\log K) > \frac{2}{1-p}$ then the expected number of inner codewords of Y spanned by X' is more than $\Theta(\log K) \cdot (1-p)n > 2n$, so concentration bounds tell us that the probability that $\psi(X') \sqsubseteq \psi(Y)$ is exponentially small in n.

Note that there is a slight caveat to the above argument because the numbers of inner codewords in $\psi(Y)$ spanned by $\psi(X'_i)$ are not independent across all i. For example, if $\psi(Y)$ begins with $g_2 g_1$ and $\psi(X')$ begins with $g_1 g_1$, then the second inner codeword of $\psi(Y)$ has a few "leftover bits" that easily match with $\psi(X')$'s second inner codeword. However, we can adjust for this independence by relaxing our analysis by a constant factor.

The above addresses the case when the deletion pattern, for each inner codeword, either deletes the codeword entirely or does not modify it at all. We now show how to extend this to general deletion patterns, starting with the case of $p < \frac{1}{2}$.

Note that the above argument only depended on the inner codewords having widely varying runs. By a simple counting argument, we can verify that at least a $(0.5-p)$ fraction of codewords have at most $(p + (0.5-p)/2)L = (0.5+p)L/2$ deletions. Since we are in the regime where $p < \frac{1}{2}$, applying $(0.5+p)L/2$ deletions to an inner codeword with r runs cannot make the number of runs less than $(0.5-p)r$, as it can delete at most $(0.5+p)/2$ fraction of the runs, and deleting each run reduces the number of runs by at most two. If we choose $R \gg 1/(0.5-p)^2$, then we can guarantee that even if we have a generic deletion pattern, we have a constant fraction $(0.5-p)$ of positions for which the run-count properties of all inner codewords in those positions are preserved up to a factor of $\sqrt{R}$. Thus, as the number of runs between any two inner codewords differs by a factor of at least R, even after these corruptions the ratio between the number of runs of two "preserved" inner codewords is still at least $\sqrt{R}$. Using the same argument as above and now requiring $\Theta(\log K) > \frac{2}{0.5-p}$, we can conclude that even for general deletion patterns that the probability that two random candidate codewords are confused is exponentially small.

As a technical note, working with deletion patterns directly is messy as they encode a large amount of information, much of which we do not need in our analysis. Furthermore, the caveat mentioned in the clean deletion pattern case regarding the independence of number inner codewords spanned by some $\psi(X_i')$ becomes more severe for general deletion patterns. This happens because in general deletion patterns, especially later for $p > \frac{1}{2}$, the inner codewords of $\psi(X)$ that have preserved their run-count property might nonetheless be shorter, so many (in particular, $\Theta(1/(1-p))$) inner codewords could match to single inner codewords of $\psi(Y)$. To alleviate this complexity in the analysis, we introduce a technical notion called a *matching* intended to approximate the subsequence relation. This notion allows us to capture only the run-count behavior of the deletion patterns with respect to the inner codewords while also accounting for the lack-of-independence caveat. For a deletion pattern τ, let σ be the associated deletion pattern such that $\sigma(X)$ to removes all outer codeword symbols except the ones in whose position τ preserves the run-count property of all the inner codewords (these exist because we are still in the case $p < 1/2$). In our proof, we argue that if $\tau(\psi(X))$ is a subsequence of $\psi(Y)$, then $\sigma(X)$ has a matching in Y, and that the probability that $\sigma(X)$ has a matching in Y for two codewords X and Y is exponentially small.

To extend the argument for generic deletion patterns from $p < 1/2$ to $p < 1$, we must use a "local list decoding" idea. Note that when the number of deletion patterns exceeds $1/2$, for every codeword there exists deletion patterns that destroy all or almost all of the information in the codeword, e.g. the deletion pattern that deletes all the 1s (or 0s) of the codeword. For this reason, codes cannot correct against more than $1/2$ fraction of adversarial deletions. However, one can show that this does not happen too frequently allowing us to correct oblivious and average case deletions. In contrast to the $p < 1/2$ case where we found a small, constant fraction of inner codeword positions in which the deletion pattern of the inner codeword preserved the run-count property for *all* inner codewords, we can now find a small, constant fraction of inner codeword positions in which the deletion pattern of the inner codeword preserves the run-count property for *all but a few* inner codewords. For example, even if an inner code deletion pattern deletes every other bit and thus deletes all the information of g_1, the number of runs of $g_2, g_3, \ldots, g_K$ are still preserved. We call this idea "local list decoding" because while we cannot decode our constant fraction of inner codewords uniquely, we can still pin down the inner codewords to a few possibilities. By extending our definition of matching to account for a few inner codewords potentially losing their run-count behavior, we can prove, just as for $p < 1/2$, that $\tau(\psi(X)) \sqsubseteq \psi(Y)$ implies $\sigma(X)$ has a matching in Y, and $\sigma(X)$ having a matching in Y happens with small probability.

At this point of the proof, we have combinatorially established everything we need to prove that our code is decodable in the average case (and thus against oblivious deletions). That is, we have shown that a random candidate codeword in $\psi([K]^n)$ has an exponentially small probability of being confused with another random candidate codeword. Given that codewords have an exponentially small probability of being confusable with each other, it is natural to consider choosing a code by randomly selecting a subset of $[K]^n$. Using this construction, we might try using concentration bounds to show that, for any deletion pattern τ, the probability that we have more than $\epsilon|C|$ codewords (for $\epsilon = o(N)$) that are confusable with some other codeword is at most $2^{-\omega(N)}$, and we can union bound over the at-most-2^N choices of τ. This however does not work directly as the decodability for a given deletion pattern τ depends on the decodability of other deletion patterns. For example, if $p > \frac{1}{2}$ and we happen to choose $c = \psi(11\ldots1) = 0101\ldots01$ as a codeword, then for any deletion pattern τ with pN deletions, $c' \sqsubseteq c$ for *all* candidate codewords c'. From this example alone, the probability of many codewords confusable with c is at least K^{-n} and there are many more examples of such *easily disguised* candidate codewords. Fortunately, we can prove that the number of easily disguised candidate codewords is small. In particular, we show that the majority of elements of $\psi([K]^n)$ are not easily disguised in *all* deletion patterns τ. This intuitively makes sense because, as we have shown, in any deletion pattern τ, on average, words are disguised as an exponentially small fraction of codewords, and because the easily disguised words tend to be easily disguised in every deletion pattern τ. For example, $\psi(11\dots 1_K) = 0101\dots 01$ is easily disguised for any deletion pattern τ.

After throwing out the easily disguised candidate codewords, we randomly choose a constant rate code from the remaining candidate codewords. Careful bookkeeping confirms that with positive probability we obtain a code that decodes pN-deletions in the average case. The bookkeeping is nontrivial, because just as there are a handful of words like $\psi(11\dots 1)$ that are easily disguised as other codewords with deletions, there are also a handful of *easily confused* words like $\psi(KK\dots K)$ that can be confused with many other words when deletions are applied to it. Furthermore, unlike easily disguised codewords, these easily confused words vary over the different deletion patterns, so we cannot simply throw them out. However, like for easily disguised codewords, we show the number of easily confused words is an exponentially small fraction of the codebook size in expectation, so such words do not contribute significantly to the number of incorrectly decoded codewords. Note the subtle difference between easily disguised and easily confused words: a single easily disguised word like $\psi(11\dots 1)$ causes *many* candidate codewords to fail to decode under our unique decoding, but any easily confused codeword adds *at most one* failed decoding.

We model managing easily disguised and easily confusable codewords via a directed graph, where, roughly, for each deletion pattern, we consider a graph on $[K]^n$ where $\overrightarrow{YX}$ is an edge if and only if $\tau(\psi(X)) \sqsubseteq \psi(Y)$. In our proof, we replace the subsequence relation with the matching relation (see §5.8). In this graph language, the easily disguised codewords correspond to vertices with high outdegree, and the easily confusable codewords correspond to vertices with high indegree.

Our construction illustrates the subtle nature of the oblivious deletion channel and average case errors. These settings share much of the behavior of the adversarial deletion channel such as the fact that for $p > \frac{1}{2}$, every codeword has a deletion pattern destroying all of its information. Consequently, our approach tackles the oblivious and average case errors using a combinatorial argument just as the best adversarial deletion results do [4]. Yet, the relaxed decoding requirement allows us to exploit it to correct a fraction of deletions approaching 1.

5.5 Overview of proof

In §5.4 we gave a high level overview. We now begin with a brief snapshot of the proof structure and how it is organized.

We present our general code construction in §5.6. The construction uses the "clean construction" in [4] and an outer code that we choose randomly. We analyze properties of the concatenated construction in §5.7. We begin by extracting the useful behavior of deletion patterns with respect to our codewords. This deletion pattern analysis culminates in Lemma 5.7.7, allowing us to define the *signature* (Definition 5.7.8) of a deletion pattern. The key result of §5.7 is Proposition 5.7.18. It states that for any deletion pattern τ, the probability that two random candidate codewords c, c' are confusable (in the sense that $\tau(c) \sqsubseteq c'$) is exponentially small in the code length. However, because working with deletion patterns directly is messy, the proposition is written in the language of *matchings*, a technical notion defined in Definition 5.7.11. In short, because the inner codewords are nicely behaved, we do not need to know the exact details of the behavior of a given deletion pattern τ, but rather only need certain properties of it, given by its signature. We thus define a matching to approximate the subsequence relation using only the signature of τ, so that "$\sigma(X)$ is matchable in Y" (where σ is the outer code deletion pattern given by τ's signature) holds roughly when "$\tau(\psi(X)) \sqsubseteq \psi(Y)$" holds.

Combinatorially, Proposition 5.7.18 allows us to finish the proof. As stated in §5.4, for our outer code we consider $[K]^n$ minus a small set of easily disguised candidate codewords. The notion of a easily disguised candidate codeword is well defined by Lemma 5.7.23. For a sufficiently small constant γ, we randomly choose a size $2^{\gamma n}$ outer code over the remaining outer codewords. In §5.8, we use the graph language described in §5.4 and prove Lemma 5.8.3, which roughly states that in a sparse directed graph, if we randomly sample a small subset of vertices, the induced subgraph is also sparse (for some appropriate definition of sparse) with high probability. Finally, in §5.9, we piece together these results, showing that Lemma 5.8.3 guarantees, with positive probability, that our random code combinatorially decodes against pN deletions in the average case.

5.6 Construction

Let $p \in (0,1)$, and let $\lambda = \lambda(p)$ be the smallest integer such that $(1+p)/2 < 1 - 2^{-\lambda}$. For our argument any λ such that $p < 1 - 2^{-\lambda}$ suffices. In particular, for $p < \frac{1}{2}$, we can choose $\lambda = 1$, slightly simplifying the argument as described in §5.4. However, we choose λ to be the smallest λ such that $(1+p)/2 < 1 - 2^{-\lambda}$ to ensure a clean formula for the rate.

Let δ be such that $p = 1 - 2^{-\lambda} - \delta$. Let n be a positive integer. With hindsight, choose

$$K = 2^{\lceil 2^{\lambda+5}/\delta \rceil}, \qquad R = 4K^4, \qquad L = 2R^K, \qquad N = nL. \tag{5.6}$$

Note that R is even. For the remainder of this section, the variables p, λ, δ, K, R and L are fixed.

In this way we have $1 - 2^{-\lambda} - \frac{1}{\sqrt{R}} - \frac{\delta}{2} > p$. For $i = 1, \ldots, K$, let g_i be the length L word

$$g_i = \left(0^{R^{i-1}} 1^{R^{i-1}}\right)^{L/(2R^{i-1})}. \tag{5.7}$$

Consider the encoding $\psi : [K]^* \to \{0,1\}^*$ where $\psi(X_1 \cdots X_k) = g_{X_1} g_{X_2} \cdots g_{X_k}$. We construct a concatenated code where the outer code is a length n code over $[K]$ and the inner code is $\{g_1, g_2, \ldots, g_K\} \subseteq \{0,1\}^L$. For the outer code, we choose a random code C_{out} where each codeword is chosen uniformly at random from $[K]^n$ minus a small undesirable subset that we specify later. We choose our decoding function to be *unique decoding*. That is, our decoder iterates over all codewords $c \in C$ and checks if the received word s is a subsequence of c. If it is a subsequence of exactly one c, the decoder returns that c, otherwise it fails. While this decoder is not optimal in terms of the fraction of correctly decoded codewords (it could try to break ties instead of just giving up), it is enough for this proof. Furthermore, since we are showing combinatorial decodability, we do not need the decoder to be efficient.

If, for some $p' > p$, a code can decode $p'nL$ average case or oblivious deletions, then it can decode pnL deletions. Thus, we can decrease δ until δn is an even integer, so may assume without loss of generality that δn is an even integer.

5.7 Analyzing construction behavior

Definition 5.7.1. A g_i*-segment* is an interval in $[L]$ corresponding to a run of g_i. Note that the g_i-segments partition $[L]$ and are of the form $[1 + aR^{i-1}, (a + 1)R^{i-1}]$ for $a \in \{0, \ldots, L/R^{i-1} - 1\}$.

Note that g_i has $2R^{K+1-i}$ runs. In particular, the number of runs greatly varies between inner codewords. This property makes the concatenated construction powerful because it is difficult to find common subsequences of different inner codewords. The following definition allows us to reason about the inner codewords in terms of their run counts.

Definition 5.7.2. We say an inner code deletion pattern σ *preserves* g_i if $\sigma(g_i)$ has at least

$$2R^{K+1-i}/\sqrt{R} = 2R^{K+\frac{1}{2}-i} \tag{5.8}$$

runs. Otherwise we say σ *corrupts* g_i.

We start with a basic but useful fact about deletion patterns and runs.

Lemma 5.7.3. *Suppose w is a word with r runs $I_1, \ldots, I_r$ and τ is a deletion pattern such that $\tau(w)$ has r' runs. We can think of these runs I_k as subsets of consecutive indices in $\{1, \ldots, |w|\}$. Then the number of runs I_k of w completely deleted by τ, i.e. satisfying $I \subseteq \tau$ when τ is thought of as a subset of $\{1, \ldots, |w|\}$, is at least $\frac{r-r'}{2}$.*

Proof. Deleting any run reduces the number of runs in a word by at most 2, and τ reduces the number of runs by $r - r'$, so the claim follows. □

The next lemma establishes the usefulness of widely varying runs in our construction. It says that even when an inner code deletion pattern has a large number of deletions, most of the inner codewords still look the same in terms of the number of runs. The intuition for the lemma is as follows. Consider the extreme example of an inner code deletion pattern σ that “completely corrupts” the inner codewords $g_1, \ldots, g_\lambda$. That is, σ deletes all the zeros of each of $g_1, \ldots, g_\lambda$. Since σ deletes all the zeros of g_λ, it must delete every other run of $R^{\lambda-1}$ bits, thus deleting $L/2$ bits. Applying these deletions alone to $g_{\lambda-1}$ leaves it with half as many runs of length exactly $R^{\lambda-2}$. However, since σ also deletes all zeros of $g_{\lambda-1}$, it must delete every other run of $R^{\lambda-2}$ bits of the remaining $L/2$ bits, thus deleting $L/4$ more bits. Similarly, since σ deletes all zeros of $g_{\lambda-2}$, it must delete an additional $L/8$ bits. Continuing this logic, we have σ must delete $L(1-2^{-\lambda})$ bits total. This tells us that if an inner code deletion pattern completely corrupts the inner codewords $g_1, \ldots, g_\lambda$, it needs $L(1 - 2^{-\lambda})$ deletions. This logic works even if we chose to corrupt any subset of λ inner codewords other than $\{g_1, \ldots, g_\lambda\}$. One can imagine that corrupting (according to Definition 5.7.2) inner codewords is almost as hard as completely corrupting them, so adding some slack gives the lemma.

Lemma 5.7.4. *If σ is an inner code deletion pattern with $|\sigma| \le L\left(1 - \frac{1}{2^\lambda} - \frac{1}{\sqrt{R}}\right)$, then σ preserves all but at most $\lambda - 1$ choices of g_i.*

Proof. Suppose λ is a positive integer such that σ corrupts g_i for λ different values of i, say $i_1 > i_2 > \cdots > i_\lambda$. We wish to show $|\sigma| > L\left(1 - \frac{1}{2^\lambda} - \frac{1}{\sqrt{R}}\right)$.

