NEW COMBINATORIAL BOUNDS FOR ERROR CORRECTING CODES

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF COMPUTER SCIENCE
AND THE COMMITTEE ON GRADUATE STUDIES
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Ray Li
August 2022

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

(Jacob Fox) Principal Co-Adviser

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

(Mary Wootters) Principal Co-Adviser

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

(Li-Yang Tan)

Approved for the Stanford University Committee on Graduate Studies

Acknowledgments

This thesis would not have been possible without the help of many individuals, and I am very grateful to all who have been a part of my PhD.

First, I thank my amazing advisors, Mary and Jacob. I have been incredibly fortunate to be co-advised by them. Their generous support, guidance, and encouragement have been invaluable. They have helped in many ways, finding exciting research questions to work on, providing opportunities to speak at seminars and attend conferences, organizing social events to connect with other students, allowing me to mentor undergraduate students over the summers, providing resources to be productive during the pandemic, and giving invaluable advice on research, writing, speaking, and collaborating, and the list goes on. Thank you both very much for all of your support.

I thank the rest of my thesis committee, Tsachy Weissman, Li-Yang Tan, and Nima Anari, for their time and energy listening to my defense and reading this thesis. I thank Aviad Rubinstein for serving on my quals committee and giving valuable advice and encouragement.

I thank Venkat Guruswami for investing in me when I first started doing research at CMU and for introducing me to coding theory.

A highlight of my PhD has been working with and learning from many talented and wonderful people, and I thank all of my coauthors and collaborators for making my PhD a rich experience. In alphabetical order, my coauthors are: Alexandra Kolla, Amy Kanne, Benny Sudakov, Bruce Spang, Charles Carlson, Chong Shangguan, Francisco Pernice, Itzhak Tamo, Jabari Hastings, Jacob Fox, Jonathan Mosheiff, Joshua Brakensiek, Luca Trevisan, Mary Wootters, Mina Dalirrooyfard, Nicolas Resch, Nitya Mani, Percy Liang, Shashwat Silas, Stephen Mussmann, Venkatesan Guruswami, Virginia Vassilevska-Williams, Xiaoyu He, and Zeyu Guo.

I thank the members of Stanford's Theory Group, Mary's group, and Jacob's group, for their encouragement, teaching, advice, and friendship. I thank Stanford Computer Science Department for hosting me, and the National Science Foundation for the NSF Graduate Research Fellowship.

Many supported me personally throughout my PhD. I thank all of my friends in the Bay Area for making the last five years a special season. I thank my family, Dad, Mom, and Jason, for their constant love and support. Lastly, I thank my wife Carolyn for loving me, encouraging me, and bringing me so much joy.

Contents

Acknowledgments **iv**

1 Introduction **1**
1.1 Overview of contributions . . . 2
1.1.1 Deletion errors . . . 3
1.1.2 List-decoding . . . 4
1.2 Dissertation outline . . . 6

2 Setup and Preliminaries **7**
2.1 Notation . . . 7
2.2 Coding theory: basic notation . . . 7
2.3 The Hamming model: Worst-case substitutions . . . 8
2.4 Worst-case deletions . . . 12
2.5 List-decoding . . . 12

3 Deletion codes **16**
3.1 Introduction . . . 16
3.1.1 Our results . . . 17
3.1.2 Related works . . . 18
3.1.3 Deletion correction in related models . . . 19
3.2 Proof overview . . . 20
3.3 Preliminaries and Notation . . . 23
3.4 The structure lemma and definition of types . . . 27
3.4.1 The structure lemma . . . 27
3.4.2 Definition of types . . . 31
3.5 The entropy regularity argument . . . 32
3.5.1 Flag balance . . . 32
3.5.2 Flag balance of intervals . . . 32
3.5.3 Flag balance of substrings . . . 34
3.6 Green case . . . 35
3.7 Blue-Yellow case . . . 38
3.8 Putting it all together . . . 44
3.8.1 Statistics . . . 44

3.8.2 The Imbalanced case . . . 45
3.8.3 Combining the arguments for the Imbalanced, Green, and Blue-Yellow cases . . . 45
3.8.4 Finishing the proof . . . 48

4 List decoding: binary alphabet 50
4.1 Introduction . . . 50
4.2 Previous Work and Our Results . . . 51
4.2.1 Random Linear Codes . . . 51
4.2.2 Our list-size upper bounds for random linear codes . . . 52
4.2.3 Our lower bounds: Pinning down the output list size. . . . 53
4.2.4 Generalizations and variations . . . 53
4.2.5 Related work . . . 54
4.3 Preliminaries . . . 55
4.4 Upper bound proof . . . 56
4.4.1 The approach of [49] . . . 57
4.4.2 Proof of Theorem 4.4 . . . 58
4.5 Lower bound proof . . . 61
4.5.1 Techniques. . . . 61
4.5.2 Tools from [101] . . . 63
4.5.3 List decoding RLC lower bound . . . 64

5 List decoding: large alphabet 70
5.1 Introduction . . . 70
5.1.1 Contributions . . . 72
5.1.2 Related Work . . . 74
5.1.3 Technical Overview . . . 75
5.2 Preliminaries . . . 79
5.2.1 Cycle Spaces . . . 79
5.2.2 t-Wise Intersection Matrices . . . 81
5.3 Connection to Tree Packing and an Intermediate Result . . . 82
5.4 Near-Optimal List-Recovery of RS Codes: Proof of the Main Theorem . . . 88
5.4.1 Overview of the Proof . . . 88
5.4.2 A Combinatorial Lemma . . . 89
5.4.3 Proof of Theorem 5.15 . . . 90
5.4.4 Proof of Lemma 5.17 . . . 92
5.5 Towards Conjecture 5.3: A Hypergraph Nash-Williams–Tutte Conjecture . . . 96
5.5.1 A Hypergraph Nash-Williams–Tutte Conjecture . . . 97
5.5.2 Proof of Theorem 5.26 . . . 100

6 Summary and Conclusions 105
6.1 Summary of contributions . . . 105
6.2 Future work . . . 105
6.2.1 Deletion codes . . . 105

6.2.2 List-decoding small alphabet codes 107
6.2.3 List-decoding Reed–Solomon codes 108

Bibliography **109**

Chapter 1

Introduction

This thesis is about coding theory. Coding theory studies *(error-correcting) codes*, sets of strings that protect information from noise. A natural illustration of a code is the set of valid English sentences: "yOu alr*dy kn*w what an err*r coarecting code is." is a corrupted English sentence whose meaning you can decode, despite the errors.

While the English language is a code useful to people, coding theory studies codes useful to computers, specifically codes with mathematical guarantees. To formalize these guarantees, we typically imagine a communication setup (see Figure 1.1): a sender, conventionally named Alice, wants to send a message to a receiver, conventionally named Bob, through a noisy channel, so she sends Bob an encoded message called a *codeword* (e.g., an English sentence) with enough redundancy that Bob can decode the message, even in the presence of noise. The success of this protocol largely boils down to the mathematical properties of the *code*, the set of possible codewords Alice could send (e.g., the set of all English sentences).

The goal of coding theory is to help Alice and Bob by finding good codes. A code is "good" if it is less redundant, meaning Alice's codeword is not too much longer than the message, and more robust, meaning the protocol can tolerate more noise. We formally quantify these notions later, but for now we note that there are many ways to measure robustness, depending on the meanings of "tolerate" and "noise." For instance, the "noise" could be *substitution* errors, the error in Figure 1.1, which changes individual symbols of a string, or *deletion* errors, which drop symbols in strings (so that "cat" may become "at"). Additionally, "tolerate" could mean the natural notion of *unique-decoding* or a more relaxed (but also central) notion called *list-decoding*, both to be defined later. In these various settings, this central challenge—finding codes that are both less redundant and more robust—has several facets.

Question 1.1. What is the mathematically optimal tradeoff between redundancy and robustness?

Question 1.2. Can we find explicit codes achieving the optimal redundancy versus robustness tradeoff?

Question 1.3. For such explicit codes, are there fast encoding and decoding algorithms? That is, can Alice encode her message into a codeword quickly, and can Bob decode the message from the noisy codeword quickly?

Coding theory is a broad field spanning electrical engineering, computer science, and mathematics. Because of the pervasive need to protect data from noise, codes find many applications in electrical engineering such as communication and storage. For example, "Alice" could be your phone sending a message to "Bob",

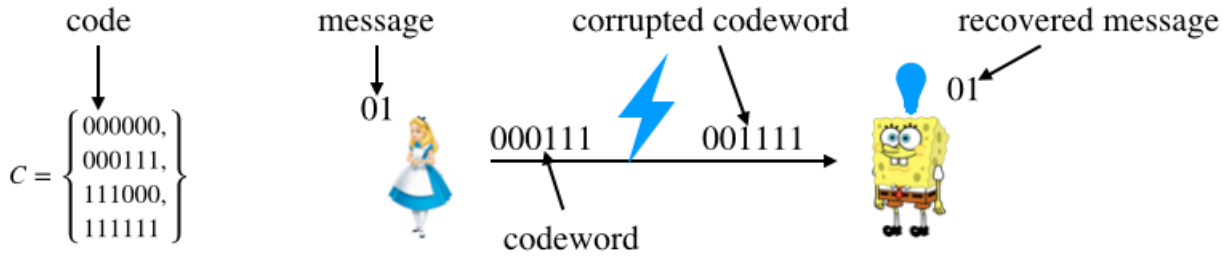


Figure 1.1: Error correcting codes: Alice want to send a message to Bob through a noisy channel. She encodes the message as a codeword of an (error-correcting) code, so that Bob can recover the message from a noisy copy of the codeword. Here, Alice uses a very simple code, encoding her message by repeating every symbol 3 times. The message has length $k = 2$, the codeword has length $n = 6$, so the rate is $R = 1/3$.

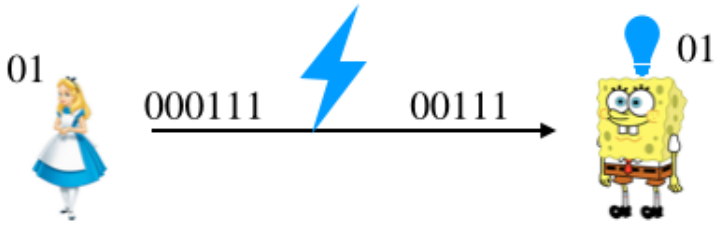
a cell tower, or "Alice" could be a satellite sending a message to "Bob", a satellite dish. For similar reasons, and because codes have such a fundamental mathematical definition, codes find applications in computer science such as pseudorandomness, computational complexity, and cryptography. For example, *list-decodable codes*, which we study in this thesis, are intimately connected to central objects in pseudorandomness. Because we use a variety of mathematical techniques to construct and analyze codes, coding theory has rich connections to diverse areas of mathematics. For example, in this thesis, we make a novel application of *regularity*, a powerful tool in additive combinatorics and extremal combinatorics, to *deletion codes*.

Despite tremendous progress since its inception over 70 years ago, coding theory has basic mathematical challenges that remain open. This thesis makes progress on several such challenges in two classic contexts: *deletion errors* and *list-decoding*. Each setting is a simple variation of the most traditional error-correction model: In deletion codes, we consider *deletion* errors rather than the more traditionally studied substitution errors. In *list-decoding*, we keep the standard error model of substitutions, but Bob only needs to output a small list of message containing the correct one.

1.1 Overview of contributions

As mentioned above, the central question in coding theory is finding codes $\mathcal{C}$ (which, recall, are simply sets of strings) with the optimal tradeoff between redundancy and robustness, for various definitions of redundancy and robustness. Non-redundancy is typically quantified by the *rate* $R = k/n$, the ratio between Alice's *message length* k and the *codeword length* n (recall a codeword is simply an element of the code). Note that, (i) unlike our illustration of English sentences, we require all codewords in a code to be the same length and (ii) unlike in English where the "message" is a nebulous idea, here the message is a concrete length k string over the same alphabet as the codeword (see Figure 1.1 for an example). We also typically think of the message and the codeword as strings over the same *alphabet* (in Figure 1.1, the alphabet is $\{0, 1\}$). We want larger rate, so that there is a smaller number $n - k$ of "redundant" symbols that Alice sends in addition to the message.

Robustness is typically is quantified by a parameter p, which roughly represents the fraction of symbols that the channel can corrupt. The exact definition depends on the error model, which depends on questions such as: Are the errors applied by an adversary, or randomly, or some other way? Are the errors substitutions, erasures, deletions, or something else? Does Bob need to recover the message exactly (unique-decoding), or



No error	000111
Substitution	001111
Erasure	00?111
Deletion	**00111**

Figure 1.2: Deletion errors: The noisy channel applies *deletions*, which drops symbols from unknown positions so that Bob receives a subsequence of Alice's codeword. In contrast to the closely related *erasure* errors, the position of deleted bits is unknown. The table on the right compares substitutions, erasures, and deletions. While substitutions and erasures are well understood, deletions are still poorly understood.

is it enough to return a list of messages containing the correct one (list-decoding)?

Given these parameters R and p, which we both want to be large, the natural and central questions are:

- What is the optimal tradeoff between R and p? (Question 1.1)
- Can we find explicit codes C (Question 1.2) with fast encoding/decoding algorithms (Question 1.3) that achieve the optimal R versus p tradeoff?

This thesis considers these questions in two classic settings: deletion errors and list-decoding.

1.1.1 Deletion errors

Codes correcting *deletions* tolerate a different type of noise than in classical settings. Deletions drop information from unknown positions so that Bob receives a subsequence of Alice's codeword; if Alice sends "cat", Bob may receive "at". By contrast, classical noise models like erasures and substitutions preserve the positions of uncorrupted symbols; if Alice sends "cat", Bob may receive "hat" or "?at". Despite their superficial similarity to erasures and substitutions, deletions are far less understood.

Codes correcting *deletion errors* are of practical and theoretical interest. Practically, deletion codes find applications in DNA storage, magnetic recording, and internet communication. Theoretically, this area asks basic combinatorics questions about subsequences: Levenshtein [91], who initiated the study of deletion codes, showed that a code C corrects p fraction of worst-case deletions if and only if the longest common subsequence of every pair of strings is less than $n - pn$.

Adversarial bit deletions. Our understanding of deletion codes is poor and in some cases embarrassing, as the following example shows. Suppose Alice is constrained to sending bits (0s and 1s), so that the codewords are binary strings. If the channel gets a budget of $n/2$ deletions, the channel, in the worst-case, could delete all the zeros or all the ones of Alice's codeword, so that Bob gets either $n/2$ 0s or $n/2$ 1s, and thus basically no information about Alice's message. This simple argument shows it is impossible for binary codes to correct $p = 1/2$ fraction of worst-case deletions. The natural question is then, what about p slightly less than $1/2$? Perhaps, for any $\delta > 0$, there could be nontrivial codes that handle $p = 1/2 - \delta$ fraction of worst-case deletions, but embarrassingly we had not been able to confirm or rule out this possibility.

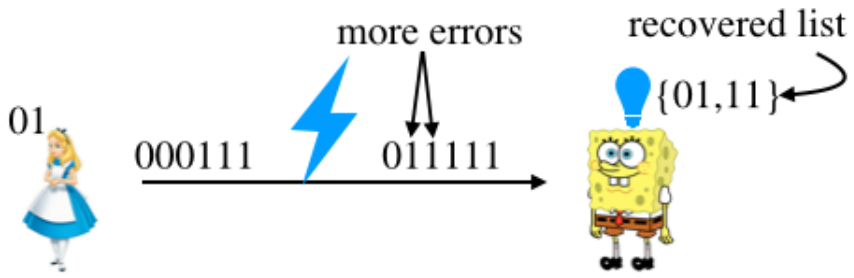


Figure 1.3: List-decoding: In list-decoding, Bob's task is easier. He only needs to output a small list of messages that contains the correct one. In exchange for this relaxed guarantee, Alice and Bob can tolerate more error (approximately twice as much).

The above question is even more basic than determining the optimal rate versus deletion fraction tradeoff in Question 1.1. The above asks for a bound on the *zero-rate threshold*, $p_{\text{del}}^{\text{thr}}$ namely the largest (supremum) deletion fraction p tolerable by a code of non-vanishing rate. While the rate R versus deletion fraction p tradeoff question asks for the curve of R versus p values achievable by a worst-case deletion code, the zero-rate threshold asks simply for the "p-intercept" of this curve. For the traditional substitutions and erasures, the zero-rate threshold has been known since 1960 [104]. By contrast the zero rate threshold for worst-case deletions is still open, despite that the more general rate versus deletion fraction question is over 50 years old [122].