Recall that a g_i segment is an interval of the form $[1 + aR^{i-1}, (a+1)R^{i-1}]$. Inductively define the collections of intervals $\mathcal{I}_1, \ldots, \mathcal{I}_\lambda$ and the sets of indices $I_1, \ldots, I_\lambda$ as follows. For $1 \le a \le \lambda$, set

$$\mathcal{I}_a \;=\; \left\{ J : J \text{ is } g_{i_a}\text{-segment, } J \subseteq \sigma \setminus \bigcup_{b=1}^{a-1} I_b \right\} \quad \text{and} \quad I_a \;=\; \bigcup_{J \in \mathcal{I}_a} J. \tag{5.9}$$

Intuitively, $\mathcal{I}_1$ as the set of runs in g_{i_1} that are entirely deleted by σ, and I_1 is the set of those deleted indices. Then, $\mathcal{I}_2$ is the set of runs of g_{i_2} deleted by σ but not already accounted for by $\mathcal{I}_1$, and I_2 is the set of bits in the runs of $\mathcal{I}_2$. We can interpret $\mathcal{I}_3, \ldots, \mathcal{I}_\lambda$ and $I_3, \ldots, I_\lambda$ similarly. By construction, $I_1, \ldots, I_\lambda$ are disjoint, and their union is a subset of σ (thought of as a subset of $[L]$), so $\sum_{b=1}^{\lambda} |I_b| \le |\sigma|$. It thus suffices to prove

$$\sum_{b=1}^{\lambda} |I_b| \;>\; L\left(1 - \frac{1}{2^\lambda} - \frac{1}{\sqrt{R}}\right). \tag{5.10}$$

Note that for any $j < j'$, every $g_{j'}$-segment is the disjoint union of $R^{j'-j}$ many g_j-segments. We thus have $[L]$ is the disjoint union of g_{i_a}-segments and I_b is also the disjoint union of g_{i_a}-segments when $b < a$ (and thus $i_b > i_a$). Hence, $[L] \setminus \cup_{b=1}^{a-1} I_b$ is the disjoint union of g_{i_a}-segments. Furthermore, as R is even, each I_b covers an even number of g_{i_a}-segments, so the segments of $[L] \setminus \cup_{b=1}^{a-1} I_b$ alternate between segments corresponding to runs of 0s in g_{i_a} and segments corresponding to runs of 1s in g_{i_a}. It follows that all runs of $g_{i_a} \setminus \cup_{b=1}^{a-1} I_b$ have length exactly $R^{i_a - 1}$, so the number of runs in the string $g_{i_a} \setminus \cup_{b=1}^{a-1} I_b$ is

$$\frac{L}{R^{i_a-1}} - \sum_{b=1}^{a-1} R^{i_b - i_a} |\mathcal{I}_b|. \tag{5.11}$$

By construction, the only g_{i_a}-segments that are deleted by σ are the intervals covered by $I_1 \cup \cdots \cup I_a$. Since σ corrupts g_{i_a}, we know $\sigma(g_{i_a})$ has less than $L/(R^{i_a-1}\sqrt{R})$ runs. By Lemma 5.7.3, we have

$$|\mathcal{I}_a| \;>\; \frac{1}{2}\left(\left(\frac{L}{R^{i_a-1}} - \sum_{b=1}^{a-1} R^{i_b-i_a}|\mathcal{I}_b|\right) - \frac{L}{R^{i_a-1}\sqrt{R}}\right). \tag{5.12}$$

Simplifying and using $|I_b| = R^{i_b-1}|\mathcal{I}_b|$ for all b, we obtain

$$|I_a| \;>\; L\left(\frac{1}{2} - \frac{1}{2\sqrt{R}}\right) - \frac{1}{2}\sum_{b=1}^{a-1} |I_b|. \tag{5.13}$$

From here it is easy to verify by induction that, for all $1 \le a \le \lambda$, we have

$$\sum_{b=1}^{a} |I_b| \;>\; L\left(1 - \frac{1}{2^a}\right)\left(1 - \frac{1}{\sqrt{R}}\right). \tag{5.14}$$

Indeed, (5.13) for $a = 1$ provides the base case, and if we know (5.14) for some $a-1$, then by (5.13) we have

$$\begin{aligned}\sum_{b=1}^{a}|I_b| &> L\left(\frac{1}{2}-\frac{1}{2\sqrt{R}}\right)+\frac{1}{2}\sum_{b=1}^{a-1}|I_b| \\ &> L\left(\frac{1}{2}-\frac{1}{2\sqrt{R}}\right)+\frac{L}{2}\left(1-\frac{1}{2^{a-1}}\right)\left(1-\frac{1}{\sqrt{R}}\right) \\ &= L\left(1-\frac{1}{2^a}\right)\left(1-\frac{1}{\sqrt{R}}\right), \end{aligned} \tag{5.15}$$

completing the induction. The induction proves (5.10), from which we have

$$|\sigma| \geq \sum_{b=1}^{\lambda}|I_b| > L\left(1-\frac{1}{2^\lambda}-\frac{1}{\sqrt{R}}\right), \tag{5.16}$$

as desired. □

Lemma 5.7.4 motivates the following definition.

Definition 5.7.5. We say an inner code deletion pattern $|\sigma|$ is *ℓ-admissible* if $|\sigma| \leq L(1-1/2^{\ell+1}-\frac{1}{\sqrt{R}})$.

If, for some ℓ, σ is ℓ-admissible, then Lemma 5.7.4 tells us σ corrupts at most ℓ different g_i. However, note that ℓ-admissibility is stronger that corrupting at most ℓ different g_i as ℓ-admissibility gives a stronger upper bound on the number of deletions in σ, which is necessary in Lemma 5.7.12.

Lemma 5.7.6. *Let $\delta > 0$. Let $\tau = \tau_1 \frown \cdots \frown \tau_n$ be a deletion pattern with at most $(1-\frac{1}{2^\lambda}-\frac{1}{\sqrt{R}}-\frac{\delta}{2})N$ deletions. There are at least δn indices i such that τ_i is $(\lambda-1)$-admissible.*

Proof. By a simple counting argument, we have $|\tau_i| > L(1-\frac{1}{2^\lambda}-\frac{1}{\sqrt{R}})$ for at most

$$\frac{n\cdot L\left(1-\frac{1}{2^\lambda}-\frac{1}{\sqrt{R}}-\frac{\delta}{2}\right)}{L\left(1-\frac{1}{2^\lambda}-\frac{1}{\sqrt{R}}\right)} \leq n\left(1-\frac{\delta/2}{1-2^{-\lambda}}\right) \leq n(1-\delta) \tag{5.17}$$

values of i. For the remaining at least δn values of i, we have τ_i is $(\lambda-1)$-admissible. □

The following corollary allows us to reduce our analysis of a deletion pattern τ to analyzing positions where τ's inner code deletion pattern is $(\lambda-1)$-admissible. We effectively assume that our deletion pattern completely deletes all inner codewords with a non-admissible index.

Lemma 5.7.7. *Let $\tau \in \mathcal{D}(nL, pnL)$. There exists $\tau' \in \mathcal{D}(\delta nL)$, $\sigma \in \mathcal{D}(n, (1-\delta)n)$ and sets $S_1, \dots, S_{\delta n}$ such that*

1. *$|S_i| = \lambda - 1$ for all i,*
2. *for all $X \in [K]^n$, we have $\tau'(\psi(\sigma(X))) \sqsubseteq \tau(\psi(X))$, and*
3. *when we write $\tau' = \tau'_1 \frown \cdots \frown \tau'_{\delta n}$ as the concatenation of δn inner code deletion patterns, we have, for all i and all $j \notin S_i$, that $\tau'_i \in \mathcal{D}(L)$ preserves g_j.*

Proof. Let $\tau = \tau_1 \frown \cdots \frown \tau_n$. By Lemma 5.7.6, there exist δn indices $\ell_1 < \cdots < \ell_{\delta n}$ such that, for $i = 1, \ldots, \delta n$, τ_{ℓ_i} is $(\lambda - 1)$-admissible. Choose $\sigma \in \mathcal{D}(n, (1-\delta)n)$ via $\sigma(X_1 \ldots X_n) = X_{\ell_1} X_{\ell_2} \ldots X_{\ell_{\delta n}}$, and choose $\tau' \in \mathcal{D}(\delta n L)$ via $\tau' = \tau_{\ell_1} \frown \cdots \frown \tau_{\ell_{\delta n}}$. We have $\tau' \circ \psi \circ \sigma(X) \sqsubseteq \tau \circ \psi(X)$ for all $X \in [K]^n$ because $\tau' \circ \psi \circ \sigma(X)$ is simply the result of deleting the remaining bits in inner codewords of non-admissible indices in $\tau \circ \psi(X)$. By construction, each $\tau_{\ell_i} \in \mathcal{D}(L)$ is $(\lambda - 1)$-admissible, so we can choose $S_1, \ldots, S_{\delta n}$ by setting S_ℓ to be the set that τ_{ℓ_i} corrupts. Note that some S_i may have size less than $\lambda - 1$, but we can arbitrarily add elements of $[K]$ to S_i until it has $\lambda - 1$ elements. This is okay as item 3 in the corollary statement remains true if we add elements to S_i. $\square$

In our analysis, for a deletion pattern $\tau = \tau_1 \frown \cdots \frown \tau_n$, we only care about the behavior of a given inner code deletion pattern τ_i as far as the set S_i of inner codewords g_j that it corrupts; instead of considering all possible deletion patterns $\tau \in \mathcal{D}(nL)$, it suffices to only consider all possible $\sigma, S_1, \ldots, S_{\delta n}$. This motivates the following definition.

Definition 5.7.8. The *signature* of a deletion pattern τ is $(\sigma, S_1, S_2, \ldots, S_{\delta n})$, where $\sigma, S_1, \ldots, S_{\delta n}$ are given by Lemma 5.7.7. If the choice of $\sigma, S_1, \ldots, S_{\delta n}$ satisfying the conditions of Lemma 5.7.7 are not unique, then choose one such collection of $\sigma, S_1, \ldots, S_{\delta n}$ arbitrarily and assign this collection to be the signature of τ.

Below we define the matchability relation $\prec$. Definition 5.7.11 allows us to worry only about the signature of a deletion pattern τ rather than τ itself. Intuitively, we can think of the matchability relation $\prec$ as an approximation of the subsequence relation $\sqsubseteq$. Proposition 5.7.12 establishes this relationship formally. Specifically, it states that if τ is a deletion pattern with signature $(\sigma, S_1, \ldots, S_{\delta n})$, then for $X, Y \in [K]^n$, we have $\sigma(X) \sqsubseteq Y$ implies that $\sigma(X)$ has a matching in Y with appropriate parameters. This means that if we want to show there are few incorrectly decoded codewords in a given code, it suffices to show that few codewords have an appropriately parameterized matching in some other codeword.

We first define type-A and type-B pairs of indices $(i, j) \in \{1, \ldots, |X|\} \times \{1, \ldots, |Y|\}$. Intuitively, pairs (i, j) are type-B only if $\tau_i(\psi(X_i))$ has many more runs than $\psi(Y_j)$, i.e. it is difficult to find subwords of $\tau_i(\psi(X_i))$ as subsequences of $\psi(Y_j)$.

Definition 5.7.9. Let $X, Y \in [K]^*$ be words over the alphabet $[K]$ and let $S_1, \ldots, S_{|X|}$ be subsets of K. Given a pair $(i, j) \in \{1, \ldots, |X|\} \times \{1, \ldots, |Y|\}$, we say (i, j) is *type-A* with respect to $X, Y, S_1, \ldots, S_{|X|}$ (or simply *type-A* if the parameters are understood) if $X_i \in S_i$ or $X_i \geq Y_j$. Call a pair (i, j) *type-B* with respect to $X, Y, S_1, \ldots, S_{|X|}$ otherwise.

Definition 5.7.10. Let $X, Y \in [K]^*$ be words over the alphabet $[K]$, let $S_1, \ldots, S_{|X|}$ be subsets of K, and let s and t be positive integers. The following algorithm constructs the $(s, t, S_1, \ldots, S_{|X|})$ *matching* of X with Y. Begin with a pair $(a, b) = (1, 1)$. The first and second coordinates correspond to indices of the strings X and Y, respectively. Define an *A-move* to be incrementing the first coordinate, a, by 1, and a *B-move* to be incrementing of the second coordinate, b, by 1.

1. If $a = |X|$ or $b = |Y|$, stop.
2. Do one of the following
   (a) If the last s moves were A-moves, make a B-move.
   (b) Else if the last t moves were B-moves, make an A-move.
   (c) Else if (a, b) is type-A, make an A-move.
   (d) Else, (a, b) must be type-B, in which case make a B-move.

3. Repeat from step 1.

Note that at the end of this algorithm, exactly one of $a = |X|$ and $b = |Y|$ is true. We say this matching is a *success* if we ended with $a = |X|$, otherwise it is a *failure*.

Definition 5.7.11. Note also that the matching is uniquely determined by $X, Y, S_1, \dots, S_{|X|}, s, t$. If this matching is a success, we say X is $(s, t, S_1, \dots, S_{|X|})$-*matchable* (or has a $(s, t, S_1, \dots, S_{|X|})$-*matching*) in Y, denoted

$$X \prec_{(s,t,S_1,\dots,S_{|X|})} Y. \tag{5.18}$$

Proposition 5.7.12. *Let $S_1, \dots, S_{\delta n}$ be subsets of $[K]$ of size exactly $\lambda - 1$. Let $\tau = \tau_1 \frown \cdots \frown \tau_{\delta n} \in \mathcal{D}(nL)$ be a deletion pattern such that for all i, $\tau_i \in \mathcal{D}(L)$ is $(\lambda - 1)$-admissible and in particular preserves g_j for all $j \notin S_i$. Suppose we have $X \in [K]^{\delta n}$ and $Y \in [K]^n$ such that $\tau(\psi(X)) \sqsubseteq \psi(Y)$ (recall $\sqsubseteq$ is the subsequence relation). Then $X \prec_{(2^\lambda, \sqrt{R}, S_1, \dots, S_{\delta n})} Y$.*

Proof. Let $s = 2^\lambda, t = \sqrt{R}$. Run the matching algorithm defined above to obtain a matching of X and Y. Let $\mathcal{M}$ be the set of all (a, b) reached by some step of the algorithm. We wish to show this matching is a success, i.e. that there exists some b such that $(|X|, b) \in \mathcal{M}$, or, equivalently, there does not exist a such that $(a, |Y|) \in \mathcal{M}$.

Since $\tau(\psi(X)) \sqsubseteq \psi(Y)$, we can find $\tau(\psi(X))$ as a subsequence of $\psi(Y)$ by greedily matching the bits of $\tau(\psi(X))$ with the bits of $\psi(Y)$. Let $\mathcal{N}$ be the set of (i, k) such that some bit of $\tau_i(\psi(X_i))$ is matched with some bit in $\psi(Y_k)$. We first establish some basic facts about $\mathcal{M}, \mathcal{N}$.

Fact 5.7.13.
1. *$(|X|, |Y|) \notin \mathcal{M}$.*
2. *$\mathcal{M}, \mathcal{N} \subseteq \{1, \dots, |X|\} \times \{1, \dots, |Y|\}$.*
3. *For all $a^* \in \{1, \dots, |X|\}, b^* \in \{1, \dots, |Y|\}$, we have $\{b : (a^*, b) \in \mathcal{M}\}$ and $\{a : (a, b^*) \in \mathcal{M}\}$ are intervals of consecutive integers of lengths at most $t + 1$ and $s + 1$, respectively.*
4. *For all $i^* \in \{1, \dots, |X|\}, k^* \in \{1, \dots, |Y|\}$, we have $\{k : (i^*, k) \in \mathcal{N}\}$ and $\{i : (i, k^*) \in \mathcal{N}\}$ are intervals of consecutive integers.*
5. *Let $\le_{\mathcal{M}}$ be a relation on $\mathcal{M}$ such that $(a, b) \le_{\mathcal{M}} (a', b')$ if and only if $a \le a'$ and $b \le b'$. Then $\mathcal{M}$ is totally ordered under $\le_{\mathcal{M}}$. As such, we can define $\operatorname{next}_{\mathcal{M}}(a, b)$ and $\operatorname{prev}_{\mathcal{M}}(a, b)$ to be the next larger and next smaller element after (a, b) under $\le_{\mathcal{M}}$, respectively. Then $\operatorname{next}_{\mathcal{M}}(a, b) \in \{(a + 1, b), (a, b + 1)\}$ and $\operatorname{prev}_{\mathcal{M}}(a, b) \in \{(a - 1, b), (a, b - 1)\}$.*
6. *Let $\le_{\mathcal{N}}$ be a relation on $\mathcal{N}$ such that $(i, k) \le_{\mathcal{N}} (i', k')$ if and only if $i \le i'$ and $k \le k'$. Then $\mathcal{N}$ is totally ordered under $\le_{\mathcal{N}}$. As such, we can define $\operatorname{next}_{\mathcal{N}}(i, k)$ and $\operatorname{prev}_{\mathcal{N}}(i, k)$ to be the next larger and next smaller element after (i, k) under $\le_{\mathcal{N}}$, respectively. Then $\operatorname{next}_{\mathcal{N}}(i, k) \in \{(i + 1, k), (i, k + 1), (i + 1, k + 1)\}$ and $\operatorname{prev}_{\mathcal{N}}(i, k) \in \{(i - 1, k), (i, k - 1), (i - 1, k - 1)\}$.*
7. *If $(a, b), (a', b') \in \mathcal{M}$, we never have both $a < a'$ and $b' < b$.*
8. *If $(i, k), (i', k') \in \mathcal{N}$, we never have both $i < i'$ and $k' < k$.* □

For $a \in \{1, \dots, |X|\}$ and $b \in \{1, \dots, |Y|\}$, define

$$\begin{aligned}
\alpha_0(b) &= \min\{a : (a, b) \in \mathcal{M}\} & \alpha_f(b) &= \max\{a : (a, b) \in \mathcal{M}\} \\
\beta_0(a) &= \min\{b : (a, b) \in \mathcal{M}\} & \beta_f(a) &= \max\{b : (a, b) \in \mathcal{M}\} \\
\iota_0^{(b)} &= \min\{i : (i, b) \in \mathcal{N}\} & \iota_f^{(b)} &= \max\{i : (i, b) \in \mathcal{N}\} \\
\kappa_0^{(a)} &= \min\{k : (a, k) \in \mathcal{N}\} & \kappa_f^{(a)} &= \max\{k : (a, k) \in \mathcal{N}\}.
\end{aligned} \tag{5.19}$$

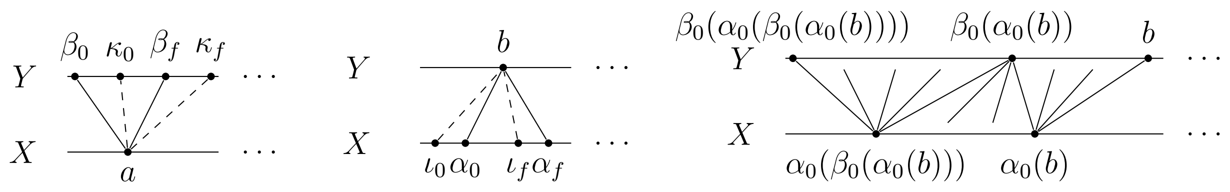


Figure 5.1: Illustrations of $\alpha_0(b), \alpha_f(b), \beta_0(a), \beta_f(a), \iota_0(b), \iota_f(b), \kappa_0(a), \kappa_f(a)$

See Figure 5.1 for illustrations of the behavior of these eight functions. We first establish a few facts about the notation $\alpha_0, \beta_0, \ldots$ that are helpful for developing intuition and are also useful later. These proofs are more involved than Fact 5.7.13 and are provided.