The main contribution of this part of the thesis is to improve the trivial limitation of $1/2$, showing that the zero rate threshold for worst-case deletions is at most $1/2 - \delta$ for an absolute constant $\delta > 0$. The proof turns out to be a challenging combinatorial argument on finding common subsequences, and along the way we develop several techniques for reasoning about common subsequences. As common subsequences appear throughout computer science, we hope that the techniques will be of independent interest and find other applications.

1.1.2 List-decoding

In list-decoding [34, 127], Bob only needs to output a small list of messages containing the correct message. In general, this relaxation allows the protocol to tolerate more noise (approximately twice as much). For this reason, list-decoding finds various applications such as communications, group testing, compressed sensing. List-decodable codes also have a fundamental mathematical definition, allowing them to find "extraneous" applications that have no obvious need for error correction, such as pseudorandomness, complexity, and cryptography.

In list-decoding, the goal is still to find codes with large rate R and that tolerate a large number of substitution errors p, while keeping the list size L small (often a constant, or at most polynomial in the codeword length n). In contrast to deletions, we have long known the optimal R versus p tradeoff and thus the answer to Question 1.1 for list-decoding. We call this optimal R vs. p tradeoff the *(list-decoding) capacity*, and we say codes approaching this optimal tradeoff *achieve (list-decoding) capacity*. However, in the standard proof of this tradeoff, the capacity achieving codes are obtained non-explicitly using the probabilistic method. As most applications require explicit list-decodable codes, the major challenge in

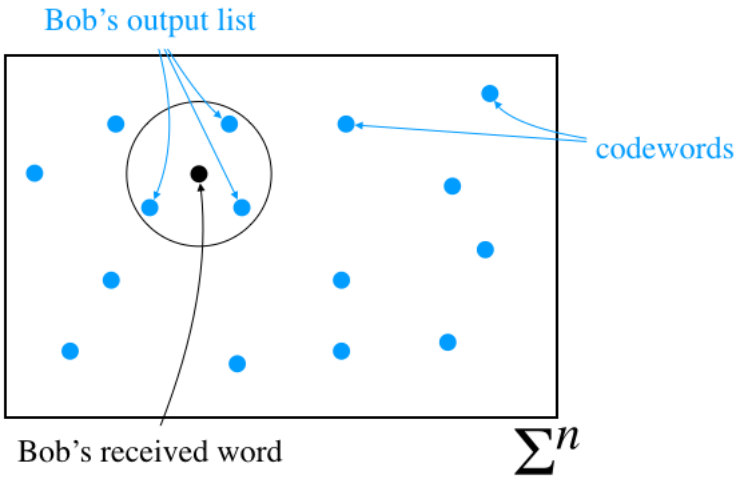


Figure 1.4: Geometric interpretation of list-decoding: List decoding asks to find sets that are not too "clustery," meaning no "neighborhood" contains too many codewords. Our work on list-decoding random linear codes asks how clustery are random subspaces over finite fields and compares the clustery-ness of random subspaces versus random sets.

list-decoding is finding *explicit* codes achieving list-decoding capacity (Question 1.2), and finding fast list-decoding algorithms (Question 1.3) for such codes.

Towards the goal of explicit list-decodable codes, we prove stronger list-decoding guarantees for more structured ensembles of codes such as random linear codes and Reed–Solomon codes. Our study of random linear codes is relevant for the setting where Alice is sending Bob bits ("binary codes"), and our study of list-decoding Reed–Solomon codes is relevant for the setting where Alice sends Bob symbols over a larger alphabet.

These results are fundamental mathematical results as well. Our study of random linear codes answers a geometric question about how "clustery" are random subspaces over finite fields (see Figure 1.4). Reed–Solomon codes are a classic and ubiquitous family of codes, making their list-decodability a tantalizing question. In analyzing Reed–Solomon codes, we discover surprising connection between list-decoding Reed–Solomon codes and the Nash-Williams–Tutte tree packing theorem.

Binary alphabets. For binary codes, uniformly random codes achieve list-decoding capacity, but we do not have any explicit codes achieving list-decodable capacity. Towards explicit list-decodable codes, we study random linear codes, which are more structured (less random) ensembles of codes than uniformly random codes.

We give an extremely tight analysis of the list-decodability of random binary linear codes, improving both the known upper and lower bounds. Previous positive results (upper bounds) on the list-decodability of random linear binary codes either hold only in certain (non-overlapping) parameter regimes, or else get substantially sub-optimal bounds on Bob's list-size. Our argument obtains improved list size bounds over all these results and works in all parameter regimes. Furthermore, we present matching lower bounds, showing that our upper-bound analysis is tight in a very strong way: we show a lower bound on Bob's list-size of random linear binary codes that matches the upper bound up to a small additive factor, showing that the

"correct" list-size is concentrated on at most 3 values.

Large alphabets. For codes with no restriction on the alphabet size, we do have explicit codes achieving capacity, the optimal R versus p tradeoff. These codes are all generalizations of the classic and ubiquitous Reed–Solomon codes. However, despite this progress, we still do not know whether Reed–Solomon codes themselves achieve capacity. In addition to being a natural question in its own right, this question is also interesting because, if Reed–Solomon codes achieve list-decoding capacity, they would offer advantages over existing explicit list-decodable codes, such as simplicity and potentially smaller alphabet sizes.

This thesis gives list-decoding bounds for Reed–Solomon codes. We give the first proof that, for small constant rates, Reed–Solomon codes can be list-decoded beyond the Johnson bound, and in fact nearly to *capacity*. We also show a surprising connection between list-decoding Reed–Solomon codes and the Nash-Williams–Tutte tree-packing theorem, and prove that a generalization of the tree-packing theorem to hypergraphs implies that Reed–Solomon codes are list-decodable to capacity (non-asymptotically). Our techniques also apply to *list-recovery*, a generalization of list-decoding.

1.2 Dissertation outline

In Chapter 3 we prove our limitation result for deletion codes, showing that the zero rate threshold for adversarial bit deletions is less than $1/2$. The results of Chapter 3 is based on the paper [50], which is joint work with Venkatesan Guruswami and Xiaoyu He. In Chapter 4, we discuss our results for list-decoding random linear binary codes. We present tight upper and lower bounds on list-size of random linear binary codes. The upper bound is based on the paper [93], joint with Mary Wootters, and the lower bound is based on the paper [59], joint with Venkatesan Guruswami, Jonathan Mosheiff, Nicolas Resch, Shashwat Silas, and Mary Wootters. In Chapter 5, we discuss our results for list-decoding Reed–Solomon codes. We present our main result that there exist Reed–Solomon codes list-decodable almost to capacity for small constant rate, and present the connection to the Nash-Williams–Tutte tree packing theorem. The results of Chapter 5 are based on the paper [43], joint with Zeyu Guo, Chong Shangguan, Itzhak Tamo, and Mary Wootters. In Chapter 6, we conclude with some open questions.

Chapter 2

Setup and Preliminaries

2.1 Notation

Unless otherwise specified, all logarithms are base 2. We let $\exp(x)$ denote 2^x. Let $\mathbb{N}^+ = \{1, 2, \dots\}$ and $[n] = \{1, 2, \dots, n\}$ for $n \in \mathbb{N}^+$. For a prime power q, let $\mathbb{F}_q$ denote the field field with q elements. We use standard Landau notation $O(\cdot), \Omega(\cdot), \Theta(\cdot), o(\cdot), \omega(\cdot)$. We use the notation $O_x(\cdot), \Omega_x(\cdot)$ to mean that dependencies on the variable(s) x are suppressed.

2.2 Coding theory: basic notation

An *(error-correcting) code* $\mathcal{C} \subset \Sigma^n$ is a set of strings. Elements of a code are called *codewords*. We call Σ the *alphabet* of the code and n the *(block) length* of the code. We call elements in the alphabet "symbols", so that codewords consist of n symbols. Sometimes, as in Chapter 3, we denote the length with a capital N for notational convenience. If $|\Sigma| = 2$, we say the code is a *binary* code. Sometimes, as in Chapter 4, we think of the alphabet of a binary code as $\Sigma = \mathbb{F}_2$. The *dimension* of the code is defined as $k = \log_{|\Sigma|} |\mathcal{C}|$, and the *rate* $R = R(\mathcal{C})$ is the ratio $R = k/n$. If $\Sigma = \mathbb{F}_q$ is a finite field and $\mathcal{C}$ is a subspace of $\mathbb{F}_q^n$, we say $\mathcal{C}$ is a *linear* code. Note that for linear codes, the dimension of the code equals the dimension of the subspace.

We are interested in asymptotic tradeoffs for codes, so throughout we imagine the length n of the code growing while other parameters remain fixed. Formally, a *family of codes* $\mathcal{C}_1, \mathcal{C}_2, \dots$ is a sequence of codes of length $n_1 < n_2 < \cdots$. We say that the family has rate R if $\lim_{i \to \infty} R(\mathcal{C}_i) = R$. Similarly, for other code parameters, such as the noise parameter p, we say the family of codes has noise parameter p if the individual codes have noise parameter approaching p. For the rest of the thesis, by extensive abuse of notation, when we speak of codes with rate R that tolerate noise parameter p, we mean a family of codes with rate approaching R and noise parameter approaching p.

This thesis focuses on *combinatorial* bounds for error-correcting codes, which means we consider what codes allow Alice and Bob to succeed, if we ignore computational constraints. In practice, we are also interested in *explicit* codes with efficient *encoding* and *decoding*. We reference these terms in the thesis so we make them precise here for clarity. A code has *efficient encoding* if Alice can do her part of the protocol efficiently: formally, there exists a bijection $\text{Enc} : \{1, \dots, |\mathcal{C}|\} \to \mathcal{C}$ computable by a deterministic polynomial time algorithm (here, polynomial means polynomial in the length n). When an encoding function

is considered, we think of the elements of $\{1, \ldots, |\mathcal{C}|\}$ as the *messages*. We sometimes equivalently think about the message set as Σ^k if the dimension $k = \log_{|\Sigma|} |\mathcal{C}|$ is an integer. A code has *efficient decoding* if Bob can do his part of the protocol efficiently: formally, there exists a function Dec that maps noisy received words to the original codeword or message (or a list of codewords/messages, in the case of list-decoding). A code is *explicit* if we can produce a description of the encoding function Enc and decoding function Dec in deterministic polynomial time.

The *Hamming distance* is a useful notion for reasoning about substitution errors. For two strings x, y of the same length, let $\Delta_H(x, y) = |\{i : x_i \neq y_i\}|$ denote the *Hamming distance* of x and y. If the alphabet is a finite field, the *Hamming weight* of a string x is the number of nonzero entries and is equal to $\Delta_H(x, 0)$ where 0 here is the all-zeros vector of length n. For a string $z \in \Sigma^n$, we let $\mathcal{B}_\Sigma(z, pn)$ denote the *Hamming ball* around z of radius pn, i.e., the set of all strings at Hamming distance at most pn from z. Typically, the alphabet Σ is understood and omitted. Clearly Hamming balls $\mathcal{B}(z, pn)$ have the same size for all z, so we let $\mathrm{Vol}(n, pn) \stackrel{\text{def}}{=} |\mathcal{B}(0, pn)|$ denote the volume of a Hamming ball of radius pn in $\{0, 1\}^n$. For binary codes, we have the following useful estimate of $\mathrm{Vol}(n, pn)$ (see [97]):

$$2^{(h(p)-o(1))n} \leq \mathrm{Vol}(n, pn) \leq 2^{h(p)n}. \tag{2.1}$$

Above, $h : [0, 1] \to [0, 1]$ is the ubiquitous *binary entropy function*, defined as

$$h(p) \stackrel{\text{def}}{=} -p \log p - (1 - p) \log(1 - p).$$

2.3 The Hamming model: Worst-case substitutions

Coding theory began with Shannon's seminal work [114], which considered codes against *random* errors, particularly random erasures and substitutions. Shortly after, Hamming [77] considered codes against *worst-case* errors, also called *adversarial* errors. These are the two most traditionally studied models of errors. The settings considered in this thesis are variations on the Hamming model. As an introduction to the models in this thesis, and as a way to introduce some of the codes and techniques we work with, we start with a brief overview of the Hamming model.

The Hamming model considers codes against worst case errors, particularly worst case substitutions. For Alice and Bob, the channel is modeled by an adversary with a budget of pn substitution errors, and is allowed to change up to pn of Alice's symbols arbitrarily. Formally, we say a code is *p-unique-decodable* against pn worst-case substitutions if, for any $z \in \Sigma^n$, $|\mathcal{B}(z, pn) \cap \mathcal{C}| \leq 1$. In this definition, we think of z as the string Bob receives: no matter what Bob receives, there is at most one codeword it could have come from, assuming at most pn bit-flips were applied. So if the channel is constrained to applying pn bit-flips, Bob has enough information to recover the original string.

The unique-decoding *radius* p, the maximum p for which a code is p-unique-decodable, is closely related to the *minimum distance* of the code. The *minimum distance* of a code is $\Delta_H(\mathcal{C}) \stackrel{\text{def}}{=} \min_{x \neq y \in \mathcal{C}} \Delta_H(x, y)$, and the *minimum relative distance* of a code is $\delta_H(\mathcal{C}) \stackrel{\text{def}}{=} \frac{\Delta_H(\mathcal{C})}{n}$. The following proposition states that the unique-decoding radius is essentially half of the minimum distance (and, if we take $n \to \infty$, then they are exactly equal for families of codes). This proposition gives the intuitive geometric interpretation of a code: a code in the Hamming model is a set of points that are "spread out" in the Hamming metric (see Figure 2.3).

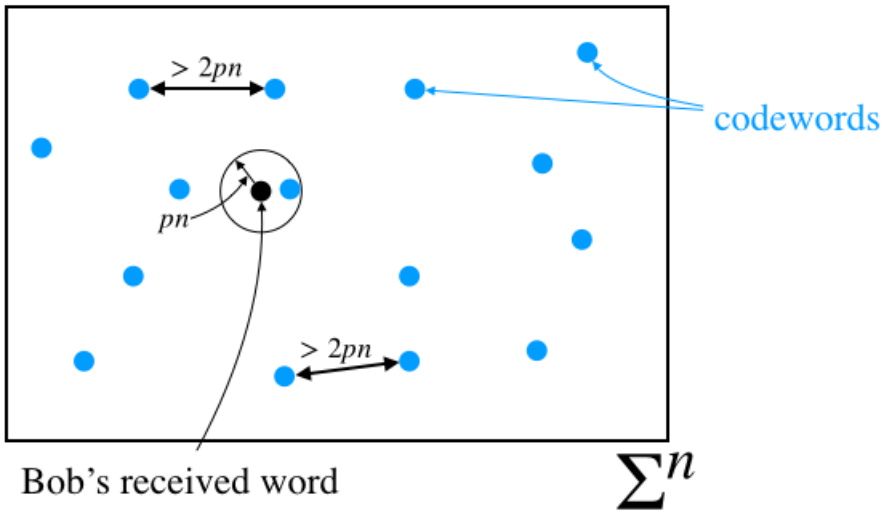


Figure 2.1: Proposition 2.1: A p-unique-decodable code is a set of points that are "spread out" in the Hamming metric. The definition says it is a set of points such that no pn-radius Hamming ball has more than one codeword, but it equivalently says that any two codewords have Hamming distance greater than $2pn$.

Proposition 2.1. *For p such that pn is an integer, a code $\mathcal{C}$ of length n is p-unique-decodable if and only if $p < \frac{\delta_H}{2}$.*

Proof. First, if $p < \frac{\delta_H}{2}$, then $\mathcal{C}$ is p-unique-decodable as no Hamming ball $\mathcal{B}(z, pn)$ contains two codewords c and c': if it did, then by the triangle inequality, $\Delta_H(c, c') \leq \Delta_H(c, z) + \Delta_H(z, c') \leq 2pn$, so $\delta_H(\mathcal{C}) \leq 2p$, a contradiction.

Now suppose $2p \geq \delta_H$. Then there exists two codewords c, c' at distance $\leq 2pn$. We can find a string z by flipping pn of the bits in x that differ from y, so that $\Delta_H(z, x) \leq pn$ and $\Delta_H(z, y) \leq 2pn - pn$. Then $\mathcal{B}(z, pn)$ has at least 2 codewords and $\mathcal{C}$ is not p-unique-decodable. □

For linear codes, we have the following simpler characterization of minimum distance: the minimum distance is the minimum Hamming weight of a nonzero codeword.

Proposition 2.2. *In a linear code, $\Delta_H(\mathcal{C}) = \min_{0 \neq c \in \mathcal{C}} \Delta_H(0, x)$.*

Proof. Clearly $\Delta_H(\mathcal{C}) \leq \min_{0 \neq c \in \mathcal{C}} \Delta_H(0, c)$. Furthermore, if there are two codewords c and c' at distance Δ, then $\Delta_H(0, c - c') = \Delta$, and $c - c' \in \mathcal{C}$, so $\Delta_H(\mathcal{C}) \geq \min_{0 \neq c \in \mathcal{C}} \Delta_H(0, x)$. □

We now look at a few classic bounds on the rate versus noise parameter tradeoff in the Hamming model, first for binary codes and then for codes with no alphabet size restriction (we informally call this the "large alphabet" setting).

Binary codes. We first present the Gilbert-Varshamov (GV) bound [40, 124], which shows the existence of p-uniquely-decodable codes with rate approaching $1 - h(2p)$. The proof, via the probabilistic method [3],

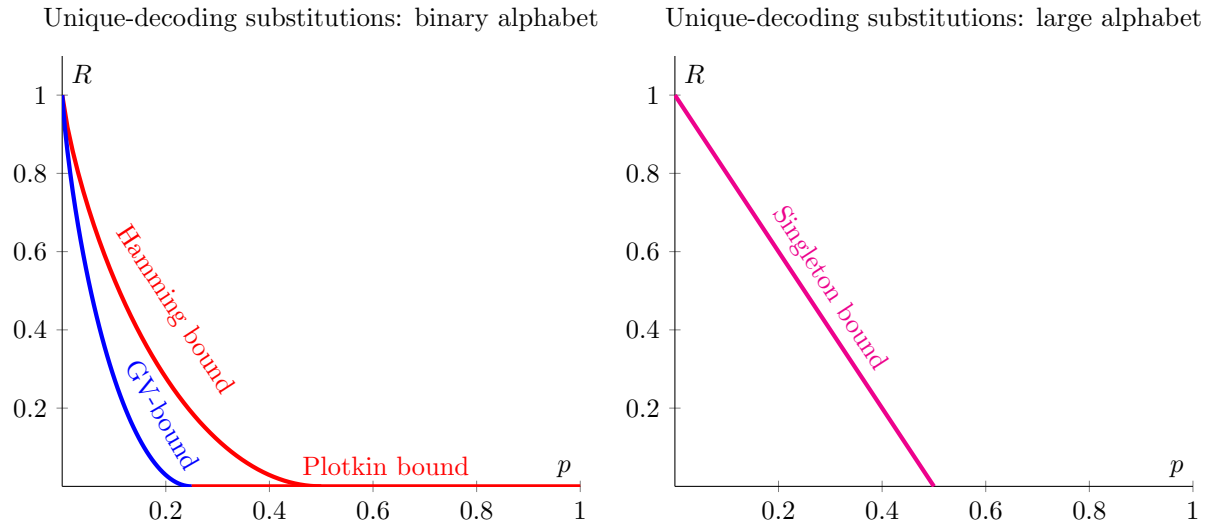


Figure 2.2: Simple bounds for unique-decoding substitutions. For binary alphabets: the Gilbert–Varshamov bound (Theorem 2.3), the Hamming Bound (Theorem 2.4), and the Plotkin bound (Theorem 2.5). For large alphabets: the Singleton bound (Theorem 2.6). Blue means a positive result, red means a negative (limitation) result, and purple means a matching upper and lower bound.

is characteristic of many proofs in coding theory: we choose a random code and show that it is p-unique-decodable with high probability, hence some instantiation of randomness gives a p-unique-decodable code. The random code here is in fact a *random linear code*, which we study the list-decoding properties of in Chapter 4.

Theorem 2.3 (GV Bound [40, 124])**.** *For all $\varepsilon > 0$ and $p \in (0, 1/4)$, for all positive integers n, there exist binary codes of rate $R = 1 - h(2p) - \varepsilon$ that are p-unique-decodable.*

Proof. Recall we let $k = Rn$. Let $G \in \mathbb{F}_2^{n \times k}$ be a matrix with independent uniformly random entries in $\mathbb{F}_2$. Let $\mathcal{C}$ be the subspace spanned by the columns of G. That is, $\mathcal{C} = \{Gv : v \in \mathbb{F}_2^k\}$. One can check that with probability $1 - 2^{-(n-k)}$, the columns of G are linearly independent, so with high probability $\mathcal{C}$ has dimension exactly k and thus rate R.

By Proposition 2.2, it suffices to show that, with high probability $\Delta_H(0, c) \geq pn$ for all nonzero codewords c. For any $v \in \mathbb{F}_2^k$, $v \neq 0$, we have Gv is distributed uniformly over $\mathbb{F}_2^n$, so $\mathbf{Pr}[Gv \in \mathcal{B}(0, pn)] = \frac{\mathrm{Vol}(n, pn)}{2^n} \leq \frac{2^{h(p)n}}{2^n} = 2^{-n(1-h(p))}$ using (2.1). Thus, by the union bound, the probability that there exists $v \neq 0$ such that $Gv \in \mathcal{B}(0, pn)$ is $2^{-n(1-h(p))} \cdot 2^{Rn} = 2^{-\varepsilon n}$. Thus, with probability $1 - 2^{-\varepsilon n}$, $\mathcal{C}$ is p-unique-decodable, as desired. □

A number of a limitations are known for unique-decodable codes. We describe a few simple ones here. The first one, known as the Hamming bound, comes from a packing argument.

Theorem 2.4 (Hamming Bound [77])**.** *For any family of p-unique-decodable binary codes of rate R, we have $R \leq 1 - h(p)$.*

Proof. For a p-unique-decodable code $\mathcal{C}$, the set of pn-radius Hamming balls around codewords are disjoint subsets of $\{0, 1\}^n$. Thus, we must have $2^n \geq |B(0, pn)| \cdot |\mathcal{C}| \geq 2^{n(h(p)-o(1))}|\mathcal{C}|$ by (2.1). Rearranging, we have $R(\mathcal{C}) \leq 1 - h(p) + o(1)$. □

The second limitation, known as the Plotkin bound, tells us that codes of non-vanishing rate are only p-unique-decodable for $p \leq 1/4$ (in fact, one can show $p < 1/4$).

Theorem 2.5 (Plotkin Bound [104])**.** *Any $1/4+\varepsilon$-unique-decodable binary code $\mathcal{C}$ satisfies $|\mathcal{C}| \leq \frac{1}{4\varepsilon}+1$.*

Proof. Since we are working over the Hamming metric, we can without loss of generality change the alphabet of the code to be the real numbers $\{-1,+1\}$. Suppose $\mathcal{C}$ has M codewords $c_1, \cdots, c_M \in \{-1,+1\}^n$. Since $\mathcal{C}$ is $1/4+\varepsilon$ unique-decodable, then $\Delta_H(c_i, c_j) \geq (1/2+2\varepsilon)n$ for all $i \neq j$ by Proposition 2.1, so we have $c_i \cdot c_j \leq (1/2-2\varepsilon)n - (1/2+2\varepsilon)n = -4\varepsilon n$. Then we have

$$0 \leq \left\| \sum_{i=1}^{M} c_i \right\|^2 = \sum_{i=1}^{M} \|c_i\|^2 + 2\sum_{i \neq j} c_i \cdot c_j \leq Mn + M(M-1)(-4\varepsilon n)$$

Simplifying and rearranging, we have $M \leq \frac{1}{4\varepsilon}+1$. □

Hence, combining Theorem 2.3 and Theorem 2.5 gives that the "zero-rate threshold" for worst-case substitutions for binary codes is $1/4$: non-vanishing rate codes can be p-unique-decodable if $p < 1/4$ but not if $p > 1/4$.

These are some of the simplest bounds, and the literature is far too vast to survey here. We refer the reader to the books [97, 65] for an introduction to coding theory and more known bounds.

Large alphabet codes. Over large alphabets (more precisely, codes with no constraint on the alphabet size), the optimal rate versus noise parameter tradeoff is known, and the limitation result is given by the Singleton bound.

Theorem 2.6 (Singleton bound [117])**.** *For a code $\mathcal{C}$ with dimension k, we have $\Delta(\mathcal{C}) \leq n-k+1$. In particular, any family of p-unique-decodable codes has rate $R \leq 1-2p$.*

Proof. By the pigeonhole principle, there exist two codewords that agree on the first $k-1$ symbols, which means their Hamming distance is at most $n+1-k$, so $\Delta(\mathcal{C}) \leq n+1-k$. Proposition 2.1 gives that $R \leq 1-2p$ for any family of p-unique-decodable codes of rate R. □

The random GV-bound argument can also be applied to codes over large alphabets, showing that for sufficiently large alphabets there exist codes with rate approaching $1-2p$, so the Singleton bound is achievable . These codes are again obtained by the probabilistic method, and thus non-explicit.

Reed–Solomon codes [106] are a classic family of explicit codes achieving the Singleton bound. Let $\alpha_1, \ldots, \alpha_n$ be distinct elements of $\mathbb{F}_q$. The $[n,k]$ Reed–Solomon code over $\mathbb{F}_q$ with evaluation points $(\alpha_1, \ldots, \alpha_n)$ is defined as

$$\mathcal{C} = \{(f(\alpha_1), \ldots, f(\alpha_n)) : f \in \mathbb{F}_q[X], \deg(f) < k\}.$$

Theorem 2.7 (Reed–Solomon codes)**.** *The above code $\mathcal{C}$ has minimum distance $\Delta(\mathcal{C}) = n-k+1$, and thus is $(1-R)$-unique-decodable.*

Proof. The proof is a simple corollary of the basic fact that degree-less-than-k polynomials have less than k roots. Indeed, let f_1, f_2 be two degree-less-than k polynomials. Then $f_1 - f_2$ has less than k roots, so

$f_1(\alpha_i) = f_2(\alpha_i)$ for at most $k-1$ values of i. This holds for any f_1, f_2, so any two codewords agree on at most $k-1$ positions, so their Hamming distance is at least $n-k+1$. □

This thesis. This thesis considers the two simple variations of the Hamming model: worst-case deletions and list-decoding. Worst-case deletions have the same setup as the Hamming model except we consider deletion errors instead of substitution errors. List-decoding is the same setup as the Hamming model except the recovery guarantee is relaxed, so that Bob, rather than outputting the correct codeword, only needs to output a small list of codewords containing the correct codeword.

2.4 Worst-case deletions

In this setting, we consider a channel that applies *worst-case deletions*. Codes correcting deletions are motivated by applications such as internet transmission [42, 29, 100, 99], DNA storage [90, 118, 105], and magnetic recording [20, 19]. However, deletions are poorly understood with many gaps in our theoretical understanding, and hence few applications in practice.

In this setting, when Alice sends n symbols on a channel that can adversarially delete a fraction p of the bits, Bob receives a subsequence of length $(1-p)n$. Crucially, Bob does not know the location of the deleted bits. To do this, Alice must restrict the sequence of transmitted bits to a code $\mathcal{C} \subset \Sigma^n$ so that every $c \in \mathcal{C}$ can be unambiguously identified from an arbitrary subsequence of c of length $(1-p)n$. It is easy to see that this property is equivalent to the property that for every two distinct codewords $c, c' \in \mathcal{C}$, the length of their longest common subsequence, denoted $\mathrm{LCS}(c, c')$, is less than $(1-p)n$. Defining $\mathrm{LCS}(\mathcal{C})$ to be the largest value of $\mathrm{LCS}(c, c')$ over all distinct pairs $c, c' \in \mathcal{C}$, we therefore have the following definition.

Definition 2.8. A code $\mathcal{C} \subseteq \{0,1\}^n$ is corrects p *fraction of worst-case deletions* if $\mathrm{LCS}(\mathcal{C}) < (1-p)n$.

In Chapter 3, we consider a very basic question about this definition: what is the zero-rate threshold, the supremum of p for which codes of non-vanishing rate can correct p fraction of worst-case deletions? As we saw with Theorem 2.3 and Theorem 2.5, this zero-rate threshold for substitutions for binary codes has long been known to be $1/4$, but for deletions, determining the zero-rate threshold is still open. In fact, our knowledge is even more embarrassing: trivially any code correcting $n/2$ deletions has at most 2 codewords (among any three length-n strings, there are two strings with either $0^{n/2}$ or $1^{n/2}$ as a common subsequence), and thus the zero-rate threshold for worst-case deletions is at most $1/2$. This trivial bound on the zero-rate threshold had remained the best-known ever since the more general rate versus deletion fraction question (Question 1.1) was studied over 50 years ago [122]. In Chapter 3, we give the first improvement of this trivial bound.

2.5 List-decoding

A code $\mathcal{C}$ is (p, L)-list-decodable if, for any $z \in \Sigma^n$, we have $|\mathcal{B}(z, pn) \cap \mathcal{C}| \leq L$. Like in unique decoding, we think of z as Bob's received string, and the definition requires that, if the channel applies pn deletions, Bob can produce a list of L codewords (the L codewords in $\mathcal{B}(z, pn)$) containing the correct one. Note that being $(p, 1)$-list-decodable is equivalent to being p-unique-decodable, so list-decoding generalizes the

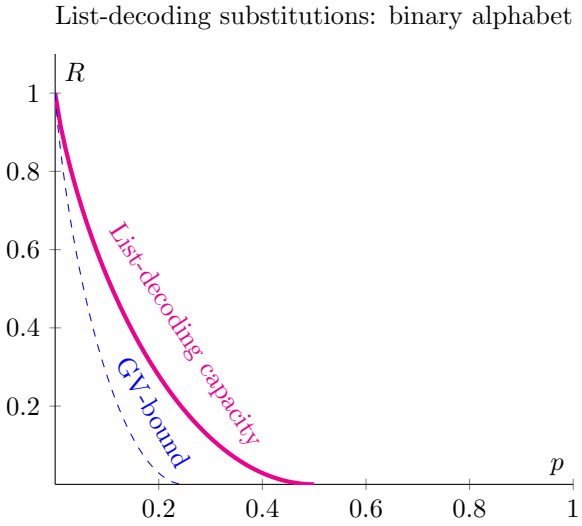


Figure 2.3: The list-decoding capacity over binary alphabets. The capacity is given by $R = 1 - h(p)$ (Theorem 2.9). The GV-bound of $1 - h(2p)$ (Theorem 2.3), the essentially best-known achievable rate versus noise parameter tradeoff for unique-decoding binary codes, is plotted for comparison.

Hamming model. Since list-decoding was introduced in the 1950's [34, 127] in the context of communication, it has found other applications, for example in pseudorandomness [123] and complexity theory [119].

The best list-decodable codes we know of are often non-explicit, random codes. In list-decoding, the rate R and noise parameter p, which we call the *(list-decoding) radius*, are the main parameters that we would like to trade off (we want both to be large). We would like to do this while keeping the list-size L small, ideally a constant (but $\text{poly}(n)$ is okay). The optimal trade off between R and p is known for list-decoding:

- For binary codes, the optimal tradeoff is $R = 1 - h(p)$. That is, for $R < 1 - h(p)$, there exist a family of list-decodable codes with list-decoding radius p and constant L, and for $R > 1 - h(p)$, the list size must be exponential in n (see Theorem 2.9 below).
- For codes with no alphabet restriction, the optimal tradeoff is $R = 1 - p$. That is, for $R < 1 - p$, there exist a family of list-decodable codes with list-decoding radius p and constant L, and for $R > 1 - p$, the list size must be exponential in n.

Codes whose rate approaches the optimal tradeoff (binary codes of rate $1 - h(p) - \varepsilon$ or large-alphabet codes of rate $1 - p - \varepsilon$) are called *capacity-achieving* codes.