Lemma 5.7.14. *1. For all $a \in \{1, \ldots, |X|\}$, if $\kappa_f(a) - \kappa_0(a) < \sqrt{R}$, then there exists $b' \in [\kappa_0(a), \kappa_f(a)]$ such that (a, b') is type-A.*

2. For all $a \in \{1, \ldots, |X|\}$ we have $\beta_f(a) - \beta_0(a) \le \sqrt{R}$ and for all $\beta_0(a) \le b' < \beta_f(a)$ we have (a, b') is type-B Furthermore, if $\beta_f(a) - \beta_0(a) < \sqrt{R}$, then $(a, \beta_f(a))$ is type-A.

3. For all $b \in \{1, \ldots, |Y|\}$, we have $\iota_f(b) - \iota_0(b) \le 2^\lambda$.

4. For all $b \in \{1, \ldots, |Y|\}$, we have (i', b) is type-A for all $i' \in [\iota_0(b) + 1, \iota_f(b) - 1]$.

5. For all $b \in \{1, \ldots, |Y|\}$, we have $\alpha_f(b) - \alpha_0(b) \le 2^\lambda$ and for all $a' \in [\alpha_0(b), \alpha_f(b) - 1]$ we have (a', b) is type-A. Furthermore, if $\alpha_f(b) - \alpha_0(b) < 2^\lambda$, then $(\alpha_f(b), b)$ is type-B.

Proof. Parts 2 and 5 follow from Definition 5.7.10.

For part 1, suppose for contradiction that (a, b') is type-B for all $\kappa_0(a) \le b' \le \kappa_f(a)$. Then $X_a \notin S_a$ and $X_a < Y_{b'}$ for all such b'. This means $\tau_a(\psi(X_a))$ has at least $2R^{K+\frac{1}{2}-X_a}$ runs while $\psi(Y_{b'})$ has at most $2R^{K+1-(X_a+1)}$ runs for $\kappa_0(a) \le b' \le \kappa_f(a)$. On the other hand, we have

$$\tau_a(\psi(X_a)) \sqsubseteq \psi\left(Y_{\kappa_0(a)} \ldots Y_{\kappa_f(a)}\right). \tag{5.20}$$

As $\kappa_f(a) - \kappa_0(a) < \sqrt{R}$ this means the right side of (5.20) has less than $\sqrt{R} \cdot 2R^{K-X_a} = 2R^{K+\frac{1}{2}-X_a}$ runs while the left side has at least that many runs, a contradiction.

For part 3, suppose for contradiction that $\iota_f(b) - \iota_0(b) - 1 \ge 2^\lambda$. Since $\psi(Y_b)$ contains $\prod_{i'=\iota_0(b)+1}^{\iota_f(b)-1} \tau_{i'}(\psi(X_{i'}))$ as a strict subsequence ($\psi(Y_b)$ additionally contains at least one bit from each of $\tau_{\iota_0}(\psi(X_{\iota_0}))$ and $\tau_{\iota_f}(\psi(X_{\iota_f}))$), we have

$$L + 2 \;\le\; (\iota_f(b) - \iota_0(b) - 1) \cdot \frac{L}{2^\lambda} + 2 \;\le\; \left(\sum_{i'=\iota_0+1}^{\iota_f-1} |\tau_{i'}(\psi(X_{i'}))|\right) + 2 \;\le\; |\psi(Y_b)| \;=\; L, \tag{5.21}$$

a contradiction.

For part 4, suppose for contradiction that (i', b) is type-B for some $\iota_0(b) < i' < \iota_f(b)$. Thus $X_{i'} \notin S_{i'}$ and $X_{i'} < Y_b$. In particular, $\tau_{i'}(\psi(X_{i'}))$ has at least $2R^{K+\frac{1}{2}-X_{i'}}$ runs, which is more than the at-most-$2R^{K-X_{i'}}$ runs of $\psi(Y_b)$. However, $\iota_0(b) < i' < \iota_f(b)$, so $(i', b) \in \mathcal{N}$ and in particular $\tau_{i'}(\psi(X_{i'})) \sqsubseteq \psi(Y_b)$, which is a contradiction. Note that $i' = \iota_0(b)$ and $i' = \iota_f(b)$ do not guarantee a contradiction because for such i', some bits of $\tau_{i'}(\psi(X_{i'}))$ might be matched with other inner codewords in $\psi(Y)$. □

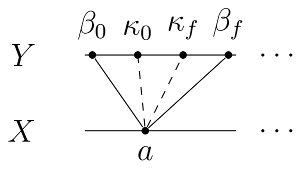


Figure 5.2: Step 2, Assume FSOC $\beta_0(a) \leq \kappa_0(a) \leq \kappa_f(a) < \beta_f(a)$

The following definition of proper indices is introduced for convenience. Intuitively, indices of Y are Y-proper if the bit-matching $\mathcal{N}$ consumes corresponding indices of X "slower" than in the algorithmic matching $\mathcal{M}$, and indices of X are X-proper if the bit-matching $\mathcal{N}$ consumes corresponding indices of Y "faster" than in the algorithmic matching $\mathcal{M}$.

Definition 5.7.15. We say an index in $a \in \{1, \ldots, |X|\}$ is X*-proper* if

$$\beta_0(a) \ \leq \ \kappa_0(a), \qquad \beta_f(a) \ \leq \ \kappa_f(a) \tag{5.22}$$

and we say an index $b \in \{1, \ldots, |Y|\}$ is Y*-proper* if

$$\iota_0(b) \ \leq \ \alpha_0(b), \qquad \iota_f(b) \ \leq \ \alpha_f(b). \tag{5.23}$$

Claim 5.7.16. *For all $a \in \{1, \ldots, |X|\}$ and all $b \in \{1, \ldots, |Y|\}$, a is X-proper and b is Y-proper.*

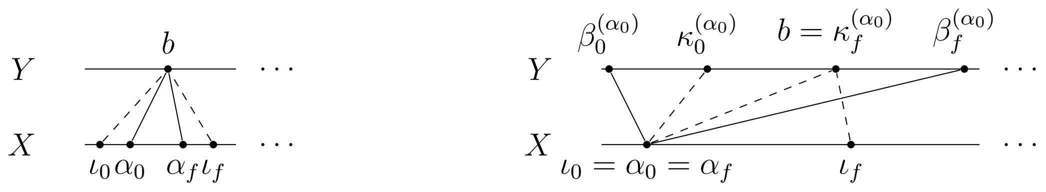
Remark 5.7.17. First we illustrate how the Claim 5.7.16 implies Proposition 5.7.12. Suppose for contradiction that Proposition 5.7.12 is false. Then there exists some a such that $(a, |Y|) \in \mathcal{M}$ and for all b we have $(|X|, b) \notin \mathcal{M}$. In particular, $\alpha_f(b) < |X|$ for all $b \in \{1, \ldots, |Y|\}$. By the claim, $\iota_f^{(|Y|)} \leq \alpha_f^{(|Y|)} < |X|$, implying that no bits from $\tau_{|X|}(\psi(X_{|X|}))$ are matched with bits of $\psi(Y)$, a contradiction of $\tau(\psi(X)) \sqsubseteq \psi(Y)$.

Proof. **Step 1.** First, note that, as $(1,1) \in \mathcal{M}, \mathcal{N}$, we have $\alpha_0^{(1)} = \beta_0^{(1)} = \iota_0^{(1)} = \kappa_0^{(1)} = 1$.

Step 2. Next, we show that for all a, if $\beta_0(a) \leq \kappa_0(a)$ then a is X-proper. That is, we show $\beta_f(a) \leq \kappa_f(a)$. Suppose for contradiction we have $\kappa_f(a) < \beta_f(a)$ so that $\beta_0(a) \leq \kappa_0(a) \leq \kappa_f(a) < \beta_f(a)$ (see Figure 5.2). By Lemma 5.7.14 part 2, we have $\beta_f(a) - \beta_0(a) \leq \sqrt{R}$ and (a, b') is type-B for all $b' \in [\beta_0(a), \beta_f(a) - 1]$. In particular, (a, b') is type-B for all $b' \in [\kappa_0(a), \kappa_f(a)]$. As $\kappa_f(a) - \kappa_0(a) < \beta_f(a) - \beta_0(a)$, we have $\kappa_f(a) - \kappa_0(a) < \sqrt{R}$, so we can apply Lemma 5.7.14 part 1 to obtain that (a, b') is type-A for some $b' \in [\kappa_0(a), \kappa_f(a)]$. This is a contradiction as all such b' must be type-B.

Step 3. Next, we show that for all b, if $\iota_0(b) \leq \alpha_0(b)$ and $\alpha_0(b)$ is X-proper, then b is Y-proper. That is, we show $\iota_f(b) \leq \alpha_f(b)$. Suppose for contradiction that $\alpha_f(b) < \iota_f(b)$ so that $\iota_0(b) \leq \alpha_0(b) \leq \alpha_f(b) < \iota_f(b)$ (see Figure 5.3a). We have $\alpha_f - \alpha_0 < \iota_f - \iota_0 \leq 2^\lambda$ by Lemma 5.7.14 part 3. Thus, by Lemma 5.7.14 part 5, $(\alpha_f(b), b)$ is type-B. By Lemma 5.7.14 part 4, (i', b) is type-A for $i' \in [\iota_0(b) + 1, \iota_f(b) - 1]$. Since $\alpha_f(b) \in [\iota_0(b), \iota_f(b) - 1]$ we must have $\alpha_f(b) = \iota_0(b)$, so $\iota_0(b) = \alpha_0(b) = \alpha_f(b)$ (See Figure 5.3b). By definition of $\alpha_f(b)$, we have $\text{next}_{\mathcal{M}}(\alpha_f(b), b) = (\alpha_f(b), b{+}1)$ so $\beta_f^{(\alpha_f(b))} \geq b{+}1$. However, since we assumed $\alpha_f < \iota_f$, we have $\text{next}_{\mathcal{N}}(\alpha_f(b), b) = (\alpha_f(b){+}1, b)$, so $\kappa_f(\alpha_f(b)) = b$. Thus

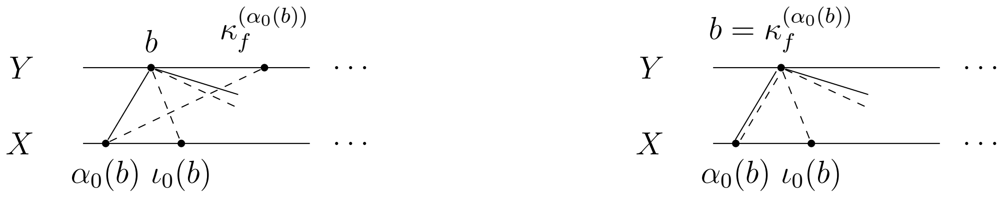
$$\beta_f(\alpha_0(b)) = \beta_f(\alpha_f(b)) \geq b + 1 > b = \kappa_f(\alpha_f(b)) = \kappa_f(\alpha_0(b)). \tag{5.24}$$



(a) Assume FSOC $\iota_0(b) \le \alpha_0(b) \le \alpha_f(b) < \iota_f(b)$

(b) $\iota_0(b) = \alpha_0(b) = \alpha_f(b) < \iota_f(b)$

Figure 5.3: Step 3



Y
X
b
$\kappa_f^{(\alpha_0(b))}$
$\alpha_0(b)$ $\iota_0(b)$
$b = \kappa_f^{(\alpha_0(b))}$
$\alpha_0(b)$ $\iota_0(b)$

(a) $b < \kappa_f(\alpha_0(b))$

(b) $b = \kappa_f(\alpha_0(b))$

Figure 5.4: Step 4

On the other hand, $\beta_f^{(\alpha_0(b))} \le \kappa_f^{(\alpha_0(b))}$ by assumption that $\alpha_0(b)$ is X-proper, which is a contradiction. This covers all possible cases, completing Step 3.

Step 4. We prove that, for all $b \in \{1, \dots, |Y|\}$, if $\alpha_0(b)$ is X-proper, then b is Y-proper. By Step 3, it suffices to prove $\iota_0(b) \le \alpha_0(b)$. Suppose for contradiction that $\alpha_0(b) < \iota_0(b)$. Since $(\alpha_0(b), b) \in \mathcal{M}$, we have $b \le \beta_f(\alpha_0(b))$. By assumption, $\alpha_0(b)$ is X-proper, so $\beta_f(\alpha_0(b)) \le \kappa_f(\alpha_0(b))$, which means $b \le \kappa_f(\alpha_0(b))$. If $b < \kappa_f^{(\alpha_0)}$, then we have $(\alpha_0(b), \kappa_f(\alpha_0(b))) \in \mathcal{N}$. However, $(\iota_0(b), b) \in \mathcal{N}$, contradicting Fact 5.7.13 part 8. Thus, $\kappa_f(\alpha_0(b)) = b$. But then $(\alpha_0(b), b) = (\alpha_0(b), \kappa_f(\alpha_0(b))) \in \mathcal{N}$ with $\alpha_0(b) < \iota_0(b)$, contradicting the minimality of $\iota_0(b)$.

Step 5. By the same argument as Step 4, we have that, for all $a \in \{1, \dots, |X|\}$, if $\beta_0(a)$ is Y-proper, then a is X-proper.

Step 6. We prove by strong induction that for pairs (a, b), ordered by $<_{\mathcal{M}}$, we have a is X-proper and b is Y-proper. Combining Steps 1,2, and 3, we have

$$\beta_0^{(1)} \le \kappa_0^{(1)}, \qquad \beta_f^{(1)} \le \kappa_f^{(1)}, \qquad \iota_0^{(1)} \le \alpha_0^{(1)}, \qquad \iota_f^{(1)} \le \alpha_f^{(1)}, \tag{5.25}$$

where the first and third inequalities are actually equalities arising from Step 1, the second inequality is established by Step 2, and the fourth is established by Step 3. Thus, 1 is both X-proper and Y-proper.

Now suppose we have some pair $(a, b) \in \mathcal{M}$ with $(1, 1) <_{\mathcal{M}} (a, b)$ and (5.20) has been established for all smaller pairs. If $\mathrm{prev}_{\mathcal{M}}(a, b) = (a - 1, b)$, then by the inductive hypothesis, we have b is Y-proper. However, as $(a - 1, b) \in \mathcal{M}$, we have $(a, b - 1) \notin \mathcal{M}$, so we have $\beta_0(a) = b$. Thus $\beta_0(a)$ is Y-proper, so a is X-proper by Step 5. Similarly, if $\mathrm{prev}_{\mathcal{M}}(a, b) = (a, b - 1)$, then by the inductive hypothesis, we have a is X-proper. Thus $\alpha_0(b) = a$ is X-proper, so b is Y-proper by Step 4. This completes the proof of Claim 5.7.16, proving Proposition 5.7.12. $\square$

The following proposition is the key result of this subsection. Following our approach, it should

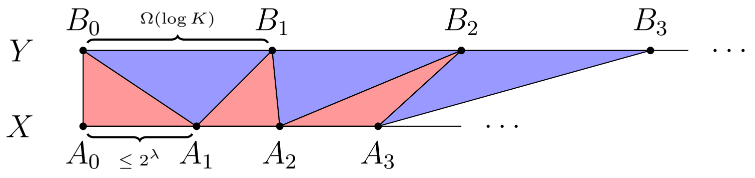


Figure 5.5: A_i's and B_i's behavior

be possible to prove this proposition for any choice of $S_1, \dots, S_{\delta n}$, not just $[\lambda-1], \dots, [\lambda-1]$. However, because of Lemma 5.7.23, which tells us that $[\lambda-1], \dots, [\lambda-1]$ is the "worst" possible choice of $S_1, \dots, S_{\delta n}$, it suffices to prove the proposition as stated.

Proposition 5.7.18. *There exists a constant $\beta > 0$ such that for any fixed deletion pattern $\sigma \in \mathcal{D}(n, (1-\delta)n)$ we have*

$$\Pr_{X,Y \sim U([K]^n)}[\sigma(X) \prec_{(2^\lambda, \sqrt{R}, [\lambda-1], \dots, [\lambda-1])} Y] < 2^{-\beta n}. \tag{5.26}$$

Proof. Let $s = 2^\lambda$. With hindsight, let $\beta = \frac{\log K}{16R}$. It suffices to prove

$$\Pr_{\substack{X \sim U([K]^{\delta n}) \\ Y \sim U([K]^n)}} \left[X \prec_{(2^\lambda, \sqrt{R}, [\lambda-1], \dots, [\lambda-1])} Y\right] < 2^{-\beta n}. \tag{5.27}$$

This suffices because for any σ, the distribution of $\sigma(X)$ for $X \sim U([K]^n)$ is the same as $X \sim U([K]^{\delta n})$.

Let $X_1, \dots, X_{\delta n}, Y_1, Y_2, \dots$ be independently chosen from $[K]$. Let $X = X_1 \dots X_{\delta n}$, $Y = Y_1 \dots Y_n$, and $Y_\infty = Y_1 Y_2 \dots$ so that $X \sim U([K]^{\delta n})$ and $Y \sim U([K]^n)$.