Binary codes. The *list-decoding capacity theorem* is a classic result that establishes the optimal rate versus error rate tradeoff for list-decoding binary codes. A similar statement holds for q-ary codes, but we focus on the binary case here.

Theorem 2.9 (List decoding capacity theorem [34, 127])**.** *Let $p \in (0, 1/2)$ and $\varepsilon > 0$.*

1. *There exist binary codes of rate $1 - h(p) - \varepsilon$ that are $(p, \lceil 1/\varepsilon \rceil)$-list-decodable.*
2. *Any binary code of rate $1 - h(p) + \varepsilon$ that is (p, L)-list-decodable up to distance p must have $L \geq 2^{\Omega(\varepsilon n)}$.*

Proof. For the first part, let $L = \lceil 1/\varepsilon \rceil$ and $R = 1 - h(p) - \varepsilon$. Let $\mathcal{C} = \{c_1, \ldots, c_{2 \cdot 2^{Rn}}\}$ where $c_1, \ldots, c_{2 \cdot 2^{Rn}}$ are i.i.d. binary strings in $\{0,1\}^n$. It is easy to check that with high probability $|\mathcal{C}| \geq 2^{Rn}$ so $\mathcal{C}$ has rate (at least) R. Then,

$$\begin{aligned}
\mathbf{Pr}[\mathcal{C} \text{ not } (p,L)\text{-list-decodable}] &= \mathbf{Pr}[\exists z \in \mathbb{F}_2^n : |\mathcal{B}(z, pn) \cap \mathcal{C}| \geq L+1] \\
&\leq 2^n \cdot \mathbf{Pr}[|\mathcal{B}(0, pn) \cap \mathcal{C}| \geq L+1] \\
&\leq 2^n \cdot (2 \cdot 2^{Rn})^{L+1} \, \mathbf{Pr}[c_1, c_2, \ldots, c_{L+1} \in \mathcal{B}(0, pn)] \\
&= 2^n \cdot (2 \cdot 2^{Rn})^{L+1} \cdot \left(\frac{\mathrm{Vol}(n, pn)}{2^n} \right)^{L+1} \\
&\leq 2^n \cdot (2 \cdot 2^{Rn})^{L+1} \cdot \left(\frac{2^{h(p)n}}{2^n} \right)^{L+1} \\
&= 2^{L+1} \cdot 2^{n - n\varepsilon(L+1)} < 2^{-\Omega(n)}.
\end{aligned}$$

Thus, with high probability $\mathcal{C}$ has rate R and is (p, L) list-decodable, as desired.

For the second part, suppose the code has rate $1 - h(p) + \varepsilon$. The number of pairs $\{(z, c) \in \mathbb{F}_2^n \times \mathcal{C} : \Delta_H(z, c) \leq pn\}$ is $\mathrm{Vol}(n, pn) \cdot 2^{Rn} \geq 2^{n(1+\varepsilon - o(1))}$, by (2.1). By the pigeonhole principle, there exists some z such that $\mathcal{B}(z, pn)$ has at least $2^{n(\varepsilon - o(1))}$ codewords, so the code is not $(p, 2^{n(\varepsilon - o(1))})$-list-decodable. □

Theorem 2.9 is remarkable because it means than even when pn is much larger than the unique-decoding radius, it still can be the case that only a constant number of codewords $c \in \mathcal{C}$ lie in any Hamming ball of radius pn. Indeed, comparing Theorem 2.9 with the GV bound Theorem 2.3, which is still essentially the best bound for uniquely-decodable codes, the maximum list-decoding radius of a (p, L) list-decodable code is roughly $h^{-1}(1-R)$, while the maximum known unique-decoding radius of a p-unique-decodable code is roughly half of the list-decoding radius, $\frac{h^{-1}(1-R)}{2}$.[1] Because of this, there has been a great deal of work attempting to understand what codes achieve *list-decoding capacity*, the bound in Theorem 2.9. Theorem 2.9 shows that a uniformly random code achieves this list-decoding capacity with high probability, but are there other, more explicit codes? In Chapter 4, we make progress towards this goal by giving a very tight analysis of *random linear binary codes*, showing that random linear binary codes have the same (and in fact slightly better, in the list-size) list-decoding properties as uniformly random codes.

Large alphabet codes. Over general alphabets, the Singleton bound upper bounds the rate versus error radius tradeoff for list-decoding.

Theorem 2.10 (Singleton bound)**.** *For (p, L)-list-decodable codes with $L \leq \operatorname{poly} n$, we have $R \leq 1 - p$.*

Proof. Suppose $R = 1 - p + \varepsilon$ and $\mathcal{C}$ is (p, L) list-decodable. By the pigeonhole principle, there exist $2^{\varepsilon n}$ codewords agreeing on the first $(1-p)n$ symbols. Let c be one such codeword. Then the Hamming ball $B(c, pn)$ has at least $2^{\varepsilon n}$ codewords, so we must have $L \geq 2^{\varepsilon n}$. □

Using the same probabilistic method argument as in Theorem 2.9, by taking random codes over an alphabet of size $2^{O(1/\varepsilon)}$ where ε is the gap to capacity, we can show that the Singleton bound is achievable. The proof is similar to Theorem 2.9 so we omit it.

[1] Here we use $h^{-1}(\cdot) : [0,1] \to [0, 1/2]$ to denote the inverse of the binary entropy function on $[0, 1/2]$, which exists as $h(\cdot)$ is one-to-one on $[0, 1/2]$

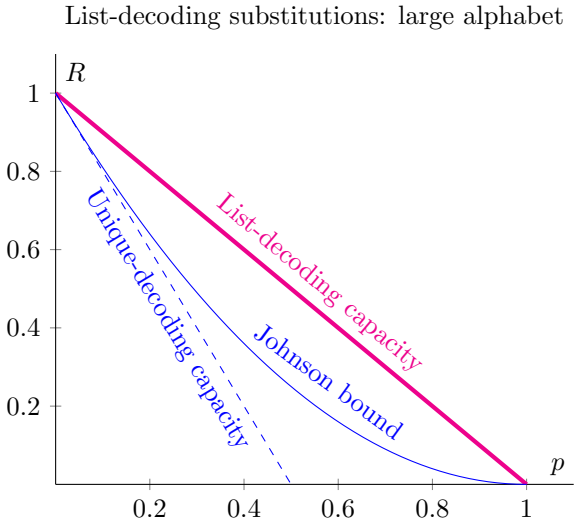


Figure 2.4: The list-decoding capacity over general alphabets. The capacity is given by $p = 1 - R$ (Theorem 2.10 and Theorem 2.11). The optimal tradeoff for unique-decoding, $p = (1 - R)/2$, (Theorem 2.6 and Theorem 2.7) is shown for comparison, along with $p = 1 - \sqrt{R}$, the Johnson bound (Theorem 2.12) applied to optimal unique-decodable codes such as Reed–Solomon codes.

Theorem 2.11. *For any $\varepsilon > 0$ and $p \in (0, 1)$, there exists rate $R = 1 - p - \varepsilon$ codes that are $(p, O(1/\varepsilon))$-list-decodable.*

Like in the binary case, the original proof of Theorem 2.11 is by the probabilistic method and thus gives non-explicit codes, but unlike in the binary case, we do have explicit codes [64, 45, 87, 86] achieving capacity in the large alphabet setting. These codes are all generalizations the classic and ubiquitous Reed–Solomon codes (Theorem 2.7). However, despite this progress, we still do not know whether Reed–Solomon codes themselves achieve list-decoding capacity. This is a natural question as Reed–Solomon codes *are* optimally list-decodable in the special case of $L = 1$ (a.k.a. unique-decoding, Theorem 2.7). Additionally, if Reed–Solomon codes achieve list-decoding capacity, they would offer advantages over existing explicit list-decodable codes, such as simplicity and potentially smaller alphabet sizes.

The *Johnson bound* helps us understand the list-decodability of Reed–Solomon codes. The Johnson bound is a classic bound that shows any p-unique-decodable code is automatically p'-list-decodable for some $p' > p$. Here, we state the alphabet-independent version of the Johnson bound [82]. For codes over a fixed alphabet size q, the bound can be strengthened (see [65, Theorem 7.3.1]).

Theorem 2.12 (Johnson bound [82])**.** *For $p \in (0, 1/2)$, if a code $\mathcal{C}$ over an alphabet of size q is p-unique-decodable, the $\mathcal{C}$ is $(1 - \sqrt{1 - 2p}, L)$-list-decodable for $L \leq O(nq)$.*

One can indeed check that for all $p \in (0, 1/2)$, we have $1 - \sqrt{1 - 2p} > p$, so indeed list-decoding allows for more error than unique decoding. As an immediate corollary of the Johnson bound, Reed–Solomon codes of rate R, which are unique-decodable up to radius $\frac{1-R}{2}$, are list-decodable up to radius $1 - \sqrt{R}$. Whether Reed–Solomon codes are list-decodable beyond this $1 - \sqrt{R}$ radius implied by the Johnson bound, and, optimistically, all the way to the optimal radius $1 - R$, is the main question considered in Chapter 5.

Chapter 3

Deletion codes

3.1 Introduction

This chapter considers the limits of reliable communication against an adversarial deletion channel. Throughout this chapter, we use capital N rather than lowercase n for the length of the code for notational convenience (n has a different meaning here). Recall $\mathrm{LCS}(\mathcal{C})$ is the largest value of $\mathrm{LCS}(x, y)$ over all distinct pairs $x, y \in \mathcal{C}$, and recall a code $\mathcal{C} \subseteq \{0,1\}^N$ is a *p-deletion correcting code* if $\mathrm{LCS}(\mathcal{C}) < (1-p)N$. If there exists a family of such codes $\mathcal{C}$ whose rates are bounded away from zero as $N \to \infty$, we say it is possible to achieve a non-vanishing rate of information communication. For any noise model of interest, one of the basic goals is to understand the threshold noise level below which it is possible to communicate with non-vanishing rate. For example, recall it is well-known that for a channel that flips an adversarially chosen set of at most pN bits, the threshold value for the error-fraction p equals $1/4$ (see discussion in Section 2.3). On the other hand, for the adversarial deletions channel, this fundamental question remains unsolved:

> *What is the largest fraction of deletions $p \in (0,1)$ for which it is possible to achieve zero-error communication with non-vanishing information rate?*

Formally, define the *zero-rate threshold* of adversarial bit-deletions is

$$p_{\mathrm{del}}^{\mathrm{thr}} := \sup\{p \mid \exists\ \alpha_p > 0 \text{ s.t for infinitely many } N \text{ there is a subset } \mathcal{C} \subset \{0,1\}^N \text{ with } \mathrm{LCS}(\mathcal{C}) < (1-p)N \text{ and } |\mathcal{C}| \geq 2^{\alpha_p N}\}\ . \tag{3.1}$$

The main question is then: What is the value of $p_{\mathrm{del}}^{\mathrm{thr}}$? We have a trivial upper bound $p_{\mathrm{del}}^{\mathrm{thr}} \leq 1/2$. Indeed, among any three strings $x, y, z \in \{0,1\}^N$, there must be two with the same majority bit, and thus a common subsequence (of all 0's or all 1's) of length at least $N/2$. Thus any $\frac{1}{2}$-deletion correcting code $\mathcal{C} \subset \{0,1\}^N$ satisfies $|\mathcal{C}| \leq 2$.

The value of $p_{\mathrm{del}}^{\mathrm{thr}}$ remains unknown. Even more starkly, as simplistic as the above argument is, it was not previously known if $p_{\mathrm{del}}^{\mathrm{thr}}$ is strictly bounded away from $\frac{1}{2}$, or whether there are in fact codes of non-vanishing rate for correcting a $(\frac{1}{2} - \delta)$ fraction of deletions for any desired $\delta > 0$. This tantalizing question was implicit in early works on deletion codes, particularly in [122], which gave bounds on the achievable tradeoffs between rate and deletion fraction, and was explicitly raised in [84]. Since then, this question has

been mentioned in several works, including the work of Bukh and Ma [15] which showed that an upper bound of $\frac{1}{2} - \frac{1}{\text{poly}\log N}$ on the correctable deletion fraction, and many recent works on deletion code constructions such as [125, 56, 70, 14, 57, 47], other works on coding theory [128], as well as the recent surveys [24, 75]. In the other direction, the best known lower bound $p_{\text{del}}^{\text{thr}} > \sqrt{2} - 1$ is due to [14], who constructed explicit binary codes of non-vanishing rate to correct a fraction of deletions approaching $\sqrt{2} - 1$ (see [112, 84] for prior constructions).

3.1.1 Our results

In this chapter, we prove the first nontrivial upper bound on $p_{\text{del}}^{\text{thr}}$.

Theorem 3.1. *There exists an absolute constant $\delta_0 > 0$ such that $p_{del}^{thr} \leq \frac{1}{2} - \delta_0$. More concretely, there exists absolute constants $A, \delta_0 > 0$ such that for all large enough N, any binary code $\mathcal{C} \subset \{0,1\}^N$ tolerating $(\frac{1}{2} - \delta_0)N$ adversarial deletions must satisfy $|\mathcal{C}| \leq 2^{(\log N)^A}$.*

We show the above in the contrapositive form—in any code $\mathcal{C} \subset \{0,1\}^N$ of quasi-polynomial size, we find two codewords $s, t \in \mathcal{C}$ with $\text{LCS}(s,t) > (\frac{1}{2} + \delta_0)N$. We made no attempts to optimize the value of δ_0 but regardless it is very small for our argument. In Section 3.2, we give an overview of the proof and an outline of this chapter. In the remainder of this introduction we survey some generalizations of Theorem 3.1 and connections to other problems in coding theory.

Non-binary alphabets. We generalize Theorem 3.1 to alphabets of larger size. Let us denote the quantity analogous to (3.1) for any fixed alphabet size $q \geq 2$, namely the zero-rate threshold for q-ary deletion codes, by $p_{\text{del}}^{\text{thr}}(q)$. The trivial upper bound is $p_{\text{del}}^{\text{thr}}(q) \leq 1 - 1/q$; this corresponds to finding a common sequence of at least N/q repeated i's between two strings that share the same most frequent symbol $i \in \{0, 1, \ldots, q-1\}$, in any code of size bigger than q. Just as in the binary case, no improvement over this trivial bound was previously known.

For any code $\mathcal{C} \subseteq \{0, 1, \ldots, q-1\}^N$ over an alphabet of size $q > 2$, we may pick some two symbols i, j and a set $\mathcal{C}_{i,j} \subseteq C$ of at least $|\mathcal{C}|/q^2$ strings whose two most frequent symbols are i and j. We can then obtain a binary code $\mathcal{C}' \subseteq \{i,j\}^{2N/q}$ by restricting each element of $\mathcal{C}_{i,j}$ to a substring of length $2N/q$ consisting only of i's and j's. Applying Theorem 3.1 to $\mathcal{C}'$, we see that some two strings in $\mathcal{C}'$ have a common subsequence with length at least $(\frac{1}{2} + \delta_0)\frac{2N}{q}$. We have thus shown the following theorem as an immediate corollary of Theorem 3.1.

Theorem 3.2. *Fix an integer $q \geq 2$. Then*

$$p_{del}^{thr}(q) \leq 1 - \tfrac{1+2\delta_0}{q} < 1 - \tfrac{1}{q} \ ,$$

where $\delta_0 > 0$ is the positive constant promised in Theorem 3.1.

We note that for the simpler model of erasures where the location of missing symbols *are* known to the decoder, the zero-rate threshold equals $1 - 1/q$.[1] Thus our results also show a formal separation between the zero-rate threshold for the models of erasures and deletions, for any fixed alphabet.

[1] The erasure fraction correctable by a code is exactly governed by its relative (Hamming) distance. The Plotkin bound shows that the rate must be vanishing for relative Hamming distance $1 - 1/q$. We know the existence and even explicit constructions of codes of non-vanishing rate and relative Hamming distance $1 - 1/q - \varepsilon$ for any $\varepsilon > 0$.

In the list-decoding model with list-size L for deletion fraction p, there can be up to L codewords that contain the (arbitary) input sequence $y \in \{0,1\}^{(1-p)N}$. The zero-rate threshold for *list-decoding* from deletions, as the list-size $L \to \infty$, is known to equal $1 - 1/q$ [70]. Thus our result also demonstrates that list-decoding is provably more powerful in terms of the deletion fractions that can be handled with non-vanishing rate.

3.1.2 Related works

Performance of random codes. An ubiquitous approach to establish strong, and in many cases the best known, *possibility* results in coding theory is to analyze random codes of certain rates. These results typically also identify the precise performance threshold of random codes [60]. For the case of binary deletion codes, however, the performance of random codes itself is hard to analyze, as we do not rigorously know a tight estimate of the expected length γN of the longest common subsequence of two random N-bit strings (γ is called the Chvátal-Sankoff constant [26]). The known bounds on this expectation γN [95], together with standard probabilistic arguments, imply that with high probability, random codes can tolerate a deletion fraction at least 0.17, but also at most 0.22 [84]. For codes over alphabet size q, random codes can correct a deletion fraction approaching $1 - 2/\sqrt{q}$ for large q [85].

As mentioned earlier, we now have constructions of binary codes that can correct a deletion fraction 0.414 [14], which is substantially better than random codes. This raised the possibility that perhaps there might be binary codes of non-vanishing rate capable of correcting a deletion fraction all the way up to the trivial limit of $\frac{1}{2}$, which we refute in this chapter.

Trade-offs for correcting $N/2 - N^{1-\theta}$ deletions. In terms of previously known limitations of deletion codes, Bukh and Ma [15] showed that for each fixed r and large enough N (specifically, at least $r^{O(r)}$), every set $\mathcal{C} \subseteq \{0,1\}^N$ of size $r+4$ satisfies

$$\mathrm{LCS}(\mathcal{C}) \geq \tfrac{N}{2} + \Omega(r^{-9})N^{1-1/r} \ . \tag{3.2}$$

Choosing r appropriately, the result (3.2) implies that there exist absolute constants b, c such that every code $\mathcal{C} \subseteq \{0,1\}^N$ with $\mathrm{LCS}(\mathcal{C}) < \frac{N}{2} + \frac{N}{(\log N)^b}$ has size $|\mathcal{C}| \leq c\frac{\log N}{\log\log N}$. Bukh and Ma demonstrated that this $N^{1-1/r}$ advantage in (3.2) is asymptotically sharp for each fixed r, by exhibiting a set $\mathcal{W}$ of $(r+4)$ N-bit strings with $\mathrm{LCS}(\mathcal{W}) \leq \frac{N}{2} + O(N^{1-1/r})$. Interestingly, this set $\mathcal{W}$ played a crucial role in the developments on *constructions* of codes to correct a large fraction of deletions in [14, 57], as well as codes achieving the zero-rate threshold for list decoding from insertions and deletions in [47]. A suitable modification of this Bukh-Ma code $\mathcal{W}$ also drives the best known 0.414-deletion correcting codes of [14].

Twins and regularity techniques. One of the ideas used in this chapter is a new regularity-type result about strings. Szemerédi's regularity lemma and its variants are ubiquitous in extremal and additive combinatorics, but applying these ideas to coding theory is a relatively recent development. The first example of such a result was proved by [5], and their regularity lemma roughly shows that every long string can be partitioned into a constant number of consecutive substrings, each of which is regular (a regular string is one in which the one-density in any long consecutive substring is close to the one-density in the whole string). They used this regularity lemma to prove that every string of length N contains two disjoint copies of some length $(1/2 - o(1))N$ subsequence (so-called "twins"). Given the similarity between finding twins

in a single string and finding long common sequences between different strings, it should not come as a surprise that these techniques are useful here as well. The main difference in our approach is that we require a stronger regularity condition, which is that every substring not only has the same one-density but has similar "oscillation statistics" at many scales with the parent string.

3.1.3 Deletion correction in related models

To offer some wider context, we now discuss some results related to the broader study of codes for deletions and synchronization errors under various channel assumptions.

Non-adversarial models. This chapter focuses on the adversarial model, where an arbitrary subset of p fraction of the codeword bits, can be deleted. There is a rich body of work on the binary deletion channel where each codeword bit is deleted i.i.d with probability p. In this case, it is known that one can have positive rate codes that ensure vanishing miscommunication probability even for p approaching 1 (so the zero-rate threshold equals 1). The interested reader can find more information about codes for the deletion channel in the surveys [100, 24].

One can consider models that are intermediate in power between i.i.d random and adversarial channels. For instance, in the *oblivious* model the deletion pattern can be chosen arbitrarily, but without knowledge of the codeword. In this case, too, the zero-rate threshold is 1, as for any $p < 1$, Guruswami and Li [57] showed the existence of codes that ensured that for every pattern of p-fraction deletions *most* codewords are communicated correctly.[2] Their work also considered the *online* model, where the decision to delete the i-th bit must be made based only on the first i bits of the codeword. They showed that the zero-rate threshold for this model (again, for the average-error criterion of ensuring most codewords are communicated correctly for any deletion pattern) equals $\frac{1}{2}$ if and only if $p_{\text{del}}^{\text{thr}} = \frac{1}{2}$. By virtue of Theorem 3.1, this implies that the zero-rate threshold for the online model is also bounded away from $\frac{1}{2}$.

Large alphabets. We focused on codes over the binary and fixed small alphabets in this chapter. This is in fact the most challenging setting for deletion codes. Indeed, if the code alphabet is allowed to grow with N, then one can include the index i along with the i-th codeword symbol, effectively reducing the deletion model to the much simpler erasure model, where Reed-Solomon codes give a simple, optimal solution. For alphabets that are large, but still independent of N, a natural greedy strategy shows the existence of codes of rate $(1 - p - \varepsilon)$ capable of correcting a fraction p of deletions, over an alphabet of size $\exp(O(1/\varepsilon))$ [70]. In particular, the zero-rate threshold approaches 1. Also, $1 - p$ is a trivial upper bound on the possible rate, even for the simpler model of p fraction of erasures. Explicit constructions of p-deletion correcting of rate approaching $1 - p$ over an alphabet size independent of N were given in [74] based on synchronization strings, which is a very elegant tool that has since found several other applications (see the survey [75]).

Insertions and deletions. Another form of errors that affect the synchronization between sender and receiver are insertions of symbols. It is well known (since [91]) that a code $\mathcal{C}$ with $\text{LCS}(\mathcal{C}) < (1 - p)N$ can tolerate any combination of a total of pN insertions and deletions (insdel errors). Thus allowing insertions as well does not change the combinatorial aspects of the underlying coding problem, as it is governed by the

[2] This average-case criterion to achieve decoding success for most, as opposed to all, codewords is necessary, as otherwise tackling the oblivious model becomes as hard as tackling the adversarial model. Alternatively, one can allow a stochastic encoder, and ensure high probability of successful transmission of each message when averaged over the choice of its random encoding.

LCS. However, for efficient algorithms for insdel errors are not implied by deletion correction algorithms, and have to be reworked [56].

In the model of list-decoding, even the combinatorial aspects are more nuanced in the presence of insertions. The trade-off between the combinations of fractions of insertions and deletions that governs the zero-rate region exhibits an interesting piece-wise linear behavior [47]. (As mentioned earlier, the zero-rate threshold for list-decoding q-ary codes from deletions alone equals $1 - 1/q$.)

Low-deletions regime. This chapter focused on the largest deletion fraction that can be corrected with non-vanishing rate. At the opposite end of the spectrum are codes to correct a deletion fraction $p \to 0$. In this case, the optimal rate behaves as $1 - O(p \log(1/p))$ and we also know explicit codes with such rate $1 - O(p \log^2(1/p))$ and efficient deletion-correction algorithms [21, 73]. There has also be an active line of recent code constructions, triggered by [13], for correction of a fixed *number* k of deletions with redundancy at most $c_k \log N$. We now have codes with the optimal (up to constant factors) redundancy of $O(k \log N)$ [21, 73, 115, 116].

3.2 Proof overview

In this section, we give a high-level overview of the proof of Theorem 3.1, as well as the organization of the rest of this chapter. Let $\mathcal{C} \subseteq \{0,1\}^N$ be a binary code of size $2^{(\log N)^A}$ for some large constant A, and our goal will be to find two elements $s, t \in \mathcal{C}$ for which $\mathrm{LCS}(s,t) \geq (1/2 + \delta_0)N$. The proof breaks down into five conceptually independent parts that roughly correspond to the Sections 3.4 through 3.8. It will be natural to explain these parts here in roughly reverse order, starting with Section 3.8.

Pigeonholing by "statistics". The only place where we use the size of $\mathcal{C}$ is in the final Section 3.8, which wraps up the proof of Theorem 3.1. We need $\mathcal{C}$ to be large enough to find by the pigeonhole principle two elements $s, t \in \mathcal{C}$ with similar "macroscopic statistics." That is, we pick s and t to have the same number of ones in every long subinterval of length $N/\operatorname{poly}\log N$, and also to share some other statistics that characterize the "frequencies at which they oscillate." As there are only $O(\operatorname{poly}\log N)$ long intervals and the statistics in question take on only $2^{\operatorname{poly}\log N}$ possible values on each interval, $\mathcal{C}$ is large enough to guarantee the existence of two s and t sharing identical statistics on all such intervals. The remaining sections explain how to define the statistics we care about and show that if s and t have identical statistics, then they must have a long LCS.

We now describe the three different high-level strategies we use for finding long common subsequences between s and t.

Strategy 1: Globally imbalanced strings. The first strategy is extremely simple: match corresponding ones (or zeros if there are more zeros) in s and t. In the case that s and t are significantly imbalanced, this strategy immediately finds an LCS of length noticeably more than $N/2$. This naive strategy is illustrated in Figure 3.1. It may be helpful to visualize the strategy as sending two runners, one down the length of each string, who must advance simultaneously while holding hands and only step on the ones in their respective strings.

Strategy 2: Green strategy. The other two strategies, which we call the Green strategy and the Blue-Yellow strategy, are both modifications of the naive strategy. We first describe the Green Strategy, which

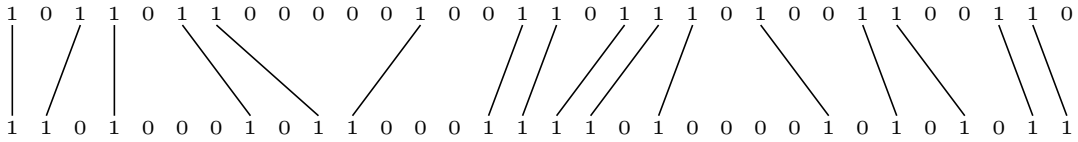


Figure 3.1: In the naive strategy, we match ones greedily between s and t. In this example, the strings are balanced so the naive strategy finds a common subsequence of length exactly 16 between s and t of length 32.

is carried out in Section 3.6, as it is simpler. The naive strategy above can be thought of as a scanning process, where two runners move along the ones in s and t simultaneously, matching bits together to find a common subsequence composed entirely of ones. In the Green strategy, we fix an "oscillation period" $\ell \geq 1$ and preprocess s by counting, for every index i of a one-bit in s, the number of zeros between the i-th one and the $(i+\ell)$-th one. For each such i, we plant a marker there, which we call a *Green ℓ-flag*, if there are more than $(1+\varepsilon)(\ell-1)$ zeros in this interval. Since there are exactly $\ell-1$ ones in this same interval, the Green ℓ-flags are meant to signal to the runners that they are entering a zero-rich patch within s. The same preprocessing is also done in t.

In the Green strategy, the two runners proceed in the same way as the naive strategy except that each time the runners reach Green flags simultaneously, they switch to stepping only on zeros for the duration of the flagged regions. As there are more zeros than ones in the flagged regions, they pick up an advantage over the naive strategy for every single pair of Green flags they simultaneously match. The Green strategy is pictured in Figure 3.2.

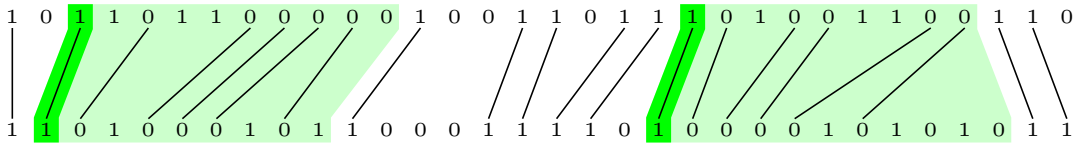


Figure 3.2: In the Green strategy, we find a long common subsequence by matching zeros instead of ones in the zero-rich patches immediately following Green ℓ-flags. In this example $\ell = 4$, the darker Green one-bits are Green ℓ-flags, and the light Green substrings following them are zero-rich patches. The Green strategy finds a common subsequence of length 20 between the s and t pictured above.

Our analysis of the success of the Green strategy is conditioned on the existence of a single oscillation period ℓ for which many of these zero-rich patches exist in both s and t. Indeed, suppose there exists ℓ for which a constant $g_\ell = \Omega(1)$ fraction of the ones in both s and t are Green ℓ-flags. Typically, we expect that the two runners hit Green flags simultaneously a constant g_ℓ^2 fraction of the time (this can be made rigorous by randomly shifting the starting position of one of the runners slightly). Thus, using the Green strategy one can find a common subsequence of length $(1/2 + g_\ell^2\varepsilon)N$. The Green case finishes the proof of Theorem 3.1 if there exists any single oscillation period ℓ for which a constant fraction of ones in s and t are Green ℓ-flags.

Unfortunately, it is not always the case that a string s has a single oscillation period ℓ as above. Indeed, if

$$s_i \stackrel{\text{def}}{=} (1^{2^i}0^{2^i})^{2^{k-i-1}},$$

then each s_i is a string of length 2^k which oscillates with period 2^i. It is not hard to check that the

concatenation $s \stackrel{\text{def}}{=} s_0 s_1 \cdots s_{k-1}$ is a string of length $N = k \cdot 2^k$, such that there are at most $O(2^k)$ Green ℓ-flags in s for any given choice of ℓ. Thus, $g_\ell = O((\log N)^{-1}) = o(1)$ for every single ℓ, so the Green strategy is insufficient for this type of string. We remark that is essentially the worst case, and one can always find two strings s and t in $\mathcal{C}$ with $g_\ell = \Theta((\log N)^{-1})$ with the same ℓ, proving $\mathrm{LCS}(s,t) \geq (1/2 + \Omega((\log N)^{-2}))N$ using the Green strategy alone. Already, this argument saves several factors of $\log N$ in the surplus term over the argument of Bukh and Ma [15].

Strategy 3: Blue-Yellow strategy. We give a third and final strategy which handles the cases in which $g_\ell = o(1)$ for all oscillation periods ℓ, which we call the Blue-Yellow strategy. This strategy, handled in Section 3.7, is the most involved of the three and we do not explain all of the technical complications here. However, the general picture is similar to the Green strategy: we send two runners along s and t matching ones, and find opportune moments to switch to matching zeros to gain an advantage.

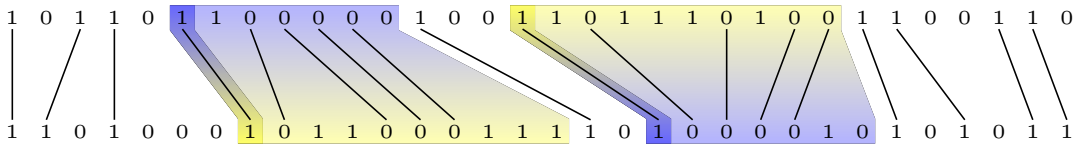


Figure 3.3: In the Blue case, we find a long common subsequence by matching zeros from extremely zero-rich patches (signalled by Blue flags) to longer relatively balanced patches (signalled by Yellow flags) and vice-versa. The Blue-Yellow strategy finds an LCS of length 18 between the s and t pictured above.

In the Blue-Yellow strategy, we also mark certain one-bits in s and t by flags, but we use flags of two different colors Blue and Yellow. A Blue ℓ-flag is a relatively rare occurrence: it signals that there is an extremely zero-rich interval afterwards containing $\ell - 1$ ones and more than $\varepsilon^{-1}(\ell - 1)$ zeros. A Yellow ℓ-flag, on the other hand, is very common: it signals there is an interval afterwards containing $\ell - 1$ ones and more than $0.9(\ell - 1)$ zeros. Also, since there is no single oscillation period that captures the behavior of s and t (or else we would apply the Green strategy), we must pay attention to flags at many different scales ℓ at the same time. The rough idea is then that the runners will switch to matching zeros when one of them reaches a Blue flag, and the other one simultaneously hits a Yellow flag of a similar scale, see Figure 3.3.