Construct a $(2^\lambda, \sqrt{R}, S_1, \dots, S_{\delta n})$-matching of X in Y_∞. Note that, as Y_∞ is an infinite random string, $\mathcal{M}$ succeeds almost surely, i.e. ends in $(|X|, b)$ for some integer b.

Let $\mathcal{M}$ be the set of all reached states (a, b) in the matching, and let $\alpha_f(b) = \max\{a : (a,b) \in \mathcal{M}\}$ and $\beta_f(a) = \max\{b : (a,b) \in \mathcal{M}\}$ as in Proposition 5.7.12. Let $A_0 = 1, B_0 = 1$. For $i \ge 1$, set $A_i = \alpha_f(B_{i-1})$ and $B_i = \beta_f(A_{i-1})$. As $A_i - A_{i-1} \le s$ for all $i \ge 1$, we have A_i is well defined for $i \le \delta n/s - 1$. By the definition of matching, $\mathcal{M}'$ succeeds if and only if $\mathcal{M}$ succeeds and the final position $(|X|, \beta_f(|X|))$ satisfies $\beta_f(|X|) < |Y|$. It thus suffices to prove

$$\Pr\left[B_{\lfloor \delta n/s \rfloor - 1} < |Y|\right] < 2^{-\beta n}. \tag{5.28}$$

The key idea of this proof is that the B_i's grow much faster than the A_i's, so that the B_i's "run out of indices in Y" faster than the A_i's "run out of indices in X", even though Y is longer than X. In particular, by definition of a matching, we have $A_{i+1} - A_i \le s = 2^\lambda$, but, on the other hand, we show that $B_{i+1} - B_i$ is, in expectation, $\Omega(\log K)$ (see Figure 5.5).

We have two technical lemmas. The proofs are straightforward, and we include them in Appendix D for completeness.

Lemma 5.7.19. *Let J be chosen uniformly from $[K]$. Let D be a random variable that is 1 if $J \in [\lambda - 1]$ and, conditioned on a fixed $J \ge \lambda$, is distributed as* $\min(\text{Geometric}(J/K), \sqrt{R})$. *Then* $\mathbf{E}[D] \ge (\log K)/4$.

Lemma 5.7.20. *Let $\lambda' \in [\lambda, K]$ and let J be chosen uniformly from $\{\lambda, \lambda+1, \ldots, \lambda'\}$. Let D be the random variable that, conditioned on a fixed J, is distributed as $\min(\mathrm{Geometric}(J/K), \sqrt{R})$. Then $\mathbf{E}[D] \geq (\log K)/4$.*

Claim 5.7.21. *Let $i \geq 1$. For any fixed $A_0, \ldots, A_i, B_0, \ldots, B_i, X_1, \ldots, X_{A_i}, Y_1, \ldots, Y_{B_i}$, we have*

$$\mathbf{E}\left[B_{i+1} - B_i | A_1, \ldots, A_i, B_1, \ldots, B_i, X_1, \ldots, X_{A_i}, Y_1, \ldots, Y_{B_i}\right] > \frac{\log K}{4}. \tag{5.29}$$

Proof. It suffices to prove that if we additionally condition on $A_{i+1} - A_i < s$, then the expectation is at least $(\log K)/4$, and that the same is true if we condition on $A_{i+1} - A_i = s$.

First, note that, for $1 \leq b < \sqrt{R}$, we have $B_{i+1} = B_i + b$ if and only if $(A_{i+1}, B_i + j)$ is type-B for $j \in \{1, \ldots, B_i + b - 1\}$ and $(A_{i+1}, B_i + b)$ is type-A. If no such b exists, we have $B_{i+1} - B_i = \sqrt{R}$. Thus, conditioned on fixed $A_0, \ldots, A_{i+1}, B_0, \ldots, B_i, X_1, \ldots, X_{A_{i+1}}, Y_1, \ldots, Y_{B_i}$, we have $B_{i+1} - B_i$ is distributed as $\min(\mathrm{Geometric}(X_{A_{i+1}}/K), \sqrt{R})$.

Suppose we condition on $A_{i+1} - A_i = s$. This is equivalent to saying $(A_i + \ell, B_i)$ is type-A for $\ell = 1, \ldots, s-1$. However, this assertion depends only on $X_{A_i}, X_{A_i+1}, \ldots, X_{A_i+s-1}$, which are independent of $X_{A_{i+1}}$, so we have $X_{A_{i+1}}$ is still distributed uniformly on $[K]$. If $X_{A_{i+1}} \in [\lambda - 1]$, then $B_{i+1} - B_i = 1$, otherwise $B_{i+1} - B_i$ is distributed as $\min(\mathrm{Geometric}(X_{A_{i+1}}/K), \sqrt{R})$, so by Lemma 5.7.19 on $D = B_{i+1} - B_i$, we have

$$\mathbf{E}[B_{i+1} - B_i | A_{i+1} - A_i = s, A_1, \ldots, A_i, B_1, \ldots, B_i, X_1, \ldots, X_{A_i}, Y_1, \ldots, Y_{B_i}] > \frac{\log K}{4}. \tag{5.30}$$

Suppose we condition on $A_{i+1} - A_i = s' < s$. This is equivalent to saying $(A_i + \ell, B_i)$ is type-A for $\ell = 1, \ldots, s'-1$ and $(A_i + s', B_i)$ is type-B. This implies $X_{A_{i+1}} \geq \lambda$ (i.e. $X_{A_{i+1}} \notin [\lambda - 1]$) and $X_{A_{i+1}} < Y_{B_i}$, Since Y_{B_i} is fixed and, without any conditioning, $X_{A_{i+1}}$ is distributed uniformly in $[K]$, we have $X_{A_{i+1}}$ is distributed uniformly on $[\lambda, \ldots, Y_{B_i}]$. By the previous argument, we have $B_{i+1} - B_i$ is distributed as $\min(\mathrm{Geometric}(X_{A_{i+1}}/K), \sqrt{R})$, so by Lemma 5.7.20 on $\lambda' = Y_{B_i}$ and $D = B_{i+1} - B_i$, we have

$$\mathbf{E}[B_{i+1} - B_i | A_{i+1} - A_i = s' < s, A_1, \ldots, A_i, B_1, \ldots, B_i, X_1, \ldots, X_{A_i}, Y_1, \ldots, Y_{B_i}] > \frac{\log K}{4}. \tag{5.31}$$

□

Corollary 5.7.22. *We have, for any fixed $B_1, \ldots, B_i$,*

$$\mathbf{E}[B_{i+1} - B_i | B_1, \ldots, B_i] > \frac{1}{4} \log K. \tag{5.32}$$

Continuing the proof of Proposition 5.7.18, let $\alpha = \frac{1}{4} \log K$ so that

$$\alpha(k-1) \ > \ \frac{1}{4} \log K \cdot \left(\frac{\delta n}{2^\lambda} - 1\right) \ > \ 2n. \tag{5.33}$$

Define the random variable $B_i' := B_i - i \cdot \alpha$. As B_i' and B_i uniquely determine each other, we have, by Corollary 5.7.22,

$$\mathbf{E}[B_{i+1}' - B_i' | B_1', \ldots, B_i'] = \mathbf{E}[B_{i+1} - B_i | B_1, \ldots, B_i] - \alpha > 0. \tag{5.34}$$

Thus, B'_i form a submartingale with $|B'_{i+1} - B'_i| < \sqrt{R}$, so by Azuma's Inequality (Lemma 2.2.3), we have

$$\mathbf{Pr}\left[B'_k - B'_1 \leq -\frac{\alpha(k-1)}{2}\right] \leq \exp\left(-\frac{(\alpha(k-1)/2)^2}{2(k-1)\cdot(\sqrt{R})^2}\right) = \exp\left(-\frac{\alpha^2(k-1)}{8R}\right) < \exp\left(-\frac{n\log K}{16R}\right). \tag{5.35}$$

Combining with (5.33) gives

$$\mathbf{Pr}[B_k \leq n] \leq \mathbf{Pr}[B'_k - B'_1 \leq -n] \leq \mathbf{Pr}\left[B'_k - B'_1 \leq -\frac{\alpha(k-1)}{2}\right] < \exp\left(-\frac{n\log K}{16R}\right). \tag{5.36}$$

We conclude

$$\mathop{\mathbf{Pr}}_{\substack{X\sim U([K]^{\delta n})\\ Y\sim U([K]^n)}}[X \prec_{(\sqrt{R},2^\lambda,[\lambda-1],\dots,[\lambda-1])} Y] \leq \mathbf{Pr}[m_{\delta n} \leq n] \leq \mathbf{Pr}[B_k \leq n] < \exp\left(-\frac{n\log K}{16R}\right). \tag{5.37}$$

As $\exp(-\frac{n\log K}{16R}) < 2^{-\frac{n\log K}{16R}}$, we have (5.27) is true, completing the proof of Proposition 5.7.18. ☐

To conclude this section, we formalize the intuition that the "worst possible deletion pattern", that is, the deletion pattern that makes decoding most difficult in some sense, is the deletion pattern such that each $(\lambda-1)$-admissible inner deletion pattern corrupts $g_1,\dots,g_{\lambda-1}$. This fact is intuitive, as it is hard, for example, to find g_1 as a subsequence of a random $\psi(Y)$ because g_1 has many runs. Thus if our deletion pattern τ corrupts the inner codewords with the most runs, there is a greater probability that over random X, Y we have $\tau(\psi(X))$ is a subsequence of $\psi(Y)$. However, note that, to make a clean assertion, we continue to argue using the matching relation rather than subsequence relation.

Lemma 5.7.23. *Let $r, s \in \mathbb{N}$ and $S_1,\dots,S_{\delta n}$ be subsets of $[K]$ of size exactly $\lambda - 1$. Then for all $Y \in [K]^n$ we have*

$$\#\left\{X \in [K]^{\delta n} : X \prec_{(2^\lambda,\sqrt{R},S_1,\dots,S_{\delta n})} Y\right\} \leq \#\left\{X \in [K]^{\delta n} : X \prec_{(2^\lambda,\sqrt{R},[\lambda-1],\dots,[\lambda-1])} Y\right\} \tag{5.38}$$

Proof. We start with a claim.

Claim 5.7.24. *For every set $A \subseteq [K]$ with size exactly $\lambda-1$, there exists a bijection $h_A : [K] \to [K]$ such that $h_A(x) \in [\lambda-1]$ for $x \in A$ and $h_A(x) \geq x$ for $x \notin A$.*

Proof. Pair each element $x \in A \setminus [\lambda-1]$ with an element $y \in [\lambda-1] \setminus A$ arbitrarily and for each pair (x, y) set $h_A(x) = y, h_A(y) = x$. Note this always gives $x > y$. This is possible as $A \setminus [\lambda-1]$ and $[\lambda-1] \setminus A$ have the same size. Then set $h_A(x) = x$ for all other x. It is easy to check this function satisfies $h_A(x) \geq x$ for $x \notin A$. ☐

Fix $Y \in [K]^n$. For all i, let $h_i : S_i \to [\lambda-1]$ be a bijection such that $h_i(x) \in [\lambda-1]$ for $x \in S_i$ and $f(x) \geq x$ for all other x. This exists by Claim 5.7.24. Let $h : [K]^{\delta n} \to [K]^{\delta n}$ be such that $h(X_1,\dots,X_{\delta n}) = h_1(X_1)h_2(X_2)\cdots h_{\delta n}(X_{\delta n})$. Since each of $h_1,\dots,h_{\delta n}$ are bijections, h is a bijection as well.

Let X be such that $X \prec_{(2^\lambda, \sqrt{R}, S_1, \dots, S_{\delta n})} Y$. We claim $h(X) \prec_{(2^\lambda, \sqrt{R}, [\lambda-1], \dots, [\lambda-1])} Y$. Let $\mathcal{M}$ be a $(2^\lambda, \sqrt{R}, S_1, \dots, S_{\delta n})$-matching of X in Y and let $\mathcal{M}'$ be a $(2^\lambda, \sqrt{R}, [\lambda-1], \dots, [\lambda-1])$-matching of $h(X)$ in Y. We know $\mathcal{M}$ is a successful matching, and we wish to show $\mathcal{M}'$ is a successful matching.

If (a, b) is type-A with respect to $X, Y, S_1, \dots, S_{|X|}$, then $Y_b \leq X_a$ or $X_a \in S_a$, so, by definition of h, either $h(X_a) \in [\lambda-1]$ or $Y_b \leq X_a \leq h(X_a)$, so (a, b) is type-A with respect to $h(X), Y, [\lambda-1], \dots, [\lambda-1]$. Let $\ell \geq 0$ be an integer and let $(a, b) \in \mathcal{M}$ and $(a', b') \in \mathcal{M}'$ be the state of the matchings after the ℓth step of the matchings of X in Y and $h(X)$ in Y, respectively. It is easy to see by induction that $a \leq a'$ and $b' \leq b$; if $a < a'$ then after one move we still have $a \leq a'$, and if $a = a'$ and $\mathcal{M}$s next move is an A-move, then $\mathcal{M}'$s next move must also be an A-move. Since $\mathcal{M}$ succeeds, we have $(|X|, b) \in \mathcal{M}$ for some $b < |Y|$, so we conclude that $(|X|, b') \in \mathcal{M}'$ for some $b' \leq b < |Y|$. Thus $\mathcal{M}'$ succeeds as well.

Since h is a bijection such that

$$h\left(\left\{X \in [K]^{\delta n} : X \prec_{(2^\lambda, \sqrt{R}, S_1, \dots, S_{\delta n})} Y\right\}\right) \subseteq \left\{X \in [K]^{\delta n} : X \prec_{(2^\lambda, \sqrt{R}, [\lambda-1], \dots, [\lambda-1])} Y\right\}, \tag{5.39}$$

we have

$$\#\left\{X \in [K]^{\delta n} : X \prec_{(2^\lambda, \sqrt{R}, S_1, \dots, S_{\delta n})} Y\right\} \leq \#\left\{X \in [K]^{\delta n} : X \prec_{(2^\lambda, \sqrt{R}, [\lambda-1], \dots, [\lambda-1])} Y\right\} \tag{5.40}$$

as desired. □

5.8 Technical combinatorial lemmas

Lemma 5.8.1. *Let n be a positive integer and suppose we have $0 < \beta < 1$. Suppose that $\gamma = \beta/3$, $M = 2^{\gamma n}$, and $\epsilon \geq 2^{-(\gamma - o(1))n/2}$. Suppose G is a graph on $N = K^{(1-o(1))n}$ vertices such that each vertex has degree at most $d = Nr$ for some $r = r(n) = 2^{-(\beta - o(1))n}$. Suppose C_{out} is chosen as a random subset of $V(G)$ by including each vertex of $V(G)$ in C_{out} with probability M/N. Then for sufficiently large n, we have $\mathbf{Pr}_{C_{out}}[|E(G|_{C_{out}})| > \epsilon M] < 2^{-\omega(n)}$.*

Remark 5.8.2. Essentially this lemma is saying that when the edge density is extremely small, around $2^{-\beta n}$, then for all but an extremely small set of choices for C_{out}, C_{out} is extremely sparse.

Proof. Let $E = E(G)$. We know E satisfies $|E| \leq dN/2 < N^2 r$.

Enumerate the edges $1, \dots, |E|$ arbitrarily. Let $Y_1, \dots, Y_{|E|}$ be Bernoulli random variables denoting whether the ith edge is in C_{out}. Let $Z = Y_1 + \cdots + Y_{|E|}$. We would like to show

$$\mathop{\mathbf{Pr}}_{C_{out}}[Z > \epsilon M] < 2^{-n}. \tag{5.41}$$

We do this by computing a sufficiently large moment of Z and using

$$\mathop{\mathbf{Pr}}_{C_{out}}[Z > \epsilon M] \leq \frac{\mathbf{E}[Z^k]}{(\epsilon M)^k}. \tag{5.42}$$

For a tuple of (not necessarily distinct) edges $(e_1, \dots, e_k)$, denote $V(e_1, \dots, e_k)$ to be the set of vertices on at least one of the edges $e_1, \dots, e_k$. Alternatively, we say $V(e_1, \dots, e_k)$ is the set of vertices *covered* by edges $e_1, \ldots, e_k$. Note that for all $e_1, \ldots, e_k$, we have $2 \le V(e_1, \ldots, e_k) \le 2k$. Let $P_{k,\ell} = \#\{(e_1, \ldots, e_k) : |V(e_1, \ldots, e_k)| = \ell\}$ denote the number of ordered tuples of edges from E that cover exactly ℓ vertices.

Claim.

$$P_{k,\ell} \;<\; N^\ell \cdot r^{\ell/2} \cdot \ell^{2k+\ell-1} \cdot k^k \tag{5.43}$$

Proof. We bound $P_{k,\ell}$ by first bounding the number of sets (unordered tuples) of edges covering exactly ℓ vertices and then multiply by $k!$. We first compute the number of ways to construct a forest of trees covering ℓ vertices using only edges from E. We do this by casework on the number of connected components. Let $1 \le c \le \lfloor \ell/2 \rfloor$ be an integer. To construct a forest with c connected components, we first choose c disjoint edges $e_1, \ldots, e_c$. This can be done in at most $|E|^c$ ways. We have $|V(e_1, \ldots, e_c)| = 2c$. For $c+1 \le i \le \ell - c$, choose a vertex $v \in V(e_1, \ldots, e_{i-1})$ and an edge vw such that $w \notin V(e_1, \ldots, e_{i-1})$. By construction, for $c+1 \le i \le \ell - c$, we have $|V(e_1, \ldots, e_i)| = i + c$. Thus the i^{th} edge can be added in at most $(i+c-1)d$ ways. The $i = \ell - c^{\text{th}}$ edge completes a forest covering ℓ vertices. Recalling that $|E| < N^2 r$ and $d = Nr$, we have that the total number of ways to construct a forest in this fashion is at most

$$|E|^c \cdot 2cd \cdot (2c+1)d \cdots (\ell-1)d \;\le\; |E|^c \cdot d^{\ell-2c} \cdot \ell^{\ell-2c} \;<\; N^\ell \cdot r^{\ell-c} \cdot \ell^{\ell-2c}. \tag{5.44}$$

We have used $\ell - c$ edges thus far. There are $k - \ell + c$ remaining edges. As $|V(e_1, \ldots, e_k)| = \ell$, these remaining edges must connect one of the $\binom{\ell}{2}$ pairs of vertices in $V(e_1, \ldots, e_{k-\ell+c})$. As $c \le \lfloor \ell/2 \rfloor < \ell$, we have $k - \ell + c < k$. The remaining edges can thus be chosen in at most $\binom{\ell}{2}^{k-\ell+c} < \ell^{2k}$ ways. Using (5.44) and multiplying by $k!$ we have

$$P_{k,\ell} \;<\; k! \cdot \sum_{c=1}^{\ell/2} N^\ell \cdot r^{\ell-c} \cdot \ell^{\ell-2c} \cdot \ell^{2k} \;<\; k^k \cdot (\ell/2) \cdot N^\ell \cdot r^{\ell/2} \cdot \ell^{\ell-2} \cdot \ell^{2k} \;<\; N^\ell \cdot r^{\ell/2} \cdot \ell^{2k+\ell-1} \cdot k^k. \tag{5.45}$$

□

Note that $M\sqrt{r} \ll 1$. With this claim, we have

$$\begin{aligned}
\mathbf{E}[Z^k] &= \mathbf{E}[(Y_1 + \cdots + Y_{|E|})^k] \\
&= \sum_{(e_1,\ldots,e_k) \in \{1,\ldots,|E|\}^k} \mathbf{E}[Y_1 \cdots Y_k] \\
&= \sum_{\ell=2}^{2k} \sum_{|V(e_1,\ldots,e_k)|=\ell} \mathbf{E}[Y_{e_1} \cdots Y_{e_k}] \\
&= \sum_{\ell=2}^{2k} \sum_{|V(e_1,\ldots,e_k)|=\ell} \left(\frac{M}{N}\right)^\ell \\
&= \sum_{\ell=2}^{2k} \left(\frac{M}{N}\right)^\ell \cdot P_{k,\ell} \;<\; \sum_{\ell=2}^{2k} \left(\frac{M}{N}\right)^\ell \cdot N^\ell \cdot r^{\ell/2} \cdot \ell^{2k+\ell-1} \cdot k^k \\
&< \sum_{\ell=2}^{2k} \ell^{2k+\ell-1} \cdot k^k \;<\; 2k \cdot (2k)^{4k-1} \cdot k^k = 2^{4k} \cdot k^{5k}.
\end{aligned} \tag{5.46}$$

Finally, choosing $k = \log n$, we have

$$\mathbf{Pr}[Z \geq \epsilon M] \;\leq\; \frac{\mathbf{E}[Z^k]}{(\epsilon M)^k} \;<\; \frac{2^{4k}k^{5k}}{\epsilon^k M^k} \;<\; \left(\frac{16k^5}{\epsilon 2^{\gamma n}}\right)^k \;\leq\; \left(\frac{16k^5}{2^{\gamma n/2}}\right)^k \;<\; 2^{-\omega(n)} \tag{5.47}$$

as desired. □

Lemma 5.8.3. *Let n be a positive integer, and suppose $0 < \beta < 1$ and $\gamma = \beta/4$. Suppose $M = 2^{\gamma n}$ and $\epsilon \geq 2^{-\gamma n/2}$. Suppose G is a directed graph on $N = K^{n(1-o(1))}$ vertices such that each vertex has outdegree at most Nr for some $r = 2^{-(\beta - o(1))n}$. Choose a subset $C_{out} \subseteq V(G)$ at random by including each vertex of $V(G)$ in C_{out} with probability M/N so that $\mathbf{E}[|C_{out}|] = M$. Then, for sufficiently large n, we have*

$$\mathop{\mathbf{Pr}}_{C_{out}}\left[\#\left\{X \in C_{out} : \exists Y \in C_{out}\,\text{s. t.}\,\overrightarrow{YX} \in E(G)\right\} > \epsilon|C_{out}|\right] \;<\; 2^{-\omega(n)}. \tag{5.48}$$

Proof. As, by assumption, Y has outdegree at most $N \cdot r$ for all $Y \in V(G)$, the average indegree of G is at most $N \cdot r$. Thus at most $\epsilon/8$ fraction of all words in $V(G)$ have indegree larger than $\frac{8}{\epsilon} \cdot N \cdot r$. Call this set of vertices W. We have

$$\mathop{\mathbf{E}}_{C_{out}}\left[\#\left\{X \in C_{out} : \text{indeg}_G(X) > \frac{8}{\epsilon} \cdot N \cdot r\right\}\right] \;=\; \mathbf{E}[|C_{out} \cap W|] \;\leq\; \frac{\epsilon}{8}M. \tag{5.49}$$

Since $|C_{out} \cap W|$ is the sum of i.i.d Bernoulli random variables, we have by Lemma 2.2.1

$$\mathop{\mathbf{Pr}}_{C_{out}}\left[\#\left\{X \in C_{out} : \text{indeg}_G(X) > \frac{8}{\epsilon} \cdot N \cdot r\right\} > \frac{\epsilon}{4}M\right] \;=\; \mathbf{Pr}\left[|C_{out} \cap W| > \frac{\epsilon}{4}M\right] \;<\; e^{-\frac{1}{2}\cdot\frac{\epsilon}{8}\cdot M} < 2^{-\omega(n)}. \tag{5.50}$$

Now consider the undirected graph H on $V(G)$ such that $XY \in E(H)$ if $\overrightarrow{XY} \in E(G)$ and $Y \notin W$ or $\overrightarrow{YX} \in E(G)$ and $X \notin W$. For every vertex v, the in-edges of v in G correspond to edges in H only if the indegree is at most $\frac{8}{\epsilon} \cdot N \cdot r$, and the outdegree of v in G is always at most Nr. Therefore the degree of every vertex in H is at most $r'N$ where $r' = \left(\frac{8}{\epsilon} + 1\right) \cdot r < 2^{-(\beta - \gamma - o(1))n}$.

We are including each vertex of $V(G)$ (and thus each vertex of $V(H)$) in C_{out} independently with probability M/N. Let $\epsilon' = \epsilon/4, \beta' = \frac{3}{4}\beta, \gamma' = \gamma$ so that $\gamma' = \beta'/3, \epsilon \geq 2^{-(\gamma + o(1))n/2}, M = 2^{\gamma n}$, and $r' = 2^{-(\beta' - o(1))n}$. By Lemma 5.8.1 for $\beta', \gamma', M, \epsilon'$, and r', and H, we have for sufficiently large n that

$$\mathop{\mathbf{Pr}}_{C_{out}}\left(E(H|_{C_{out}}) \;>\; \frac{\epsilon}{4}M\right) < 2^{-\omega(n)}. \tag{5.51}$$

Also, by Chernoff bounds, we have

$$\mathop{\mathbf{Pr}}_{C_{out}}\left[|C_{out}| \;<\; \frac{3}{4}M\right] < 2^{-\Omega(M)} \;\Longrightarrow\; \mathop{\mathbf{Pr}}_{C_{out}}\left[|C_{out}| \;<\; \frac{3}{4}M\right] < 2^{-\omega(n)}. \tag{5.52}$$

Note that if the number of X such that $\text{indeg}_{G|_{C_{out}}}(X) > 0$ at least $\epsilon|C_{out}|$, that is,

$$\#\left\{X \in C_{out} : \exists Y \in C_{out}\,\text{s. t.}\,\overrightarrow{YX} \in E(G)\right\} > \epsilon|C_{out}|, \tag{5.53}$$

then one of the following must be false.

1. $|C_{out}| \geq \frac{3}{4}M$
2. $\#\{X \in C_{out} : \deg_{H|_{C_{out}}}(X) > 0\} \leq \frac{\epsilon}{2}M$
3. $|C_{out} \cap W| \leq \frac{\epsilon}{4}M$

Indeed, suppose to the contrary all of these were true. The number of vertices in $C_{out} \cap W$ with positive indegree in $G|_{C_{out}}$ is at most $|C_{out} \cap W| \leq \frac{\epsilon}{4}M$. If a vertex $X \in C_{out} \setminus W$ has positive indegree in $G|_{C_{out}}$ then there exists $Y \in C_{out}$ such that $\overrightarrow{YX} \in E(G)$, so $XY \in E(H)$ by definition of H and thus $\deg_{H|_{C_{out}}}(X) > 0$. Thus the number of vertices in $C_{out} \setminus W$ with positive indegree is at most $\frac{\epsilon}{2}M$ by property 2. Hence the total number of vertices in C_{out} with positive indegree in $G|_{C_{out}}$ is at most $\frac{3}{4}\epsilon M \leq \epsilon |C_{out}|$.

Each of items 1, 2, and 3 is false with probability $2^{-\omega(n)}$ by (5.50), (5.51), and (5.52), so the probability any of them occur is at most $2^{-\omega(n)}$, as desired. □

5.9 Proof of construction (Theorem 5.4.2)

Proof of Theorem 5.4.2. We would like to show there exists a code C and an $\epsilon = o_N(1)$ such that, for any deletion pattern τ deleting pN bits,

$$\#\{x \in C : \exists y \in C \text{ s.t. } x \neq y \text{ and } \tau(x) \leq y\} \;\leq\; \epsilon |C|. \tag{5.54}$$

For $Y \in [K]^n$, define

$$f(Y) \;=\; \mathop{\mathbf{Pr}}_{Z \sim U([K]^{\delta n})} \left[Z \prec_{(2^\lambda, \sqrt{R}, [\lambda-1], \dots, [\lambda-1])} Y \right]. \tag{5.55}$$

Let $\sigma \in \mathcal{D}(n, (1-\delta)n)$ be a deletion pattern for our outer code $[K]^n$. Let $S_1, \dots, S_{\delta n}$ be subsets of $[K]$ of size exactly $\lambda - 1$. Let $G^*_{\sigma, S_1, \dots, S_{\delta n}}$ be the graph on the vertex set $[K]^n$ such that $\overrightarrow{YX}$ is an edge if and only if $\sigma(X) \prec_{(2^\lambda, \sqrt{R}, S_1, \dots, S_{\delta n})} Y$. Note that for all $\sigma \in \mathcal{D}(n, (1-\delta)n)$ and all $Y \in [K]^n$ we have, by Lemma 5.7.23,

$$\begin{aligned}
\#\{X \in [K]^n : \sigma(X) \prec_{(2^\lambda, \sqrt{R}, S_1, \dots, S_{\delta n})} Y\} &= K^{(1-\delta)n} \cdot \#\{Z \in [K]^{\delta n} : Z \prec_{(2^\lambda, \sqrt{R}, S_1, \dots, S_{\delta n})} Y\} \\
&\leq K^{(1-\delta)n} \cdot \#\{Z \in [K]^{\delta n} : Z \prec_{(2^\lambda, \sqrt{R}, [\lambda-1], \dots, [\lambda-1])} Y\} \\
&= K^n \cdot f(Y).
\end{aligned} \tag{5.56}$$

In the graph language, this means every $Y \in [K]^n$ has outdegree at most $K^n \cdot f(Y)$ in every $G^*_{\sigma, S_1, \dots, S_{\delta n}}$.

We remove all Y with large $f(Y)$. Let $\beta = \log K / 16R$. By Proposition 5.7.18, we know the average value of $f(Y)$ is $2^{-\beta n}$. Hence at most $K^n/2$ such Y satisfy $f(Y) \geq 2 \cdot 2^{-\beta n}$. These are the *easily disguised* candidate codewords mentioned in §5.4. There is thus a set $W \subseteq [K]^n$ such that $|W| = K^n/2$ and each $Y \in W$ satisfies $f(Y) < 2 \cdot 2^{-\beta n} = 2^{-(\beta - o(1))n}$. For all $\sigma \in \mathcal{D}(n, (1-\delta)n), S_1, \dots, S_{\delta n}$, consider the subgraph $G_{\sigma, S_1, \dots, S_{\delta n}} := G^*_{\sigma, S_1, \dots, S_{\delta n}}|_W$. By construction, for all $\sigma, S_1, \dots, S_{\delta n}$, the outdegree of every vertex of $G_{\sigma, S_1, \dots, S_{\delta n}}$ is at most $(K^n/2) \cdot r$ for some $r = 2^{-(\beta - o(1))n}$.

Let $\gamma = \beta/4$. Let $M = 2^{\gamma n}$ and $\epsilon = 2^{-\gamma n/2}$. Choose a subset $C_{out} \subseteq W$ by including each vertex of W in C_{out} independently with probability $M/|W|$ so that $\mathbf{E}[|C_{out}|] = M$.

By Lemma 5.8.3 for $N = K^n/2, \beta, \gamma, M, \epsilon, r$, and $G_{\sigma,S_1,\dots,S_{\delta n}}$, we have that, for all σ and sufficiently large n,

$$\Pr_{C_{out}}\left[\#\{X \in C_{out} : \exists Y \in C_{out} \text{ s. t. } \overrightarrow{YX} \in E(G_{\sigma,S_1,\dots,S_{\delta n}})\} > \epsilon|C_{out}|\right] < 2^{-\omega(n)}. \tag{5.57}$$

This is equivalently

$$\Pr_{C_{out}}\left[\#\{X \in C_{out} : \exists Y \in C_{out} \text{ s. t. } \sigma(X) \prec_{(2^\lambda,\sqrt{R},S_1,\dots,S_{\delta n})} Y\} > \epsilon|C_{out}|\right] < 2^{-\omega(n)}. \tag{5.58}$$

Union bounding over the at-most-2^n possible values of σ and $\binom{K}{\lambda-1}^{\delta n}$ values of $S_1,\dots,S_{\delta n}$ gives

$$\begin{aligned}\Pr_{C_{out}}&\left[\exists \sigma, S_1,\dots,S_{\delta n} \text{ s. t. } \#\big\{X \in C_{out} : \exists Y \in C_{out} \text{ s. t. } \sigma(X) \prec_{(2^\lambda,\sqrt{R},S_1,\dots,S_{\sigma n})} Y\big\} > \epsilon|C_{out}|\right]\\ &< 2^n \cdot \binom{K}{\lambda-1}^{\delta n} \cdot 2^{-\omega(n)}\\ &< 2^n \cdot 2^{n\delta(\lambda-1)\log K} \cdot 2^{-\omega(n)} = 2^{-\omega(n)}.\end{aligned} \tag{5.59}$$

Additionally, with probability approaching 1, $|C_{out}| \ge \frac{3}{4}M$. Thus, there exists a C_{out} with $|C_{out}| \ge \frac{3}{4}M$ such that the following holds for all $\sigma \in \mathcal{D}(n,(1-\delta)n)$ and all $S_1,\dots,S_{\delta n} \in \binom{[K]}{\lambda-1}$: at most $\epsilon|C_{out}|$ of the codewords X are $(2^\lambda, \sqrt{R}, S_1,\dots,S_{\delta n})$ matchable in some other codeword $Y \in C_{out}$.

By Lemma 5.7.7, there exists $\sigma \in D(n,(1-\delta)n)$ and $\tau' \in \mathcal{D}(\delta n L)$ such that the following holds: If we write $\tau' = \tau_1' \frown \cdots \frown \tau_{\delta n}'$ then each τ_i' is $(\lambda-1)$-admissible and preserves all g_j for all j not in some size-$\lambda-1$ set S_i, and furthermore we have $\tau'(\psi(\sigma(X))) \sqsubseteq \tau(\psi(X))$ for all $X \in [K]^n$. By choice of C_{out}, for at least $(1-\epsilon)|C_{out}|$ choices of $X \in C_{out}$, $\sigma(X)$ is not $(2^\lambda, \sqrt{R}, S_1,\dots,S_{\delta n})$ matchable in all $Y \in C_{out}$. Thus, for these X, we have $\tau'(\psi(\sigma(X))) \not\sqsubseteq \psi(Y)$ for all $Y \in C_{out}$ by Lemma 5.7.12. Since $\tau'(\psi(\sigma(X))) \sqsubseteq \tau(\psi(X))$ for all $X \in [K]^n$, we have for these $(1-\epsilon)|C_{out}|$ choices of X that $\tau(\psi(X)) \not\sqsubseteq \psi(Y)$ for all $Y \in C_{out}$. Thus, choosing $C = \psi(C_{out})$ gives our desired average deletion code.

Recall λ is the smallest integer such that $(1+p)/2 < 1-2^{-\lambda}$ and $\delta = 1-2^{-\lambda}-p$ so that $2^\lambda = \Theta(1/(1-p))$ and $\delta = \Theta(1-p)$. Recall $K = 2^{\lceil 2^{\lambda+5}/\delta\rceil}$ and $R = 4K^4$ so that $K = 2^{\Theta(1/(1-p)^2)}$. The rate of the outer code is $\log 2^{\gamma n}/n\log K$ and the rate of the inner code is $\log K/L$. The total rate is thus

$$\mathcal{R} = \frac{\log K}{48R \cdot 2R^K} = 2^{-2^{\Theta(1/(1-p)^2)}}. \tag{5.60}$$

This completes the proof of Theorem 5.4.2. □

Chapter 6

Online Deletions

6.1 Introduction and related work

The online model sits between the oblivious and adversarial noise models. An online channel chooses whether to erase/flip/delete the ith bit c_i of a codeword c without knowledge of the future bits of the codeword, and is limited to corrupting at most pn bits total for some fixed p. Chen, Jaggi, and Langberg [5], in an impressive work, determined the exact capacities of the online erasure channel and the online bit-flip channel. A recent work studies a seemingly slight (but in fact fundamental) restriction of the online model where the channel's decision regarding the i'th bit depends only on the first $(i-1)$ bits and is independent of the current bit [10]. They proved that in the setting of erasures, the capacity in this restricted online model increases to match the capacity $1-p$ of the binary erasure channel, as opposed to $1-2p$ in the true online model. To our knowledge, codes against online deletions have not been studied.