In the above diagram, the top runner reaches a Blue flag first, and the bottom runner happens upon a Yellow flag at the same time. This signals both of them to switch to matching zeros, which allows the top runner to gain a great advantage (since the patch past a Blue flag is so zero-rich). The bottom runner may lose out slightly in efficiency because Yellow intervals can have slightly fewer zeros than ones, but on net we see that more bits are used from the two patches together than would have been otherwise. In the long run, we expect Blue flags to appear approximately equally frequently in s and t, so the advantages and losses balance out to a net gain on both sides.

Now we explain roughly how the Blue-Yellow strategy circumvents the obstacle that the Green strategy ran into. The Green strategy by itself cannot prove Theorem 3.1, since there exists strings s such as $s = s_0 \dots s_{k-1}$ where s_i oscillates with period 2^i, so that s has only $O(N/\log N)$ Green ℓ-flags at any single scale ℓ. Thus, it is insufficient to consider flags at only a single scale. To take a concrete example, supposing $t = s_{\sigma(0)} \dots s_{\sigma(k-1)}$ for a typical permutation σ, the Green strategy fails to find an common subsequence of length more than $(1/2 + \Omega((\log N)^{-2}))N$ between s and t. In this example, instead of focusing on a single ℓ, we instead consider all one-bits in s and t that are the Blue ℓ-flags for some $\ell \geq N^{1-\varepsilon}$. This threshold $N^{1-\varepsilon}$

is chosen so that there are $\Theta(\varepsilon^2 N)$ such flags in each of s and t. Because these are the flags at the largest ε-fraction of scales, we see that for every single $\ell \geq N^{1-\varepsilon}$, most ℓ-intervals in s and t have density close to $1/2$, so most ℓ-flags in s and t are Yellow. As a result, we expect that as the two runners scan through s and t, most Blue ℓ-flags in one will be matched with a Yellow flag in the other, resulting in favorable situations as in Figure 3.3. The Blue-Yellow strategy succeeds by accumulating all these advantages across the $\Omega(\varepsilon N)$ Blue flags in each of s and t, to find an LCS of length $(1/2 + \Omega(1))N$.

String regularity. In order to guarantee that the two runners remain relatively synchronized as the Blue-Yellow strategy proceeds, we need Blue flags to be somewhat consistently distributed within s and within t. We get this desired property by proving a string regularity lemma similar to that of [5] (see also [16, 4, 78]). Our regularity lemma, proved in Section 3.5 differs from previous versions for words in that we use an *entropy increment* argument, rather than the usual density increment argument.

The structure lemma. The remainder of the chapter is designed to set up the strings s and t so that one of the above three strategies can succeed in finding a long common subsequence. To do this, we prove a structure lemma in Section 3.4 about strings, which says each string falls in one of three cases (1) Globally imbalanced, (2) Green at some oscillation period ℓ, or (3) Blue-Yellow at some oscillation period ℓ. These types are defined such that, if two strings s and t have the same type and same oscillation period (if applicable), one can find a long common subsequence of s and t using the corresponding strategy.

With this structure lemma, we simply need to put all the pieces together (Section 3.8). We partition each codeword s into $\operatorname{poly}\log N$ substrings $s_1, \ldots, s_{\operatorname{poly}\log N}$ and apply the structure lemma to each substring s_i. By pigeonhole, there exist two strings s and t such that, for each $i = 1, \ldots, \operatorname{poly}\log N$, the substrings s_i and t_i are the same "type," meaning that they are in the same case of the structure lemma and have the same oscillation period (if applicable). Then in each pair of substrings s_i and t_i one can find a long common subsequence using one of the three strategies, giving an overall large LCS.

It is important for two technical reasons to split into substrings s_i and t_i, rather than applying the structure lemma directly to the entire strings s and t. First, we need to randomly shift the starting position of one of the "runners" to get enough synchronized flags, incurring a loss of order up to $|s_1|$, so it is necessary that this loss is $o(N)$. Second, our regularity lemma guarantees Blue flags to have sufficient regularity at most, but not all, substrings lengths. Thus, we may not obtain the desired regularity until we consider substring lengths down to around $O(N/\operatorname{poly}\log N)$.

Organization. Section 3.3 collects the common notations, definitions, and preliminary lemmas we need for the rest of the chapter. In Section 3.4, we prove a structure lemma which divides strings into three types, one suitable to each of the three strategies above. In Section 3.5, we perform an additional technical argument to prove a "regularity-type" property of strings necessary for the runners to remain synchronized in the Blue-Yellow case (so that one does not race too far ahead of the other). These two sections together set the stage for Sections 3.6, 3.7, and 3.8 to handle the Green case, the Blue-Yellow case, and complete the proof, respectively.

3.3 Preliminaries and Notation

Constants. Throughout, fix $\varepsilon = 10^{-6}$ and $\gamma = 10^{-15} = 0.001\varepsilon^2$.

Indicators. For a boolean statement φ, let $\mathbb{I}[\varphi]$ be 1 if φ is true and 0 if φ is not true.

Strings. Throughout the remainder of this chapter, all strings are binary. Let $\{0,1\}^*$ be the set of all binary strings of any nonnegative length. For strings s_1 and s_2, let $s_1 s_2$ denote the string concatenation of s_1 with s_2.

A *subsequence* in a string s is any string obtained from s by deleting zero or more symbols. In contrast, a substring is a subsequence consisting of consecutive symbols of s (substrings are also sometimes referred to as *subwords* elsewhere, but we do not use this terminology). Thus, if $s = 10001$, then 101 is a subsequence of s but not a substring.

Intervals and sets. Throughout, for real numbers x and y we define an *interval* $I = [\![x, y]\!]$ to be the set of integers a such that $x \le a \le y$ (rather than the set of real numbers a). We similarly define intervals $(\!(x, y]\!]$ and $[\![x, y)\!)$ and $(\!(x, y)\!)$ as subsets of the integers. The *size* of an interval is the number of integers in the interval. For $\alpha \in (0,1)$ and real number x, let $(1 \pm \alpha)x$ denote the interval $[\![(1-\alpha)x, (1+\alpha)x]\!]$.

For integers $m \ge 0$ and $i \ge 1$, we let $I_{m,i} \stackrel{\text{def}}{=} [\![(i-1)\cdot 2^m + 1, i \cdot 2^m]\!]$. We call such an $I_{m,i}$, where both the size is a power of two and the endpoints are aligned with the same power of two, a *dyadic interval*.

For a string s with L ones and an interval $I = [\![x, y]\!]$, let s_I denote the contiguous substring of s between the x-th one of s (or the beginning of the string if $x \le 0$) and the $(y+1)$-st one of s (or until the end of the string if $y \ge L$), including the first but excluding the second. For example, if $w = 1001011$, we have $w_{[\![1,2]\!]} = 10010$. For $m \ge 0$ and $i \ge 1$ write $s_{m,i}$ as shorthand for $s_{I_{m,i}}$. Informally, we refer to $s_{m,1}, s_{m,2}, s_{m,3}, \ldots$ as the *substrings of s at scale m*. We note that leading zeros of a string are not included in any dyadic substring $s_{m,i}$, but this is negligible as we typically work with strings that start with a one.

We write $z(s)$ for the number of zeros in a string s.

Reversing strings that begin with a one. In the Blue-Yellow strategy on two strings s and t, we apply the Blue-Yellow matchings in pairs: (i) matching Blue flags in s with Yellow flags in t, and (ii) matching Blue flags in t with Yellow flags in s. Arguments (i) and (ii) can be see as applying the same lemma (Lemma 3.22) when (ii) is viewed as applying the lemma on the reversals of s and t. However, since throughout we index our substrings s_I by the one-bits rather than all bits, it is helpful to slightly modify the definition of string reversal as follows. Given a string s starting with a one, let $\mathrm{rev}(s)$ denote the string obtained by reversing the order of all the bits in s after the first bit. Thus, $\mathrm{rev}(s)$ is only defined for strings starting with a one. It is easy to check the following properties of rev.

Lemma 3.3. *Let w be a string that begins with a one and has L ones in total.*

1. *The strings w and $\mathrm{rev}(w)$ have the same length and number of ones.*
2. *For an interval $I = [\![x, y]\!] \subset [L]$, we have $\mathrm{rev}(w_I) = \mathrm{rev}(w)_{[\![L+1-y, L+1-x]\!]}$.*
3. *When w has $L = 2^n$ ones we have $\mathrm{rev}(w_{m,i}) = \mathrm{rev}(w)_{m, 2^{n-m}+1-i}$.*

Proof. The first item is obvious. For the second item, it suffices to consider when x and y are integers: indeed, for real numbers x and y, we have $[\![x, y]\!] = [\![\lceil x \rceil, \lfloor y \rfloor]\!]$ and $[\![L+1-y, L+1-x]\!] = [\![L+1-\lfloor y \rfloor, L+1-\lceil x \rceil]\!]$, so we may replace x and y with $\lceil x \rceil$ and $\lfloor y \rfloor$. The zeros between the i-th and $(i+1)$-st one of w map to the zeros between the $(L+1-i)$-th and $(L+2-i)$-th one of $\mathrm{rev}(s)$. Let z_i denote the number of zeros between

the i-th and $(i+1)$-st one of w. Then

$$\mathrm{rev}(w_{[\![x,y]\!]}) = \mathrm{rev}(10^{z_x}10^{z_{x+1}}1\cdots 10^{z_y}) = 10^{z_y}10^{z_{y-1}}1\cdots 10^{z_x} = \mathrm{rev}(w)_{[\![L+1-y,L+1-x]\!]}.$$

The third item follows from the second:

$$\begin{aligned}\mathrm{rev}(w_{m,i}) = \mathrm{rev}(w_{[\![(i-1)\cdot 2^m+1,i\cdot 2^m]\!]}) &= \mathrm{rev}(w)_{[\![2^n+1-i\cdot 2^m,2^n-(i-1)\cdot 2^m]\!]}\\ &= \mathrm{rev}(w)_{[\![2^m(2^{n-m}-i)+1,2^m\cdot(2^{n-m}-i+1)]\!]} = \mathrm{rev}(w)_{m,2^{n-m}-i+1}.\end{aligned}$$ □

Example 3.4. For $w = 1001011$, we have

$$\mathrm{rev}(w_{[\![1,2]\!]}) = \mathrm{rev}(10010) = 10100 = (1110100)_{[\![3,4]\!]} = \mathrm{rev}(w)_{[\![3,4]\!]}.$$

Lemma 3.5. *(*rev *preserves LCS) For strings s and t starting with a one,* $\mathrm{LCS}(s,t) = \mathrm{LCS}(\mathrm{rev}(s),\mathrm{rev}(t))$.

Proof. The LCS always matches the first bits if they are equal, and reversing strings preserves the LCS. □

Flags. We now define flags, a key notion that measures the oscillation frequencies within string.

Definition 3.6 (Flags)**.** For a positive integer ℓ and a string w, define an index $i \in \mathbb{Z}$ to be *an ℓ-flag of rate r in w* if $(\ell-1)^{-1}z(w_{[\![i,i+\ell]\!]}) = r$. Here $r \in [\![0,+\infty]\!]$ (so it can take on the value $+\infty$) and we define $0^{-1}\cdot 0 = 0$ and $0^{-1}\cdot m = +\infty$ for any positive integer m. The rate of an ℓ-flag i in w is the ratio of zeros to ones (strictly) between the i-th one and the $(i+\ell)$-th one of w, and we would like to find ℓ-flags with high rate in order to execute the zero-matching strategies described in the previous section. We say ℓ-flag i of rate r is

$$\begin{cases}\text{Blue} & \text{if } r > \varepsilon^{-1},\\ \text{Green} & \text{if } r > 1+2\varepsilon,\\ \text{Yellow} & \text{if } r > 0.9,\\ \text{Red} & \text{if } r \le 0.9.\end{cases}$$

Note that Blue flags are Green flags and Green flags are Yellow flags. For a string w with L ones, for each $i \in [L]$, define $b_w(i)$ to be the largest power of two $\ell \in [L]$ such that i is a Blue ℓ-flag in w, and 0 if no such ℓ exists. We say i is a Blue ℓ^+-flag in w if $b_w(i) \ge \ell$.

Imbalanced strings. For $\delta \in (0,1/2)$, we say a string w with L ones is *δ-imbalanced* if its number of zeros $z(w)$ is not in $(1\pm\delta)L$. For convenience, we simply say that w is *imbalanced* if it is ε-imbalanced with $\varepsilon = 10^{-6}$ defined at the beginning of this section. We will reason about imbalanced substrings w_I of w at various scales, and exploit the existence of these imbalanced substrings.

Lemma 3.7. *Let w be a string with L ones, and let $\ell \ge 2\varepsilon^{-1}$. Suppose that i is a Blue or Green ℓ-flag in w, or $i \le L-\ell+1$ is a Red ℓ-flag. On the interval $I = [\![i, \min(i+\ell-1, L)]\!]$, the substring w_I is imbalanced.*

Proof. If i is a Green ℓ-flag, substring w_I has at least $(1+2\varepsilon)(\ell-1) > (1+\varepsilon)\ell \ge (1+\varepsilon)|I|$ zeros. Since Blue flags are Green flags, w_I is also imbalanced if i is a Blue ℓ-flag. If i is a Red ℓ-flag, substring w_I has at most $0.9(\ell-1) < (1-\varepsilon)|I|$ zeros. In all three cases, w_I is imbalanced, as desired. □

It is also easy to see that if two imbalanced strings have the same length n and the same number of ones, then their LCS is a constant fraction larger than $n/2$.

Lemma 3.8. *Let $\delta \in (0, 1/2)$. Let s and t be δ-imbalanced strings of the same length with the same number L of ones in each. Then* $\mathrm{LCS}(s,t) \geq (1/2 + \delta/5)|s|$.

Proof. If the number of zeros is at most $(1-\delta)L$, then the all-ones string is a common subsequence of length $L \geq \frac{1}{2-\delta}|s| \geq (1/2+\delta/5)|s|$. If the number of zeros is at least $(1+\delta)L$, then the all-zeros string is a common subsequence of length at least $\frac{1+\delta}{2+\delta}|s| \geq (1/2+\delta/5)|s|$. □

Prefixes and suffixes. For a string w with L ones, and $\Delta \in [\![-L, L]\!]$, let $\mathrm{Trim}_\Delta(w) \stackrel{\text{def}}{=} w_{[\![\max(1,1-\Delta),\min(L,L-\Delta)]\!]}$. Thus, $\mathrm{Trim}_\Delta(w)$ is a prefix of w if $\Delta \geq 0$ and a suffix otherwise. The following lemma (see Figure 3.4) shows that finding long common subsequences across prefixes and suffixes of many subintervals of s and t implies that $\mathrm{LCS}(s,t)$ is large overall.

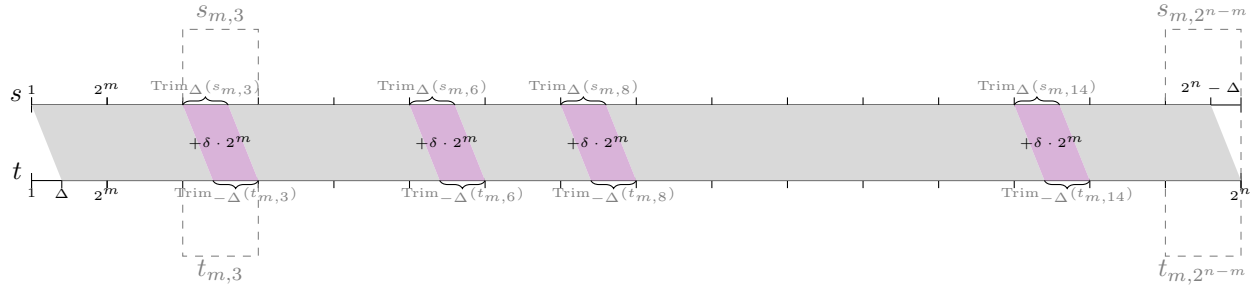


Figure 3.4: If, for some Δ, we find a large LCS between many (prefixes and suffixes of) dyadic substrings of s and t, then $\mathrm{LCS}(s,t)$ is large overall. We note that Lemma 3.9 works as long as the subintervals (in purple) have LCS beating the trivial matching by $\delta \cdot 2^m$ *on average*, even though the figure depicts *each* subinterval having LCS advantage $\delta \cdot 2^m$. In the diagram, the set Z from Lemma 3.9 is $\{3, 6, 8, 14\}$. [4]

Lemma 3.9 (Prefix/Suffix LCS)**.** *Let $\delta > 0$, let m and n be integers with $0 \leq m \leq n - 10 - \log \delta^{-1}$ and $L = 2^n$, and let $Z \subset [2^{n-m}]$ satisfy $|Z| \geq 2^{n-m}/10$. Suppose that s and t are strings with L ones each, and there exists $\Delta \in [\![-2^m, 2^m]\!]$ and $\delta > 0$ such that*

$$\sum_{i \in Z} \mathrm{LCS}(\mathrm{Trim}_\Delta(s_{m,i}), \mathrm{Trim}_{-\Delta}(t_{m,i})) \geq |Z| \cdot (2^m - |\Delta| + \delta \cdot 2^m).$$

Then we have

$$\mathrm{LCS}(s,t) \geq \left(1 + \frac{\delta}{20}\right) L.$$

Proof. Finding a common sequence between s and t is equivalent to exhibiting a matching between the bits of s and the bits of t such that only equal bits are matched and such that the matching is "non-crossing," meaning that earlier bits of s are matched with earlier bits of t.

Consider the matching where we match the i-th one in s with the $(i+\Delta)$-th one in t. This matches $L - |\Delta|$ ones. In this matching, for each $i \in Z$, the $2^m - |\Delta|$ ones of $\mathrm{Trim}_\Delta(s_{m,i})$ are exactly matched to the

[4] Technically, the figure is invalid because we need $m \leq n - 10 - \log \delta^{-1}$ and here $m = n - 4$, but we ignore this issue for illustration.

$2^m - |\Delta|$ ones of $\mathrm{Trim}_{-\Delta}(t_{m,i})$. For each $i \in Z$, replace the matching between the ones of $\mathrm{Trim}_{\Delta}(s_{m,i})$ and $\mathrm{Trim}_{-\Delta}(t_{m,i})$ with a matching for the LCS of $\mathrm{Trim}_{\Delta}(s_{m,i})$ and $\mathrm{Trim}_{-\Delta}(t_{m,i})$.

All of these replacements can be done simultaneously and independently while keeping the matching non-crossing. Each of the $|Z|$ replacement operations deletes $2^m - |\Delta|$ pairs, and in total the replacements add $|Z|(2^m - |\Delta| + \delta \cdot 2^m)$ pairs. Thus in total the replacements increase the number of matched pairs by at least $|Z| \cdot \delta \cdot 2^m$. Thus, the total length of this common subsequence is at least (recall $L = 2^n$)

$$L - |\Delta| + |Z| \cdot (\delta 2^m) \geq L - 2^m + \frac{2^{n-m}}{10} \cdot \delta \cdot 2^m \geq L - \frac{\delta}{2^{10}} L + \frac{\delta}{10} L \geq \left(1 + \frac{\delta}{20}\right) L.$$

In the second inequality, we used that $m \leq n - 10 - \log \delta^{-1}$. Thus $\mathrm{LCS}(s,t) \geq \left(1 + \frac{\delta}{20}\right) L$, as desired. □

3.4 The structure lemma and definition of types

3.4.1 The structure lemma

Throughout this section, we reason about a single string w with L ones, and assume w starts with a one. Recall that $\varepsilon = 10^{-6}$. Our main structure lemma is as follows.

Lemma 3.10 (Structure Lemma)**.** *If $w \in \{0,1\}^*$ is a string that starts with one and has exactly $L = 2^n$ ones, and n is sufficiently large (in terms of ε), then at least one of the following conditions hold.*

1. *There exists an interval $I \subseteq [L]$ of size $|I| \geq \varepsilon^2 L$ such that w_I is imbalanced.*
2. *There exists $1 \leq \ell \leq L$ such that the number of Green ℓ-flags in w is at least $\varepsilon^2 L$.*
3. *There exists $1 \leq m \leq n$ such that the number of Blue $(2^m)^+$-flags in w is at least $\varepsilon^2 L$, and for every $\ell \geq 2^m$ the number of Red ℓ-flags in w is at most $600\varepsilon L$.*

The three cases of the Structure Lemma correspond exactly to the three matching strategies outlined in the Overview (Section 3.2). Case 1 is when w is imbalanced at a macroscopic scale, i.e., a linear-length subword with density far from $\frac{1}{2}$. Case 2, the "Green case", is when w "fluctuates on a single scale" and can be treated by studying that scale only, i.e., using the Green strategy described in the Overview. Case 3 is when w is "sporadic" and must be analyzed at many scales simultaneously, which is done with the Blue-Yellow strategy in the Overview. Before we prove Lemma 3.10, we need the following technical lemma, which gives part of Lemma 3.10 under the additional assumption that many of the zeros in w are concentrated at Blue flags.

Lemma 3.11. *Suppose $\alpha > 0$, n is sufficiently large, $L = 2^n$, $1 \leq \ell \leq L$, and $w \in \{0,1\}^*$ is a string with L ones. If*

$$\frac{1}{\ell} \sum_{i \in B_\ell} z(w_{[\![i, i+\ell]\!)}) \geq \alpha L,$$

where B_ℓ is the set of all Blue ℓ-flags of w, then at least one of the following conditions hold.

1. *There exists an interval $I \subseteq [L]$ of size $|I| \geq \varepsilon^2 L$ such that w_I is imbalanced.*
2. *The number of Blue ℓ-flags in w is at least $\varepsilon^2 L$.*

3. The number of Blue ℓ^+-flags in w is at least $(\alpha - 22\varepsilon)\varepsilon L/16$.

Proof. We assume Conditions 1 and 2 do not hold and prove Condition 3. We first prove a slightly stronger lower bound (see (3.4)) on the number of Blue ℓ^+ flags than claimed in Condition 3 in the special case that ℓ is a power of 2, and will then handle the case of general ℓ.

Since $|B_\ell| < \varepsilon^2 L$, if we define $A_\ell \stackrel{\text{def}}{=} \{i \in B_\ell \mid z(w_{[\![i,i+\ell)\!]}) > 2\varepsilon^{-1}\ell\}$ we find that

$$\frac{1}{\ell}\sum_{i\in B_\ell\setminus A_\ell} z(w_{[\![i,i+\ell)\!]}) \le \frac{1}{\ell}\cdot(2\varepsilon^{-1}\ell)\cdot|B_\ell\setminus A_\ell| \le 2\varepsilon L, \quad \text{so} \quad \frac{1}{\ell}\sum_{i\in A_\ell} z(w_{[\![i,i+\ell)\!]}) \ge (\alpha-2\varepsilon)L\ .$$

By the pigeonhole principle, we can pick a residue class $r \in \{1,\dots,\ell\}$ such that the set $S \subseteq [L/\ell]$ defined by $S \stackrel{\text{def}}{=} \{j \mid (j-1)\ell + r \in A_\ell\}$ satisfies

$$\sum_{j\in S} z(w_{[\![(j-1)\ell+r,j\ell+r)\!]}) \ge (\alpha-2\varepsilon)L.$$

Write $b_j \stackrel{\text{def}}{=} z(w_{[\![(j-1)\ell+r,j\ell+r)\!]})$. We have that for each $j \in S$, $b_j > 2\varepsilon^{-1}\ell$, and $b_S \ge (\alpha-2\varepsilon)L$ (using the summation notation $b_S \stackrel{\text{def}}{=} \sum_{j\in S} b_j$).

Let $L' = L/\ell$. Since ℓ and L are powers of 2, L' is as well. We consider the family of all dyadic intervals $I_{m,i} \stackrel{\text{def}}{=} [\![(i-1)\cdot 2^m+1, i\cdot 2^m]\!]$ where $0 \le m \le \log_2(L')$ and $1 \le i \le L'\cdot 2^{-m}$. Let $\mathcal{I}$ be the set of such intervals $I_{m,i}$ maximal under the property that $b_{I_{m,i}} > 2\varepsilon^{-1}\ell\cdot|I_{m,i}|$. By maximality of the $I_{m,i}$, the intervals of $\mathcal{I}$ are pairwise disjoint. Furthermore, for $j \in S$, we have $b_{I_{0,j}} = b_j > 2\varepsilon^{-1}\ell$, so each $j \in S$ is in some interval of $\mathcal{I}$.

We claim that these intervals satisfy $b_{I_{m,i}} \le 4\varepsilon^{-1}\ell\cdot|I_{m,i}|$. Suppose otherwise. Either $I_{m,i} = [L']$ or there is a dyadic interval $I_{m+1,\lceil i/2\rceil}$ in $[L']$ containing $I_{m,i}$ with twice the size of $I_{m,i}$. In the former case, $z(w_{[L]}) = b_{[L']} > 4\varepsilon^{-1}\ell L' > L + \varepsilon L$, which would imply condition 1. In the latter case, $I_{m+1,\lceil i/2\rceil}$ is an interval containing $I_{m,i}$ satisfying $b_{I_{m+1,\lceil i/2\rceil}} > 2\varepsilon^{-1}\ell\cdot|I_{m+1,\lceil i/2\rceil}|$, contradicting the maximality of $I_{m,i}$. This proves the claim.

Since every $i \in S$ lies in some element of $\mathcal{I}$, we have

$$4\varepsilon^{-1}\ell\cdot\sum_{I_{m,i}\in\mathcal{I}}|I_{m,i}| \ge \sum_{I_{m,i}\in\mathcal{I}} b_{I_{m,i}} \ge b_S \ge (\alpha-2\varepsilon)L,$$

and so $\sum_{\mathcal{I}}|I_{m,i}| \ge \frac{1}{4}(\alpha-2\varepsilon)\varepsilon L/\ell$.

Thus the dyadic intervals in $\mathcal{I}$ have an abundance of zeroes in total. We need to convert this into an abundance of ℓ^+-flags. To this end, let us define $J_{m,i} \stackrel{\text{def}}{=} [\![(i-1)\cdot 2^m\ell + r, i\cdot 2^m\ell + r)\!]$, defined so that $b_{I_{m,i}} = z(w_{J_{m,i}})$. The key observation is that if $I_{m,i} \in \mathcal{I}$, then any $j \in J_{m,i-1}$ is a Blue ℓ^+-flag. Indeed, for such a j we have $[\![j, j+2^{m+1}\ell)\!] \supseteq J_{m,i}$, so

$$z(w_{[\![j,j+2^{m+1}\ell)\!]}) \ge z(w_{J_{m,i}}) = b_{I_{m,i}} > 2\varepsilon^{-1}\ell\cdot|I_{m,i}| = \varepsilon^{-1}\cdot 2^{m+1}\ell,$$

so j is a Blue $(2^{m+1}\ell)$-flag, and thus an ℓ^+-flag as desired since ℓ is a power of 2.

We conclude that all of the elements of

$$T \stackrel{\text{def}}{=} [L] \cap \bigcup_{I_{m,i}\in\mathcal{I}} J_{m,i-1} \tag{3.3}$$

are Blue ℓ^+-flags of w.

Note that the union in (3.3) may not be a disjoint union. Indeed, although any two intervals $I_{m_1,i_1}, I_{m_2,i_2} \in \mathcal{I}$ must be disjoint, their predecessor intervals $I_{m_1,i_1-1}, I_{m_2,i_2-1}$ may not be disjoint if $m_1 \neq m_2$ (and the corresponding $J_{m,i}$ may not be either). To address this issue, we pass to a subcollection $\mathcal{I}' \subseteq \mathcal{I}$ so that for any two $I_{m_1,i_1}, I_{m_2,i_2} \in \mathcal{I}'$, their respective shifts I_{m_1,i_1-1} and I_{m_2,i_2-1} are disjoint. Notice that for disjoint $I_{m_1,i_1}, I_{m_2,i_2} \in \mathcal{I}$ with $m_1 \leq m_2$, their shifts I_{m_1,i_1-1} and I_{m_2,i_2-1} intersect only if $I_{m_1,i_1} \subsetneq I_{m_2,i_2-1}$. Thus, it suffices to find a subcollection $\mathcal{I}' \subseteq \mathcal{I}$ so that no interval I_{m_1,i_1} lies in the shift I_{m_2,i_2-1} of another. Such a subcollection can be picked greedily by scanning from right to left through $\mathcal{I}$, skipping any I_{m_1,i_1} which appears in the shift I_{m_2,i_2-1} of an interval already picked, and breaking ties by preferring larger intervals. Any greedily chosen I_{m_2,i_2} removes at most $|I_{m_2,i_2}|$ in total interval size from $\mathcal{I}'$, so the total size of the intervals in $\mathcal{I}'$ is at least half of the total size of intervals in $\mathcal{I}$.

Thus, $\sum_{\mathcal{I}'} |I_{m,i}| \geq \frac{1}{2} \sum_{\mathcal{I}} |I_{m,i}|$, and $\{J_{m,i-1} \mid I_{m,i} \in \mathcal{I}'\}$ is now a disjoint collection. Inside $\mathcal{I}'$, there can be at most one interval $I_{m,i}$ for which $J_{m,i-1}$ does not lie entirely inside $[L]$, and $J_{m,i-1}$ must have size at most $\varepsilon^2 L$ or else

$$z(w_{J_{m,i-1}}) > (1+\varepsilon) \cdot 2^m,$$

for $2^m \geq \varepsilon^2 L$, implying condition 1. It follows that w has at least

$$|T| \geq \sum_{I_{m,i} \in \mathcal{I}'} |J_{m,i-1}| - \varepsilon^2 L \geq \frac{1}{2} \sum_{I_{m,i} \in \mathcal{I}} |J_{m,i-1}| - \varepsilon^2 L \geq \frac{(\alpha - 10\varepsilon)\varepsilon}{8} L, \tag{3.4}$$

Blue ℓ^+-flags. This proves the lemma, in fact with a stronger lower bound on number of Blue ℓ^+ flags, when ℓ is a power of 2.

Now suppose ℓ is not a power of 2. If ℓ' is the smallest power of 2 greater than or equal to ℓ, then all Blue $(\ell')^+$-flags are also Blue ℓ^+-flags and $1 \leq \ell' \leq L$ since L is a power of 2. Furthermore, if $B_{\ell'}$ is the set of Blue ℓ'-flags, $z(w_{[\![i,i+\ell)\!]}) \leq z(w_{[\![i,i+\ell')\!]}) \leq \varepsilon^{-1}(\ell' - 1)$ if $i \notin B_{\ell'}$. Thus, since $|B_\ell| < \varepsilon^2 L$ we get

$$\frac{1}{\ell'} \sum_{i \in B_\ell \setminus B_{\ell'}} z(w_{[\![i,i+\ell)\!]}) \leq \varepsilon L,$$

which implies

$$\frac{1}{\ell'} \sum_{i \in B_{\ell'}} z(w_{[\![i,i+\ell')\!]}) \geq \frac{1}{\ell'} \sum_{i \in B_\ell \cap B_{\ell'}} z(w_{[\![i,i+\ell)\!]}) \geq \frac{1}{2\ell} \sum_{i \in B_\ell} z(w_{[\![i,i+\ell)\!]}) - \varepsilon L \geq (\alpha/2 - \varepsilon) L,$$

and so the conditions of the lemma are true with modified parameters $\ell' \geq \ell$ a power of 2 and $\alpha' = \alpha/2 - \varepsilon$. It follows by applying (3.4) with these parameters instead that w must have at least

$$\frac{(\alpha' - 10\varepsilon)\varepsilon}{8} L \geq \frac{(\alpha - 22\varepsilon)\varepsilon}{16} L$$

Blue ℓ^+-flags in the general case, thus completing the proof. □

Now we are ready to prove Lemma 3.10.

Proof of Lemma 3.10. We assume conditions 1 and 2 do not hold for some w, and prove condition 3.