In §6.2, we present Theorem 6.2.1, a result from [17]. We then outline our lower-bound-type proof in §6.3 and present a full proof in §6.4.

6.2 New result relating $p_0^{(adv)}$ with $p_0^{(on)}$

Formally, an online deletion channel OnAdv consists of n functions $\{\mathrm{OnAdv}_i : i \in [n]\}$ such that $\mathrm{OnAdv}_i : \mathcal{X}^i \times \mathcal{Y}^{i-1} \to \mathcal{Y}$, where $X = \{0,1\}$ and $Y = \{\langle 0\rangle, \langle 1\rangle, \langle\rangle\}$ is a set of strings and $\langle\rangle$ denotes the empty string, satisfies $\mathrm{OnAdv}(x_1,\ldots,x_i,y_1,\ldots,y_{i-1}) \in \{\langle x_i\rangle, \langle\rangle\}$. The resulting string received by the output is the concatenation of the outputs of $\mathrm{OnAdv}_1,\ldots,\mathrm{OnAdv}_n$. Note that all online adversaries are valid adversaries in the omniscient adversary case, so all codes correcting against pn adversarial deletions can also correct against pn online deletions. Notice that an online channel can delete all the 0s or 1s in the string, so as in the case of adversarial deletions, it is impossible to communicate with positive rate when the deletion fraction is at least $\frac{1}{2}$. However, as the omniscient adversary has more information than the online channel, it could be the case that one can code against a fraction of deletions approaching $\frac{1}{2}$ for the online channel, whereas this is not possible against an omniscient adversary. Our next result rules out this possibility — if the zero-rate threshold $p_0^{(adv)}$ for adversarial deletions is bounded away from $1/2$, then so is the zero-rate threshold for online deletions.

Theorem 6.2.1. *For any $p > 1/(3-2p_0^{(adv)})$ and $\mathcal{R} > 0$, there exists a deterministic online channel*

OnAdv *such that, for sufficiently large* n*, for any code* C *of block length* n *and rate* $\mathcal{R} > 0$*, and any decoder* $\mathrm{Dec} : \{0,1\}^{(1-p)n} \to C$*, we have* $\mathbf{Pr}_{c \in C}[\mathrm{Dec}(\mathrm{OnAdv}(c)) \neq c] \geq \eta$ *for some absolute constant* $\eta > 0$.

In contrapositive form, the above states that in order to construct codes against adversarial deletions with error fraction approaching $\frac{1}{2}$, it suffices to construct deterministic codes decodable in the average case against a fraction of *online* deletions approaching $\frac{1}{2}$.

Note that since online deletions contain as a special case oblivious deletions, if we insist on deterministic codes where every codeword is correctly decoded, then the question is just as hard as adversarial deletions. So it is important to allow some error, either in the form of randomized encoding, or some small fraction of codewords to be decoded incorrectly (i.e., an average-error criterion). This is why the online strategy of Theorem 6.2.1 ensures that a constant fraction of codewords are miscommunicated.

6.3 Outline of theorem 6.2.1 proof

We give a high level outline of the proof of Theorem 6.2.1. Recall that the goal is to prove that, assuming $p_0^{(adv)} < \frac{1}{2}$, for some $p < \frac{1}{2}$, for every code of rate bounded away from 0, there exists a deterministic online deletion channel applying up to pn deletions that prevents successful decoding in the average case. That is, when a uniformly random codeword is transmitted, the probability of incorrect decoding is bounded away from 0. Note first that it suffices to find a *randomized* online deletion strategy guaranteeing average-case decoding error because, if for a given code the channel has a random strategy to guarantee average-case decoding error, then sampling a strategy over the randomness of the channel gives a deterministic online strategy that inflicts similar probability of incorrect decoding.

To do this, we assume that the adversarial zero rate threshold is $p_0^{(adv)} < \frac{1}{2}$, and set $p_0^{(on)} = 1/(3 - 2p_0^{(adv)})$. Our choice of $p_0^{(on)}$ satisfies $p_0^{(adv)} < p_0^{(on)} < \frac{1}{2}$. Now we show that for any $p_0^{(on)} < p < \frac{1}{2}$, there exists a randomized strategy that inflicts pn deletions in an online manner to a deterministic code, and guarantees that the probability, over the randomness of a uniformly chosen codeword and the randomness of the adversary, of incorrect decoding is positive (we show it is at least $\frac{1}{10}$).

We adapt the "wait-push" strategy of Bassily and Smith [2] for an online adversary against erasures. In the "wait phase" of this strategy, the adversary observes a prefix x^* of ℓ bits from the transmitted word x without erasing any bits. Then it constructs a list $\mathcal{L}_{x^*}$ of candidate codewords, among which is the actual codeword chosen by the encoder. In the "push phase", the adversary chooses a codeword x' randomly from $\mathcal{L}_{x^*}$, which, with positive probability, corresponds to a different message than the actually transmitted word. The adversary then, for the last $n - \ell$ bits of the transmission, erases bit x_i of x where $x_i \neq x'_i$ (x'_i the ith bit of x').

Our strategy adapts the wait push strategy to deletions. In our wait phase, we observe a prefix of the string until we know the codeword x *exactly*. We choose in advance a random bit and delete every instance of that bit that we see during the wait phase. After the wait phase, suppose we have seen qn bits and deleted rn bits, where $r < q$, and we also know the codeword x exactly. Using the definition of $p_0^{(adv)}$, we can show that, for most choices of x, there is another codeword y such that (i) the wait phase lengths qn for x and y are the same, (ii) their majority bits b in the wait phases are the same, (iii) the number of bits rn deleted in the wait phases is the same, and (iv) the length-$(1-q)n$ suffixes x^* and y^* of x and y, respectively, have a common subsequence s^* satisfying $|s^*| > (1-p_0^{(adv)})|x^*|$. We can form pairs of such codewords (x, y) in advance, so that when we see either x or y we push the suffixes x^* or y^* to s^*, so that the received word in both cases is $b^{rn}s^*$. For these typical choices of x, the total number of required deletions to obtain $b^{rn}s^*$ is at most $qn/2 + p_0^{(adv)}(1-q)n$. If q is bounded away from 1 (in our case, less than $(1-p)n$), then we can bound the number of deletions away from $\frac{1}{2}$ (in our case, less than $p_0^{(on)} = 1/(3-2p_0^{(adv)})$). On the other hand, if q is large (larger than $(1-p)n$), then we can simply observe the first $(1-p)n$ bits and delete the rest. Since we can only choose one strategy to run, we choose one of these two strategies at random. This gives our modified wait push strategy.

6.4 Relating zero-rate threshold of online and adversarial deletions

Recall that the *zero-rate threshold for adversarial deletions*, $p_0^{(adv)}$, is the supremum of all p such that there exists a family of code $C \subseteq \{0,1\}^n$ with $|C| = 2^{\Omega(n)}$ satisfying $\mathrm{LCS}(C) > (1-p)n$. Additionally, recall that the *zero-rate threshold for online deletions*, $p_0^{(on)}$, is the supremum of p such that there exist families of deterministic codes with rate bounded away from 0 that can correct against pn online deletions in the *average case* (when a uniformly random codeword is transmitted). That is, when a uniformly random codeword is transmitted, the probability of a decoding error, over the choice of the codeword and possibly the randomness of the online channel strategy, is $o(1)$ in the block length of the code. Since an online strategy can guess the minority bit in the codeword and delete all its occurrences, adversary can always delete the majority bit, we trivially have

$$p_0^{(adv)} \le p_0^{(on)} \le \frac{1}{2}. \tag{6.1}$$

Recall that the currently best known lower bound on $p_0^{(adv)}$ is based on the code construction in [4] and implies $p_0^{(adv)} \ge \sqrt{2}-1$. An outstanding question in this area is whether $p_0^{(adv)} = 1/2$. Our main result in this section ties this question to the corresponding question for online deletions. Namely, we have

Theorem 6.4.1. *If $p_0^{(adv)} = \frac{1}{2}$ if and only if $p_0^{(on)} = \frac{1}{2}$.*

By virtue of (6.1), Theorem 6.4.1 follows if we show that $p_0^{(adv)} < \frac{1}{2}$ implies $p_0^{(on)} < \frac{1}{2}$. The idea to show this is based on a suitable adaptation of the "wait-push" idea of [2]. We wait qn bits until we know the codeword, then push so that it is confused with another codeword. In the first wait phase, we delete the minority bit for a budget of $qn/2$ deletions, and while we push we need at most $(1-q)np_0^{(adv)}$ deletions. If we assume that $p_0^{(adv)} < \frac{1}{2}$, we end up with a total of αn deletions for some $\alpha < \frac{1}{2}$. The exact details are more subtle, but this is the rough idea.

From the definition of $p_0^{(adv)}$, we have the following fact.

Fact 6.4.2. *For any $\mathcal{R} > 0$ and $p > p_0^{(adv)}$, there exists n_0 such that for all $n \ge n_0$, for any $C \subseteq \{0,1\}^n$ with $|C| \ge 2^{\mathcal{R}n}$, there exist two strings $x, y \in C$ such that* $\mathrm{LCS}(x,y) > (1-p)n$.

Corollary 6.4.3. *For any $\mathcal{R} > 0$ and $p > p_0^{(adv)}$, there exists n_0 such that for all $n \ge n_0$, for any*

$C \subseteq \{0,1\}^n$ with $|C| \geq 2^{\mathcal{R}n}$, there exists $C' \subseteq C$ such that $|C'| \geq |C| - 2^{\mathcal{R}n/2}$, and all elements of C' are confusable with another element of C. That is, for all $x \in C'$ there exists $y \in C$ such that $x \neq y$ and $\mathrm{LCS}(x,y) > (1-p)n$.

Proof. We can apply Fact 6.4.2 for $\mathcal{R}' = \mathcal{R}/2$. Start with $C' = \emptyset$. While $|C \setminus C'| > 2^{\mathcal{R}n/2}$, find an x that is confusable with some y in C and add it to C'. This is possible by Fact 6.4.2. $\square$

The following is the precise statement of our result relating the zero-rate thresholds for online and adversarial deletions.

Theorem 6.4.4. *We have*

$$p_0^{(on)} \leq \frac{1}{3 - 2p_0^{(adv)}}. \tag{6.2}$$

In particular, if $p_0^{(adv)} < \frac{1}{2}$, then $p_0^{(on)} < \frac{1}{2}$.

Remark 6.4.5. If it is the case that the bound in [4] is tight and $p_0^{(adv)} = \sqrt{2} - 1 \approx 0.414$, then Theorem 6.4.4 gives $p_0^{(on)} \leq 0.4605$.

The rest of this section is devoted to proving Theorem 6.4.4. We start with the following helpful definition.

Definition 6.4.6. Given a code C and a codeword $x \in C$ passing through an online deletion channel, define the *wait phase* and the *push phase* for a codeword as follows. Let $q_x n$ be the smallest index such that $x_1 \dots x_{q_x n}$ uniquely determines the codeword. The wait phase refers to the time until Adv receives bit $q_x n$, but not acted on it (chosen whether to transmit or delete it). Thus in the wait phase Adv receives bits $1, \dots, q_x n$ and has acted on bits $1, \dots, q_x n - 1$. The push phase is the time after the end of the wait phase. We say the *wait length* of x is $q_x n$ and the *push length* is $(1 - q_x)n$.

For a codeword x, let q_x denote the *relative wait length* (wait length divided by n), and let $r_{x,b}$ denote number b-bits that appear in $x_1 x_2 \dots x_{q_x n}$, so that $r_{x,0} + r_{x,1} = q_x$. Let $b_x \in \{0,1\}$ denote the bit that appears more frequently in $x_1 \dots x_{q_x n}$ (break ties arbitrarily).

Proof of Theorem 6.4.4. Since $p_0^{(on)} \leq \frac{1}{2}$, (6.2) holds if $p_0^{(adv)} = \frac{1}{2}$, so assume $p_0^{(adv)} < \frac{1}{2}$.

Let p be such that

$$\frac{1}{2} > p > \frac{1}{3 - 2p_0^{(adv)}}. \tag{6.3}$$

We show that, for large enough n, when Adv gets pn online deletions, he can, with constant probability independent of n, guarantee that, if the encoder sends $x \in C$, the decoder receives a string s that is a substring of $y \in C$ where $x \neq y$.

Call a pair of codewords x, y *online-confusable* if $q_x = q_y \leq 1 - p$, $r_{x,0} = r_{y,0}$, $b_x = b_y$, and

$$\mathrm{LCS}(x[q_x n :], y[q_x n :]) > (1 - q_x)(1 - p_0^{(adv)})n. \tag{6.4}$$

For an online-confusable pair x, y, let $s^*(x,y)$ denote a common subsequence of $x[q_x n :]$ and $y[q_x n :]$ with length at least $(1 - q_x)(1 - p_0^{(adv)})n$, and let $s(x,y)$ denote the string $(b_x)^{r_{x,b} n} s^*(x,y)$.

Note that $s(x,y)$ is a common substring of x and y and, by choice of p in (6.3) has length at least

$$\begin{aligned}
|s(x,y)| &= r_{x,b}n + (1-q_x)(1-p_0^{(adv)})n \\
&\geq \frac{q_x n}{2} + (1-q_x)(1-p_0^{(adv)})n \\
&= \left(1 - p_0^{(adv)} - \left(\frac{1}{2} - p_0^{(adv)}\right) q_x\right) n \\
&\geq \left(1 - p_0^{(adv)} - \left(\frac{1}{2} - p_0^{(adv)}\right)(1-p)\right) n \\
&> (1-p)n \,, \qquad (6.5)
\end{aligned}$$

where the last step follows from the bound on p in (6.3).
We now claim we can find many disjoint online-confusable pairs. For fixed q^*, r^*, b^*, let

$$C(q^*, r^*, b^*) \ := \ \{x \in C : q_x = q^*, r_{x,b^*} = r^*, b_x = b^*\}. \qquad (6.6)$$

By Lemma 6.4.2 on the set $\{x[q^*n :] : x \in C(q^*, r^*, b^*)\}$, among any $2^{\mathcal{R}n/2}$ (in fact, any $2^{\mathcal{R}(1-q^*)n/2}$) codewords in $C(q^*, r^*, b^*)$, there exists some two codewords x and y in $C(q^*, r^*, b^*)$ such that $\mathrm{LCS}(x[q^*n :], y[q^*n :]) > (1-q^*)(1-p_0^{(adv)})n$. Thus, among any $2^{\mathcal{R}n/2}$ codewords in $C(q^*, r^*, b^*)$, there are some two that are online-confusable. Thus for every q^*, r^*, b^*, we can construct disjoint pairs of codewords (x,y) that are online-confusable, so that all but $2^{\mathcal{R}n/2}$ of the elements of $C(q^*, r^*, b^*)$ are paired. For (x,y) in a pair, call x and y *partners*. Thus, all but $2n^2 \cdot 2^{\mathcal{R}n/2}$ of the elements of C have a partner. For sufficiently large n, this means that at least $0.99|C|$ elements have a partner.

The online channel choose to run one of the following strategies, each with probability $\frac{1}{2}$.

1. Strategy 1
   (a) Choose a bit b uniformly from $\{0,1\}$.
   (b) During the wait phase, delete every $x_i = 1-b$. After the wait phase, we know the codeword x and have transmitted $r_{x,b}n$ copies of the bit b.
   (c) If $x \notin C(q_x, r_{x,b}, b)$ (i.e. $x \in C(q_x, r_{x,1-b}, 1-b)$) or x has no partner codeword, transmit the remaining bits (i.e. give up).
   (d) If $x \in C(q_x, r_{x,b}, b)$ and x has a partner y, then let s^*_{xy} by the canonical common subsequence of x^*, y^* with $|s^*_{xy}| \geq (1-q_x)(1-p_0^{(adv)})n$. Since we know that the remaining bits sent by the encoder is x^*, we can delete bits so that the decoder receives s^*.
   (e) If the online channel reaches pn deletions at any point, transmit the remaining bits.
2. Strategy 2
   (a) Transmit the first $(1-p)n$ bits.
   (b) Delete the last pn bits.

The idea behind the strategy is that either the wait phase is short, in which case strategy 1 requires fewer than $n/2$ deletions, or the wait phase is long in which case strategy 2 gives less than $n/2$ deletions. In particular with probability $\frac{1}{2}$, we employ a strategy (either strategy 1 or strategy 2) that, with constant probability, needs at most $\min(q_x n, q_x n/2 + (1-q_x)p_0^{(adv)}n)$ deletions, and we check this is at most pn.

Suppose for codeword $x \in C$, the wait length is $q_x n$. Condition on the fact that x has a partner y with $q_x = q_y$ and canonical common subsequence s^*_{xy} of x^* and y^* with $|s^*_{xy}| \geq (1 - q_x)(1 - p_0^{(adv)})n$. By the above argument, this assumption holds with probability more than 0.99.

If $q_x > 1-p$, then knowing $x_1 \dots x_{(1-p)n}$ does not uniquely determine the codeword x, so there exists another codeword $y \in C$ with the same prefix $y_1 \dots y_{(1-p)n}$. Then when the encoder sends each of x and y, the online channel applying strategy 2 transmits the same string $x_1 \dots x_{(1-p)n} = y_1 \dots y_{(1-p)n}$. Since we assume the choice of codeword in C is uniformly at random, x and y are equally likely to be transmitted, so the decoder fails in this case with probability at least $\frac{1}{2}$.

If $q_x \leq 1 - p$, then with probability $\frac{1}{2}$ over the choice of b, we have $x \in C(q_x, r_{x,b}, b)$, in which case strategy 1 reaches step (d). Since $x \in C(q_x, r_{x,b}, b)$, we have $x_1 \dots x_{q_x n}$ has majority bit b, so step (b) deletes at most $q_x n/2$ bits. Since $|s^*_{xy}| \geq (1 - q_x)(1 - p_0^{(adv)})n$, step (d) deletes $|x^*| - |s^*_{xy}| \leq (1 - q_x)p_0^{(adv)} n$ bits. The total number of bits required by strategy 1 is thus

$$\frac{q_x n}{2} + (1-q_x)p_0^{(adv)} n = \left(p_0^{(adv)} + \left(\frac{1}{2} - p_0^{(adv)}\right) q_x\right) n < \left(p_0^{(adv)} + \left(\frac{1}{2} - p_0^{(adv)}\right)(1-p)\right) n < pn \tag{6.7}$$

by our choice of p. Because x and y are partners, the encoder receives $b^{q_x n} s^*_{xy}$ for both x and y being sent through the online channel. Again, as each of x and y is equally likely, the decoder fails with probability at least $\frac{1}{2}$.

We have shown that with probability at least 0.99 over the choice of codeword, either strategy 1 chooses the correct value of b with probability $\frac{1}{2}$, and thus causes decoding error with probability at least $\frac{1}{4}$, or strategy 2 causes decoding error with probability at least $\frac{1}{2}$. Adv chooses the "correct" strategy with probability $\frac{1}{2}$, so with total probably at least $0.99 \cdot \frac{1}{2} \cdot \min(\frac{1}{4}, \frac{1}{2}) > \frac{1}{10}$, there is a decoding error, as desired. □

Chapter 7

Conclusion

7.1 Open problems

These are a number of open questions concerning deletion codes. Here are some of them, along with the state of the art for each.

1. What is the zero rate threshold $p_0^{(adv)}$ for adversarial deletions?

   The best known bounds are $p_0^{(adv)} \in [\sqrt{2}-1, 1/2]$ [4].

2. What is the capacity of the binary deletion channel?

   The capacity is known to approach $1-h(p)$ as $p \to 0$ [11, 23]. For $p \to 1$, the capacity is known to be $c_0(1-p)$, where $c_0 \in [1/9, 0.4143]$ [12, 36].

3. What are efficiently decodable codes for the binary deletion channel with rate $c_0(1-p)$ for larger c_0, perhaps reaching or beating the best known existential capacity lower bound of $c_0 = 1/9$?

   Theorem 4.3.1 obtains $c_0 = 1/120$ [18].

4. What are efficient codes for the binary deletion channel with rate $1 - O(h(p))$ for $p \to 0$?

5. What is the capacity of for random channels applying *insertions and deletions*?

   In the random error model, decoding deletions, insertions, and insertions and deletions are not the same. Indeed, it is not even clear how to define random insertions. One could define insertions and deletions via the Poisson repeat channel where each bit is replaced with a Poisson many copies of itself (see [12, 35]). However, random insertions do not seem to share the similarity to random deletions that adversarial deletions share with adversarial insertions; we can decode against arbitrarily large Poisson duplication rates, whereas for codes of block length n we can decode against a maximum of n insertions or deletions [13]. Alternatively one can consider a model of random insertions and deletions where, for every bit, the bit is deleted with a fixed probability p_1, a bit is inserted after it with a fixed probability p_2, or it is transmitted unmodified with probability $1-p_1-p_2$ [41]. One could also investigate settings involving memoryless insertions, deletions, and substitutions [34].

6. The existence of codes that decode against pn oblivious deletions for any $p < 1$ is now known [17]. Can we modify the proof to show existence of codes decoding against a combination of pn oblivious *insertions and deletions* for every $p < 1$?

7. Can we find explicit codes for oblivious deletions that are constructable, encodable, and decodable in polynomial time?
8. For erasures, the capacity of the random error channel and the capacity of the oblivious error channel are both $1 - p$. For bit flips, the random and oblivious error channels also have the same capacity at $1 - h(p)$ [31]. Can we construct codes that decode oblivious deletions with rate approaching the capacity of the random deletion channel (note this capacity itself is *not* known)? On the other end, could we upper bound the best possible rate for correcting a fraction p of oblivious deletions by $o(1 - p)$ as $p \to 1$? Such an upper bound would fundamentally distinguish the behavior of the deletions from errors and erasures.
9. Can one find codes correcting p fraction of *online* deletions when p approaches $\frac{1}{2}$? By the connection to adversarial deletions established in this work, this would imply $p_0^{(adv)} = 1/2$.
10. We defined $p_0^{(on)}$ to be the zero-rate threshold for coding against online deletions with *deterministic* codes under *average-error* criterion, and proved that this threshold equals $1/2$ if and only if $p_0^{(adv)} = 1/2$. Could we establish a similar result for $p_0^{(on,s)}$ in the case of correcting online deletions with *stochastic* codes and *worst-case error* criterion (similar to the guarantee offered by Theorem 5.2.1 for oblivious deletions)?

    For online errors and erasures, the capacity of the online bit-flip and erasure channels are known exactly [5], and capacity upper bounds have guarantees against stochastic codes under both worst-case and average error criterion [2, 9].

Appendix A

Proof of Lemma 2.2.2

Proof. For each i, we can find a random variable B_i such that $B_i \geq A_i$ always, B_i takes values in $[0, \beta]$, and $\mathbf{E}[B_i] = \alpha$. Applying Lemma 2.2.1 gives

$$\begin{aligned}
\mathbf{Pr}\left[\sum_{i=1}^{n} A_i \geq n\gamma\right] &\leq \mathbf{Pr}\left[\sum_{i=1}^{n} B_i \geq n\gamma\right] \\
&\leq \mathbf{Pr}\left[\sum_{i=1}^{n} \frac{B_i}{\beta} \geq \left(1 + \left(\frac{\gamma - \alpha}{\alpha}\right)\right) \frac{n\alpha}{\beta}\right] \\
&\leq \exp\left(-\frac{\left(\frac{\gamma-\alpha}{\alpha}\right)^2 \cdot \frac{n\alpha}{\beta}}{3}\right) \\
&= \exp\left(-\frac{(\gamma - \alpha)^2 n}{3\alpha\beta}\right). \qquad \text{(A.1)}
\end{aligned}$$

□

Appendix B

Relationships between error models

In this appendix, we prove the following lemma, which shows that constructing (deterministic) codes decodable against oblivious deletions in the average case (over random messages, in the sense of Theorem 5.4.2), suffices to construct randomized codes with low decoding error probability for every message, as guaranteed by Theorem 5.2.1.

Lemma B.0.1. *Let $p \in (0, 1)$. Suppose we have a family of codes C of length n and rate $\mathcal{R}$ such that, for any deletion pattern τ with at most pn deletions, at most $\epsilon = o_n(1)$ of the codewords are decoded incorrectly. Then, for any $\delta < \frac{1}{2}$, provided n is sufficiently large, we can find a family of stochastic codes C' of length n and rate $\mathcal{R}(1-\delta) - o(1)$ that corrects pn oblivious deletions with probability $1 - 3\epsilon$.*

Proof. For set of words C^* a pair (c, τ) (where $c \in C^*$ and τ is a deletion pattern) C^*-bad if $\tau(c) \subseteq c'$ for some $c' \in C^* \setminus \{c\}$, and C^*-good otherwise.

Let $M = |C| = 2^{\mathcal{R}n}$. For any τ, let A_τ denote the set of c such that (c, τ) is C-bad. Then $|A_\tau| \leq \epsilon|C|$ by assumption that C decodes against pn deletions in the average case. Let $d = \lceil h(p)/\mathcal{R} \rceil$ be a constant over varying n, so that there are at most $M^d > \binom{n}{pn}$ choices of τ.

Let $t = M^\delta$, and $N = \lfloor M^{1-\delta}/2 \rfloor$. We construct C' iteratively. For $1 \leq k \leq N$, chose a random subset of exactly t codewords from $C \setminus \cup_{i=1}^{k-1} E_i$ to form E_k. With these sets of codewords, we associate each message m_i with a set E_i. We encode each message m_i by choosing uniformly at random one of t codewords in some set E_i.

Note that $C' \subseteq C$ and $|C'| = Nt = M/2$. It is easy to see the rate of C' is $\mathcal{R}(1-\delta) - o(1)$.

We claim that, with positive probability over the choice of C', C' corrects pn oblivious deletions with probability $1 - 3\epsilon$. Define B_τ to be all c such that (c, τ) is C'-bad. As $B_\tau \subseteq A_\tau$, we have $|B_\tau| \leq \epsilon M$. We wish to show that the probability there exists an B_τ and a message m_i with encoding set E_i such that $|B_\tau \cap E_i| \geq 3\epsilon|E_i|$ is less than 1. Fix a τ. We show that the probability $|B_\tau \cap E_i| \geq 3\epsilon|E_i|$ is tiny. When we chose E_i, we can imagine we picked the t elements of E_i one after the other. Each one was chosen from at least $M/2$ codewords, so each codeword lands in $|B_\tau \cap E_i|$ with probability at most $\epsilon M/(M/2) = 2\epsilon$. By Chernoff bounds, $\mathbf{Pr}\left[|B_\tau \cap E_i| \geq 3\epsilon|E_i|\right] \leq e^{-t/3}$. By union bound over all E_i and all B_τ, we have

$$\mathbf{Pr}\left[\exists E_i \exists \tau : |B_\tau \cap E_i| \geq 3\epsilon|E_i|\right] \;\leq\; N \cdot M^d \cdot e^{-t/3} \;<\; M^{d+1}/e^{M^\delta/3} \tag{B.1}$$

This is less than 1 for $n > \frac{2}{\delta\mathcal{R}} \log(3(d+1))$, so our code corrects pn oblivious deletions with probability $1 - 3\epsilon$, as desired. □

Appendix C

Comparison of oblivious deletions with oblivious bit flips

First, for completeness, we outline the proof that that there exist codes achieving oblivious bit-flip capacity. After this, we show that it is not possible to follow the same approach to construct oblivious deletion codes.

Theorem C.0.2 ([6, 30])**.** *The capacity of oblivious bit-flip channels is* $1 - h(p)$.

Proof that capacity $1 - h(p)$ *is achievable.* Note that $1 - h(p)$ is an upper bound for the capacity as the capacity of the binary symmetric channel is $1 - h(p)$ and the adversary can always just choose the error vector randomly.

We consider a stochastic code C where each message m is mapped uniformly into a set $E(m)$ of size t. Choose $t = n$ and $\epsilon = 1/\log n$. This code decodes successfully if, for all m,

$$\Pr_{c \in E(m)} \left[c + e \in \bigcup_{c \in C \setminus E(m)} B_{pn}(c') \right] \;\geq\; 1 - \epsilon \tag{C.1}$$

where $B_{pn}(x) \subseteq \{0,1\}^n$ is the set of all words within Hamming distance pn of x.

Choose C to have $2^{\mathcal{R}n}$ codewords at random. Fix a message m, the rest of the codeword $C \setminus E(m)$, and the error vector e. We have

$$\Pr_{c \sim \{0,1\}^n} \left[c \in \bigcup_{c \in C \setminus E(m)} B_{pn}(c' + e) \right] \;\leq\; \frac{1}{2^n} |C| \cdot |B_{pn}(0)| \;\approx\; \frac{2^{\mathcal{R}n} \cdot 2^{h(p)n}}{2^n}, \tag{C.2}$$

which is $2^{-\Omega(n)}$ if $\mathcal{R} < 1 - h(p)$. In this case

$$\begin{aligned}
&\mathop{\mathbf{Pr}}_{E(m)}\left[\mathop{\mathbf{Pr}}_{c\in E(m)}\left[c + e \notin \bigcup_{c\in C\setminus E(m)} B_{pn}(c')\right] < 1-\epsilon\right] \\
&= \mathop{\mathbf{Pr}}_{E(m)}\left[\mathop{\mathbf{Pr}}_{c\in E(m)}\left[c \notin \bigcup_{c\in C\setminus E(m)} B_{pn}(c'+e)\right] < 1-\epsilon\right] \\
&= \mathop{\mathbf{Pr}}_{E(m)}\left[\left|E(m) \cap \bigcup_{c\in C\setminus E(m)} B_{pn}(c'+e)\right| > \epsilon t\right] \\
&\le 2^{-\Omega(\epsilon n t)} = 2^{-\tilde{\Omega}(n^2)}. \qquad \text{(C.3)}
\end{aligned}$$

Thus, applying a union bound over all m and error vectors e, we get

$$\mathop{\mathbf{Pr}}_{C}\left[\exists m : \mathop{\mathbf{Pr}}_{c\in E(m)}\left[c + e \in \bigcup_{c\in C\setminus E(m)} B_{pn}(c')\right] > \epsilon\right] \le 2^{\mathcal{R}n} \cdot 2^n \cdot 2^{-\tilde{\Omega}(n^2)} \le 2^{-\tilde{\Omega}(n^2)}. \qquad \text{(C.4)}$$

Thus when $\mathcal{R} < 1 - h(p)$, we can construct a rate $\mathcal{R}$ stochastic code correcting with high probability p oblivious bit flips. □

For oblivious deletions the above technique does not work. This is established by the following lemma.

Lemma C.0.3. *Let C be a random length n rate $\mathcal{R}$ stochastic binary code such that each message is encoded uniformly at random in one of t codewords. If $p \ge 0.4$ and $0 < \epsilon < 1$, then for any message m and deletion pattern τ we have*

$$\mathop{\mathbf{Pr}}_{C}\left[\mathop{\mathbf{Pr}}_{c\in E(m)}\left[\exists c' \in C \setminus E(m) : \tau(c) \sqsubseteq c'\right] > \epsilon\right] \;>\; 2^{-h(p)n} \qquad \text{(C.5)}$$

where $D_k(s) = \{s' : |s'| = |s| - k, s' \le s\}$.

In plain English, this lemma says that for a fixed message and deletion pattern, the probability that message decodes incorrectly too many times ($> \epsilon$ fraction of the time) is too large ($> 2^{-h(p)n}$). In short, the reason the lemma is true is because the probability the entire code C contains a "really bad" word (e.g. a word with at least $0.91n$ runs of 1s) is too large. The following remark shows us, using Lemma C.0.3, why we cannot follow the same randomized approach as Theorem C.0.2.

Remark C.0.4. Following the approach of Theorem C.0.2, we would like to conclude, using union bound,

$$\begin{aligned}
&\mathop{\mathbf{Pr}}_{C}\left[\forall m \forall \tau \mathop{\mathbf{Pr}}_{c\in E(m)}\left[\exists c' \in C \setminus E(m) : \tau(c) \sqsubseteq c'\right] > \epsilon\right] \\
&\le \sum_m \sum_\tau \mathop{\mathbf{Pr}}_{C}\left[\mathop{\mathbf{Pr}}_{c\in E(m)}\left[\exists c' \in C \setminus E(m) : \tau(c) \sqsubseteq c'\right] > \epsilon\right] \\
&\le 2^{\mathcal{R}n}\binom{n}{pn} \mathop{\mathbf{Pr}}_{C}\left[\mathop{\mathbf{Pr}}_{c\in E(m)}\left[\exists c' \in C \setminus E(m) : \tau(c) \sqsubseteq c'\right] > \epsilon\right] \\
&< 2^{\mathcal{R}n} 2^{h(p)n} 2^{-h(p)n-\Omega(n)} = 2^{-\Omega(n)} \qquad \text{(C.6)}
\end{aligned}$$

Unfortunately the last inequality (indicated in red) is false by Lemma C.0.3.

Proof. It suffices to prove for $p = 0.4$. Indeed, as p increases,

$$\mathop{\mathbf{Pr}}_{c \in E(m)} \left[\tau(c) \in \bigcup_{c' \in C \setminus E(m)} D_{pn}(c') \right] \tag{C.7}$$

increases as $\tau(c)$ contains fewer symbols.

If $p = 0.4$, then $\tau(c)$ has $0.6n$ characters. Note that if c is uniform on $\{0,1\}^n$, then $\tau(c)$ is uniform on $\{0,1\}^{0.6n}$. It is easy to check that for a uniformly chosen $0.6n$ string, the probability it is a subsequence of the length $0.91n$ string $a_{0.91n} = 0101\ldots01$ is $1 - 2^{-\Omega(n)}$: the position in the longest string increases by 1.5 each time, so in expectation the string spans $0.9n$ characters of $a_{0.91n}$, so it is not a subsequence with exponentially small probability by Chernoff bounds.

There are $2^{0.09n}$ length n strings that start with $a_{0.91n}$. Call this set of strings A. Thus, the probability that C contains an element of A is at least $2^{-0.91n}$ since that is the probability the first element is an element of A. Conditioned on C containing an element of A, the probability that, for some $c \in E(m)$, $\tau(c)$ is a subsequence of some element of A is at least the probability that c is a subsequence of $a_{0.91n}$, which is $1 - 2^{-\Omega(n)}$. Thus, conditioned on C containing an element of A, we have

$$\mathbf{E}\left[\#\{c : \exists c' \in C \setminus E(m) \,\text{s.t.}\, \tau(c) \le c'\}\right] = t(1 - 2^{-\Omega(n)}) \tag{C.8}$$

Conditioned on C being fixed, consider the indicator random variables X_i for whether each $c_i \in E(m)$ satisfies $\tau(c_i)$ is a subsequence of some $c' \in C \setminus E(m)$. The $X_1, \ldots, X_t$ are i.i.d, so the probability $\sum X_i/t > \epsilon$ is approximately 1, (for sure, it is $1 - 2^{-\Omega(n)}$).

Thus we conclude

$$\mathop{\mathbf{Pr}}_{C}[\#\{c : \exists c' \in C \setminus E(m) \,\text{s.t.}\, \tau(c) \le c'\} > \epsilon t] \ge (1 - 2^{-\Omega(n)}) \mathop{\mathbf{Pr}}_{C}[C \cap A \ne \emptyset] > 2^{(0.91 - o(1))n} > 2^{-H(0.4)n} \tag{C.9}$$

as $H(0.4) \approx 0.97$. □

Intuitively C.0.3 should be true as deletion codes behave poorly when chosen completely randomly. In the adversarial deletion channel, random codes correct only $0.17n$ deletions [25], while the best known constructions correct $0.41n$ deletions [4].

Appendix D

Proofs of Lemma 5.7.19 and Lemma 5.7.20

Lemma D.0.5. *For any $j \in [1, K]$, if Z is a random variable distributed as* $\min(\text{Geometric}(j/K), \sqrt{R}-1)$*, then* $\mathbf{E}[Z] > \frac{K}{2j} - 1$.

Proof. We have

$$\begin{aligned}
\mathbf{E}[D] &= \left(\frac{j}{K}\right) \cdot 1 + \left(\frac{j}{K}\right)\left(1 - \frac{j}{K}\right) \cdot 2 + \cdots + \left(\frac{j}{K}\right)\left(1 - \frac{j}{K}\right)^{\sqrt{R}-2} \cdot (\sqrt{R} - 1) \\
&= \frac{1 - \left(1 - \frac{j}{K}\right)^{\sqrt{R}-1}}{j/K} - (\sqrt{R} - 1)\left(1 - \frac{j}{K}\right)^{\sqrt{R}-1} \\
&\geq \frac{1 - \left(1 - \frac{1}{K}\right)^{\sqrt{R}-1}}{j/K} - (\sqrt{R} - 1)\left(1 - \frac{1}{K}\right)^{\sqrt{R}-1} > \frac{K}{2j} - 1 \qquad \text{(D.1)}
\end{aligned}$$

The last inequality follows from recalling $R = 4K^4$ and twice applying

$$\left(\sqrt{R} - 1\right)\left(\frac{K-1}{K}\right)^{\sqrt{R}-1} < 2K^2 \cdot \left(\frac{K-1}{K}\right)^{K^2} < 2K^2 \cdot \left(\frac{1}{2}\right)^{K} < 1, \qquad \text{(D.2)}$$

which is true since $K > 8$. □

Lemma (Lemma 5.7.19)**.** *Let J be chosen uniformly from $[K]$. Let D be a random variable that is 1 if $J \in [\lambda - 1]$ and, conditioned on a fixed $J \geq \lambda$, is distributed as* $\min(\text{Geometric}(J/K), \sqrt{R})$. *Then* $\mathbf{E}[D] \geq (\log K)/4$.

Proof. Applying Lemma D.0.5,

$$\begin{aligned}
\mathbf{E}[D] &= \frac{1}{K}\sum_{j=1}^{K} \mathbf{E}[D | J = j] > \sum_{j=1}^{\lambda-1} \frac{1}{K} \cdot 1 + \sum_{j=\lambda}^{K} \frac{1}{K}\left(\frac{K}{2j} - 1\right) \\
&\geq -1 + \sum_{j=\lambda}^{K}\left(\frac{1}{2j}\right) > \frac{1}{2}(\ln K - \ln \lambda - 1) - 1 > \frac{1}{4}\log K. \qquad \text{(D.3)}
\end{aligned}$$

□

Lemma (Lemma 5.7.20). *Let $\lambda' \in [\lambda, K]$ and let J be chosen uniformly from $\{\lambda, \lambda{+}1, \ldots, \lambda'\}$. Let D be the random variable that, conditioned on a fixed J, is distributed as $\min(\text{Geometric}(J/K), \sqrt{R})$. Then $\mathbf{E}[D] \geq (\log K)/4$.*

Proof. Applying Lemma D.0.5,

$$\begin{aligned}\mathbf{E}[D] &= \frac{1}{\lambda' - \lambda + 1}\sum_{j=\lambda}^{\lambda'} \mathbf{E}[D|J=j] > \frac{1}{\lambda' - \lambda + 1}\sum_{j=\lambda}^{\lambda'}\left(\frac{K}{2j} - 1\right) \\ &> \frac{1}{K - \lambda + 1}\sum_{j=\lambda}^{K}\left(\frac{K}{2j} - 1\right) > -1 + \sum_{j=\lambda}^{K}\left(\frac{1}{2j}\right) > \frac{1}{4}\log K. \qquad \text{(D.4)}\end{aligned}$$

□

Bibliography

[1] Noga Alon, Jehoshua Bruck, Farzad Farnoud, and Siddharth Jain. On the duplication distance of binary strings. In *IEEE International Symposium on Information Theory, ISIT 2016, Barcelona, Spain, July 10-15, 2016*, pages 260–264, 2016. doi: 10.1109/ISIT.2016.7541301. URL http://dx.doi.org/10.1109/ISIT.2016.7541301. 1.2

[2] Raef Bassily and Adam D. Smith. Causal erasure channels. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014, Portland, Oregon, USA, January 5-7, 2014*, pages 1844–1857, 2014. doi: 10.1137/1.9781611973402.133. URL http://dx.doi.org/10.1137/1.9781611973402.133. 6.3, 6.4, 10

[3] Boris Bukh and Jie Ma. Longest common subsequences in sets of words. *SIAM J. Discrete Math.*, 28(4):2042–2049, 2014. doi: 10.1137/140975000. URL https://doi.org/10.1137/140975000. 3.2

[4] Boris Bukh, Venkatesan Guruswami, and Johan Håstad. An improved bound on the fraction of correctable deletions. *IEEE Trans. Information Theory*, 63(1):93–103, 2017. doi: 10.1109/TIT.2016.2621044. URL http://dx.doi.org/10.1109/TIT.2016.2621044. 1.4, 2.3, 3.2, 3.3, 3.3, 3.7, 3.7.1, 4.2, 4.4, 5.4, 5.4, 5.4, 5.5, 6.4, 6.4.5, 1, C

[5] Zitan Chen, Sidharth Jaggi, and Michael Langberg. A characterization of the capacity of online (causal) binary channels. In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing*, STOC '15, pages 287–296, New York, NY, USA, 2015. ACM. ISBN 978-1-4503-3536-2. doi: 10.1145/2746539.2746591. URL http://doi.acm.org/10.1145/2746539.2746591. 6.1, 10

[6] Imre Csiszár and Prakash Narayan. The capacity of the arbitrarily varying channel revisited: Positivity, constraints. *IEEE Trans. Information Theory*, 34(2):181–193, 1988. doi: 10.1109/18.2627. URL https://doi.org/10.1109/18.2627.

[7] Imre Csiszár and Prakash Narayan. Capacity and decoding rules for classes of arbitrarily varying channels. *IEEE Trans. Information Theory*, 35(4):752–769, 1989. doi: 10.1109/18.32153. URL https://doi.org/10.1109/18.32153. 5.3

[8] Daniel Cullina and Negar Kiyavash. An improvement to Levenshtein's upper bound on the cardinality of deletion correcting codes. *IEEE Trans. Information Theory*, 60(7):3862–3870, 2014. URL http://dx.doi.org/10.1109/TIT.2014.2317698. 3.2

[9] Bikash Kumar Dey, Sidharth Jaggi, Michael Langberg, and Anand D. Sarwate. Upper bounds on the capacity of binary channels with causal adversaries. *IEEE Trans. Information Theory*, 59(6):3753–3763, 2013. doi: 10.1109/TIT.2013.2245721. URL http://dx.doi.org/10.1109/TIT.2013.2245721. 10

[10] Bikash Kumar Dey, Sidharth Jaggi, Michael Langberg, and Anand D. Sarwate. A bit of delay is sufficient and stochastic encoding is necessary to overcome online adversarial erasures. In *IEEE International Symposium on Information Theory, ISIT 2016, Barcelona, Spain, July 10-15, 2016*, pages 880–884, 2016. doi: 10.1109/ISIT.2016.7541425. URL `http://dx.doi.org/10.1109/ISIT.2016.7541425`. 6.1

[11] Suhas Diggavi and Matthias Grossglauser. On transmission over deletion channels. In *Proceedings of the 39th Annual Allerton Conference on Communication, Control, and Computing*, pages 573–582, 2001. 4.2, 2

[12] Eleni Drinea and Michael Mitzenmacher. On lower bounds for the capacity of deletion channels. *IEEE Trans. Information Theory*, 52(10):4648–4657, 2006. doi: 10.1109/TIT.2006.881832. URL `https://doi.org/10.1109/TIT.2006.881832`. 1.3.2, 1.4, 4.2, 4.3, 4.4, 4.5, 2, 5

[13] Eleni Drinea and Michael Mitzenmacher. Improved lower bounds for the capacity of i.i.d. deletion and duplication channels. *IEEE Trans. Information Theory*, 53(8):2693–2714, 2007. doi: 10.1109/TIT.2007.901221. URL `https://doi.org/10.1109/TIT.2007.901221`. 4.2, 4.3, 4.5, 5

[14] Peter Elias. Coding for two noisy channels. *Information Theory: Third London Symposium*, pages 61–74, 1956. 1.4

[15] Dario Fertonani, Tolga M. Duman, and M. Fatih Erden. Bounds on the capacity of channels with insertions, deletions and substitutions. *IEEE Trans. Communications*, 59(1):2–6, 2011. doi: 10.1109/TCOMM.2010.110310.090039. URL `http://dx.doi.org/10.1109/TCOMM.2010.110310.090039`. 4.2

[16] Venkatesan Guruswami and Ray Li. Efficiently decodable insertion/deletion codes for high-noise and high-rate regimes. In *IEEE International Symposium on Information Theory, ISIT 2016, Barcelona, Spain, July 10-15, 2016*, pages 620–624, 2016. doi: 10.1109/ISIT.2016.7541373. URL `http://dx.doi.org/10.1109/ISIT.2016.7541373`. 3.1, 4.4

[17] Venkatesan Guruswami and Ray Li. Coding against deletions in oblivious and online models. abs/1612.06335, 2017. URL `http://arxiv.org/abs/1612.06335`. Manuscript. 1.3.2, 5.1, 6.1, 6

[18] Venkatesan Guruswami and Ray Li. Efficiently decodable codes for the binary deletion channel. abs/1705.01963, 2017. URL `http://arxiv.org/abs/1705.01963`. Manuscript. 4.1, 3

[19] Venkatesan Guruswami and Adam D. Smith. Optimal rate code constructions for computationally simple channels. *J. ACM*, 63(4):35:1–35:37, 2016. doi: 10.1145/2936015. URL `http://doi.acm.org/10.1145/2936015`. 5.3

[20] Venkatesan Guruswami and Carol Wang. Deletion codes in the high-noise and high-rate regimes. *IEEE Trans. Information Theory*, 63(4):1961–1970, 2017. doi: 10.1109/TIT.2017.2659765. URL `http://dx.doi.org/10.1109/TIT.2017.2659765`. 2.3, 3.2, 3.3, 3.3, 3.3, 3.4.1, 3.6.2, 4.2, 4.4, 4.4.1

[21] Bernhard Haeupler and Amirbehshad Shahrasbi. Synchronization strings: codes for insertions and deletions approaching the singleton bound. In *Proceedings of the 49th An-*

nual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, June 19-23, 2017, pages 33–46, 2017. doi: 10.1145/3055399.3055498. URL `http://doi.acm.org/10.1145/3055399.3055498`. 3.2, 4.3, 4.4.4, 4.5

[22] Richard W Hamming. Error detecting and error correcting codes. *Bell System technical journal*, 29(2):147–160, 1950. 1

[23] Adam Kalai, Michael Mitzenmacher, and Madhu Sudan. Tight asymptotic bounds for the deletion channel with small deletion probabilities. In *IEEE International Symposium on Information Theory, ISIT 2010, June 13-18, 2010, Austin, Texas, USA, Proceedings*, pages 997–1001, 2010. doi: 10.1109/ISIT.2010.5513746. URL `http://dx.doi.org/10.1109/ISIT.2010.5513746`. 4.2, 2

[24] Yashodhan Kanoria and Andrea Montanari. Optimal coding for the binary deletion channel with small deletion probability. *IEEE Trans. Information Theory*, 59(10):6192–6219, 2013. doi: 10.1109/TIT.2013.2262020. URL `http://dx.doi.org/10.1109/TIT.2013.2262020`. 4.2

[25] Ian A. Kash, Michael Mitzenmacher, Justin Thaler, and Jonathan Ullman. On the zero-error capacity threshold for deletion channels. In *Information Theory and Applications Workshop, ITA 2011, San Diego, California, USA, February 6-11, 2011*, pages 285–289, 2011. doi: 10.1109/ITA.2011.5743594. URL `https://doi.org/10.1109/ITA.2011.5743594`. 3.2, 4.2, 4.4, C

[26] Adam Kirsch and Eleni Drinea. Directly lower bounding the information capacity for channels with i.i.d.deletions and duplications. *IEEE Trans. Information Theory*, 56(1):86–102, 2010. doi: 10.1109/TIT.2009.2034883. URL `http://dx.doi.org/10.1109/TIT.2009.2034883`. 4.2

[27] Mladen Kovacevic and Petar Popovski. Zero-error capacity of a class of timing channels. *IEEE Trans. Information Theory*, 60(11):6796–6800, 2014. doi: 10.1109/TIT.2014.2352613. URL `http://dx.doi.org/10.1109/TIT.2014.2352613`. 1.2

[28] Ankur A. Kulkarni and Negar Kiyavash. Nonasymptotic upper bounds for deletion correcting codes. *IEEE Trans. Information Theory*, 59(8):5115–5130, 2013. URL `http://dx.doi.org/10.1109/TIT.2013.2257917`. 3.2

[29] Eric S Lander, Lauren M Linton, Bruce Birren, Chad Nusbaum, Michael C Zody, Jennifer Baldwin, Keri Devon, Ken Dewar, Michael Doyle, William FitzHugh, et al. Initial sequencing and analysis of the human genome. *Nature*, 409(6822):860–921, 2001. 1.2

[30] Michael Langberg. Oblivious communication channels and their capacity. *IEEE Trans. Information Theory*, 54(1):424–429, 2008. doi: 10.1109/TIT.2007.911217. URL `http://dx.doi.org/10.1109/TIT.2007.911217`. 5.3, C.0.2

[31] Amos Lapidoth and P. Narayan. Reliable communication under channel uncertainty. *IEEE Trans. Information Theory*, 44(6):2148–2177, 1998. doi: 10.1109/18.720535. URL `http://dx.doi.org/10.1109/18.720535`. 5.3, 8

[32] Vladimir I. Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. *Dokl. Akad. Nauk*, 163(4):845–848, 1965 (Russian). English translation in Soviet Physics Doklady, 10(8):707-710, 1966. 1.2, 2.3, 3.2

[33] Vladimir I. Levenshtein. Bounds for deletion/insertion correcting codes. In *Proceedings of the IEEE International Symposium on Information Theory*, 2002. 3.2

[34] Hugues Mercier, Vahid Tarokh, and Fabrice Labeau. Bounds on the capacity of discrete memoryless channels corrupted by synchronization and substitution errors. *IEEE Trans. Information Theory*, 58(7):4306–4330, 2012. doi: 10.1109/TIT.2012.2191682. URL http://dx.doi.org/10.1109/TIT.2012.2191682. 5

[35] Michael Mitzenmacher. A survey of results for deletion channels and related synchronization channels. *Probability Surveys*, 6:1–33, 2009. doi: 10.1214/08-PS141. URL http://dx.doi.org/10.1214/08-PS141. 1.2, 1.3.2, 4.1, 5

[36] Mojtaba Rahmati and Tolga M. Duman. Upper bounds on the capacity of deletion channels using channel fragmentation. *IEEE Trans. Information Theory*, 61(1):146–156, 2015. doi: 10.1109/TIT.2014.2368553. URL http://dx.doi.org/10.1109/TIT.2014.2368553. 1.4, 4.2, 2

[37] Ron M. Roth and Gitit Ruckenstein. Efficient decoding of reed-solomon codes beyond half the minimum distance. *IEEE Trans. Information Theory*, 46(1):246–257, 2000. doi: 10.1109/18.817522. URL https://doi.org/10.1109/18.817522. 3.5

[38] Leonard J. Schulman and David Zuckerman. Asymptotically good codes correcting insertions, deletions, and transpositions. *IEEE Trans. Information Theory*, 45(7):2552–2557, 1999. doi: 10.1109/18.796406. URL https://doi.org/10.1109/18.796406. 2.3, 3.2

[39] Claude Shannon. A mathematical theory of communication. *Bell Systems Technical Journal*, 27(3):379423, 1948. 1, 1.2, 1.4

[40] Madhu Sudan. Decoding of reed solomon codes beyond the error-correction bound. *J. Complexity*, 13(1):180–193, 1997. URL http://people.csail.mit.edu/madhu/papers/1996/reeds-journ.pdf. 3.5

[41] Ramji Venkataramanan, Sekhar Tatikonda, and Kannan Ramchandran. Achievable rates for channels with deletions and insertions. *IEEE Trans. Information Theory*, 59(11):6990–7013, 2013. doi: 10.1109/TIT.2013.2278181. URL http://dx.doi.org/10.1109/TIT.2013.2278181. 5

[42] S. M. Sadegh Tabatabaei Yazdi and Lara Dolecek. A deterministic polynomial-time protocol for synchronizing from deletions. *IEEE Trans. Information Theory*, 60(1):397–409, 2014. doi: 10.1109/TIT.2013.2279674. URL http://dx.doi.org/10.1109/TIT.2013.2279674. 4.2