Pick $m \in [\![0, n]\!]$ maximal such that w contains at least $\varepsilon^2 L$ Blue $(2^m)^+$-flags. To see that such an m exists, let B_1 denote the set of Blue 1-flags in w, which is just the set of one-bits in w immediately followed by at least one zero. Then, we have

$$\sum_{i \in B_1} z(w_{[\![i,i+1]\!)}) = z(w_{[L]}) \geq L - \varepsilon L,$$

assuming condition 1 does not hold. Furthermore, because Blue flags are Green flags, condition 2 being false implies there are fewer than $\varepsilon^2 L$ Blue 2^m-flags for any $0 \leq m \leq n$. In particular, $|B_1| < \varepsilon^2 L$. Thus, the conditions of Lemma 3.11 are satisfied with $\ell = 1$ and $\alpha = 1 - \varepsilon$, and we obtain that either w satisfies condition 1 or condition 2 (since Blue flags are Green flags) or the number of Blue 1^+-flags in w is at least

$$(\alpha - 22\varepsilon)\varepsilon L/16 = (\varepsilon - 23\varepsilon^2)L/16 \geq 2\varepsilon^2 L.$$

Hence we used that ε is sufficiently small. Thus, some such $0 \leq m \leq n$ exists.

By the maximality of m and the fact that there are fewer than $\varepsilon^2 L$ Blue 2^m-flags for any particular $0 \leq m \leq n$, we see that in fact $m \geq 1$ and the number of Blue $(2^m)^+$-flags is in $[\![\varepsilon^2 L, 2\varepsilon^2 L]\!)$. It remains to check that the second half of condition 3 holds.

Suppose $\ell \geq 2^m$, and let B_ℓ, G_ℓ, Y_ℓ, and R_ℓ be the sets of ℓ-flags in w which are Blue, Green, Yellow, and Red (respectively). Note that by our definitions of the colors, $B_\ell \subseteq G_\ell \subseteq Y_\ell$ and $Y_\ell \sqcup R_\ell = [L]$. Suppose for the sake of contradiction that $|R_\ell| > 600\varepsilon L$. We may assume $\ell \leq \varepsilon^2 L$, as otherwise a single Red ℓ-flag would violate condition 1. We can express $z(w_{[L]})$ as

$$z(w_{[L]}) = \frac{1}{\ell} \sum_{i=-\ell}^{L} z(w_{[\![i,i+\ell]\!)})$$

since each $z(w_{\{i\}})$ appears exactly ℓ times in the sum on the right. Setting aside the terms on the right with $i \leq 0$, we find by breaking up the sum in terms of the colors of the flags,

$$z(w_{[L]}) \leq \frac{1}{\ell} \left(\sum_{i=-\ell}^{0} z(w_{[\![i,i+\ell]\!)}) + \sum_{i \in B_\ell} z(w_{[\![i,i+\ell]\!)}) + (\ell - 1)(\varepsilon^{-1}|G_\ell \backslash B_\ell| + (1 + 2\varepsilon)|Y_\ell \backslash G_\ell| + 0.9|R_\ell|) \right).$$

We can bound $\frac{1}{\ell} \sum_{i=-\ell}^{0} z(w_{[\![i,i+\ell]\!)}) \leq z(w_{[\ell]})$, $|G_\ell \backslash B_\ell| \leq |G_\ell| < \varepsilon^2 L$ (since condition 2 is false), and $|Y_\ell \backslash G_\ell| \leq |Y_\ell| = L - |R_\ell|$. Putting these together with (3.4.1), we obtain

$$z(w_{[L]}) \leq L + 3\varepsilon L + z(w_{[\ell]}) + \frac{1}{\ell} \sum_{i \in B_\ell} z(w_{[\![i,i+\ell]\!)}) - 0.1|R_\ell|.$$

Since condition 1 is false, we have $z(w_{[L]}) \geq L - \varepsilon L$, so together with (3.4.1) and $|R_\ell| > 600\varepsilon L$ we get

$$z(w_{[\ell]}) + \frac{1}{\ell} \sum_{i \in B_\ell} z(w_{[\![i,i+\ell]\!)}) \geq 56\varepsilon L.$$

Next, we claim that $z(w_{[\ell]}) \leq \varepsilon L$. This follows from the facts that $\ell \leq \varepsilon^2 L$ and $z(w_{[\varepsilon^2 L]}) \leq \varepsilon L$ (by the

assumption that condition 1 does not hold). We get

$$\frac{1}{\ell}\sum_{i\in B_\ell} z(w_{[\![i,i+\ell)\!]}) \geq 55\varepsilon L.$$

The conditions of Lemma 3.11 are satisfied with this ℓ and $\alpha = 55\varepsilon$. Applying the lemma, we find that either w satisfies one of condition 1 or 2, or the number of Blue ℓ^+-flags in w is at least $(55\varepsilon - 22\varepsilon)\varepsilon L/16 > 2\varepsilon^2 L$. This contradicts the observation we made before that the number of Blue $(2^m)^+$-flags is in $[\![\varepsilon^2 L, 2\varepsilon^2 L)\!]$ and completes the proof. □

3.4.2 Definition of types

Using the structure theorem, we define below the *type* of a string roughly based on the case that it satisfies in the structure theorem. These three definitions of types (Imbalanced, Green, Blue-Yellow) roughly align with the three cases of the structure theorem, though for Green and Blue-Yellow types, it helps to additionally have an upper bound on ℓ, the length of the flags. Because we only prove the Structure Lemma (Lemma 3.10) for strings whose number of ones is a power of two, we also only define types for strings whose number of ones is a power of two, which is enough for our application. Recall that $\gamma = 0.001\varepsilon^2$.

Definition 3.12. Given a string w with $L = 2^n$ ones with n sufficiently large, we say the *type of* w is

1. **Imbalanced** if there exists some interval $I \subseteq [L]$ of size $|I| \geq \varepsilon^5 L$ such that w_I is imbalanced.
2. **ℓ-Green** for some integer $1 \leq \ell \leq \varepsilon^5 L$ if the number of Green ℓ-flags in w is at least $\varepsilon^2 L$.
3. **m-Blue-Yellow** if there exists $1 \leq m \leq n$ such that the number of indices $i \in [L]$ with $2^m \leq b_w(i) \leq \gamma L$ is at least $(\varepsilon^2 - \gamma)L$, and the number of Red ℓ-flags in w is at most $600\varepsilon L$ for any $\ell \geq 2^m$.

If w could be more than one type, we assign w one of the possible types arbitrarily.

Note that there are at most $1 + L + \log L = O(|w|)$ possible types for a string w. As a corollary of Lemma 3.10, each string w has a type, assuming n is sufficiently large:

Lemma 3.13. *If n is sufficiently large, each string w with $L = 2^n$ ones has a type.*

Proof. If w satisfies Case 1 of Lemma 3.10, then w has type Imbalanced.

If w satisfies Case 2 of Lemma 3.10 with parameter ℓ, then w has at least $\varepsilon^2 L$ Green ℓ-flags, and in particular there exists an $i \leq L - \varepsilon^2 L$ that is a Green ℓ-flag. If $\ell \geq \varepsilon^5 L$, then by Lemma 3.7, the interval $I = [\![i, i+\ell-1]\!]$ of size at least $\varepsilon^5 L$ gives that w_I is imbalanced, so w has type Imbalanced. If $\ell \leq \varepsilon^5 L$, then w has type ℓ-Green.

If w satisfies Case 3 of Lemma 3.10 with parameter m, then there are at least $\varepsilon^2 L$ indices with $b_w(i) \geq \varepsilon^2 L$, and the number of Red ℓ-flags in w is at most $600\varepsilon L$ for any integer $\ell \geq 2^m$. If there are at least γL indices $i \in [L]$ with $b_w(i) \geq \gamma L$, then some $i \leq L - \gamma L + 1$ satisfies $b_w(i) \geq \gamma L$, so by Lemma 3.7, there exists an interval I of size at least $\gamma L > \varepsilon^5 L$ such that w_I is imbalanced, so w is type Imbalanced. Otherwise, for at least $(\varepsilon^2 - \gamma)L$ indices $i \in [L]$, we have $2^m \leq b_w(i) \leq \gamma L$, so w is type m-Blue-Yellow. □

3.5 The entropy regularity argument

In this section, we prove the regularity-type result Lemma 3.19, which roughly states that in most dyadic substrings of a given string s, the positions of the Blue flags are distributed relatively uniformly. In the Blue-Yellow strategy, we may be matching bits in s and t that lie in nearby but different dyadic intervals (because of our random shifting argument and because the Blue-Yellow strategy consumes ones from s and t at different rates). Because of this, it is helpful to say that there many neighboring pairs of dyadic intervals with similar Blue flag distributions in s and t.

3.5.1 Flag balance

Define the L^1 distance between two discrete probability distributions p, q to be $\|p-q\|_1 \stackrel{\text{def}}{=} \sum |p(x)-q(x)|$. Recall that $b_w(i)$ is the largest power of two $\ell \in [L]$ such that i is a Blue ℓ-flag in w, and 0 if no such ℓ exists. For a string w with L ones and an interval $I \subset [L]$, let $p_{w,I}$ denote the distribution of the value of $b_w(i)$ over a uniform random $i \in I$. Put another way, the probability mass $p_{w,I}(\ell)$ is the fraction of indices $i \in I$ with $b_w(i) = \ell$.

Definition 3.14 (Blue-flag-balance)**.** For $\beta > 0$, we say that a dyadic interval $I_{m,i}$ is *β-Blue-flag-balanced in w* if $\|p_{w,I_{m-1,2i-1}} - p_{w,I_{m-1,2i}}\|_1 \le \beta$. We say that a string w with $L = 2^n$ ones is *β-Blue-flag-balanced* if the interval $I_{n,1}$ is β-Blue-flag-balanced in w.

Showing Blue-flag balance is useful because we can show that if a Blue-flag-balanced string w has many Blue flags of a certain length, then both halves of w also have many Blue flags of the same length. The next lemma formalizes this idea.

Lemma 3.15. *If string w is β-Blue-flag-balanced with $L = 2^n$ ones, then for any set S of integers, we have*

$$\left| \mathop{\mathbf{Pr}}_{i\in[L]}[b_w(i) \in S] - \mathop{\mathbf{Pr}}_{i\in[L/2]}[b_w(i) \in S] \right| \le \frac{\beta}{2}.$$

Proof. Since w is β-Blue-flag-balanced, interval $I_{n,1}$ is β-Blue-flag-balanced in w. We have

$$\begin{aligned}
\left| \mathop{\mathbf{Pr}}_{i\in[L]}[b_w(i) \in S] - \mathop{\mathbf{Pr}}_{i\in[L/2]}[b_w(i) \in S] \right| &= \sum_{\ell' \in S} |p_{w,I_{n,1}}(\ell') - p_{w,I_{n-1,1}}(\ell')| \\
&\le \sum_{\ell'} |p_{w,I_{n,1}}(\ell') - p_{w,I_{n-1,1}}(\ell')| \\
&= \left\| p_{w,I_{n,1}} - p_{w,I_{n-1,1}} \right\|_1 \\
&= \left\| \frac{p_{w,I_{n-1,1}} - p_{w,I_{n-1,2}}}{2} \right\|_1 \le \frac{\beta}{2}.
\end{aligned}$$

The third equality uses that $\frac{p_{w,I_{n-1,1}} + p_{w,I_{n-1,2}}}{2} = p_{w,I_{n,1}}$, and the last inequality uses the definition of interval $I_{n,1}$ being β-Blue-flag-balanced in w. □

3.5.2 Flag balance of intervals

Our goal is to find a scale at which most intervals are Blue-flag-balanced. We start by proving a simple lemma about probability distributions. Recall that the binary entropy of a discrete probability distribution

p is defined as $H(p) \stackrel{\text{def}}{=} -\sum_x p(x) \log p(x)$ over all values x in the support of p, and the logarithms are base 2.

Lemma 3.16. *If p^-, p, and p^+ are three discrete probability distributions supported on a finite domain Ω satisfying $p^-(x) + p^+(x) = 2p(x)$ for all $x \in \Omega$, then*

$$H(p^-) + H(p^+) \le 2H(p) - \frac{1}{4}\|p^+ - p^-\|_1^2.$$

Proof. We use Pinsker's inequality, which states in the case of discrete probability distributions that the Kullback-Leibler divergence between two distributions P and Q satisfies

$$D_{\mathrm{KL}}(P\|Q) = \sum_i P(i) \log\left(\frac{P(i)}{Q(i)}\right) \ge \frac{1}{2}\|P - Q\|_1^2.$$

In particular, applying this to the pairs $(P, Q) = (p^-, p)$ and $(P, Q) = (p^+, p)$, we obtain

$$\begin{aligned} 2H(p) - H(p^-) - H(p^+) &= \sum_i \left(p^-(i) \log p^-(i) + p^+(i) \log p^+(i) - 2p(i) \log(p(i))\right) \\ &= D_{\mathrm{KL}}(p^-\|p) + D_{\mathrm{KL}}(p^+\|p) \\ &\ge \frac{1}{2}\|p^- - p\|_1^2 + \frac{1}{2}\|p^+ - p\|_1^2 \\ &= \frac{1}{4}\|p^+ - p^-\|_1^2. \end{aligned}$$

□

We now obtain with a regularity argument the following lemma, which is the most substantial result in this section.

Lemma 3.17 (Interval Blue-flag-balance)**.** *For $\beta > 0$, $n \ge 2$, and any $w \in \{0,1\}^N$ with $L = 2^n$ ones, except for at most $32\beta^{-3} \log n$ values of $m \in [\![0, n]\!]$, the following holds: the number of dyadic intervals $I_{m,i}$ that are not β-Blue-flag-balanced in w is less than $\beta \cdot 2^{n-m}$.*

Proof. Consider the expression

$$E_m \stackrel{\text{def}}{=} 2^m \cdot \sum_{i=1}^{2^{n-m}} H(p_{w,I_{m,i}}).$$

By the definition of an β-Blue-flag-balanced interval, we obtain that whenever interval $I_{m,i}$ is not β-Blue-flag-balanced in w,

$$\|p_{w,I_{m-1,2i-1}} - p_{w,I_{m-1,2i}}\|_1 \ge \beta,$$

and so if there are t_m dyadic intervals $I_{m,i}$ that are not β-Blue-flag-balanced in w, we obtain

$$E_{m-1} \le E_m - 2^{m-1} \cdot t_m \cdot \frac{1}{4}\beta^2 \le E_m - 2^{m-3} t_m \beta^2,$$

by Lemma 3.16. Since $b_w(i) \le L$ is either 0 or a power of two for all indices $i \in [L]$, we have that $b_w(i)$ takes on one of $n + 2$ values. Thus, we have $E_n = 2^n \cdot H(p_{w,I_{n,1}}) \le \log(n + 2) \cdot 2^n$. Since we also have $E_0 \ge 0$, at most $8\beta^{-3} \log(n + 2) < 32\beta^{-3} \log n$ values of t_m are at least $\beta \cdot 2^{n-m}$, completing the proof. □

3.5.3 Flag balance of substrings

In Lemma 3.17 we showed that there exists many scales m where many (dyadic) *intervals* were β-Blue-flag-balanced in the sense of Definition 3.14. For technical reasons, it is helpful to establish that there are many *substrings* that are β-Blue-flag-balanced, and we do so in this section. The distinction here is that certain one-bits in the interval I may be Blue ℓ-flags in w, but not Blue ℓ-flags in w_I once the zeros to the right of I are taken out of consideration.

For a string w with L ones, let $p_w \stackrel{\text{def}}{=} p_{w,[L]}$. Note that the distribution $p_{w,I}$ may be different from the distribution p_{w_I}, because indices $i \in I$ that are Blue ℓ-flags in string w may not correspond to Blue ℓ-flags in substring w_I. However, the converse is true: for $I = [\![x, y]\!]$, if $i - x + 1$ is a Blue ℓ-flags of substring w_I, then i is a Blue ℓ-flag of string w. Because of this, we can show that $p_{w,I}$ and p_{w_I} are similar in distribution under certain conditions.

Lemma 3.18. *Let $\beta > 0$, w be a string with L ones, $I \subset [L]$ be an interval, and ℓ_0 be an integer, such that at most $\beta|I|$ indices $i \in I$ satisfy $b_w(i) \geq \ell_0$. Then*

$$\|p_{w,I} - p_{w_I}\|_1 \leq 2\left(\beta + \frac{\ell_0}{|I|}\right).$$

Proof. Recall that $b_w(i)$ is the largest power of two such that the i-th one of w is an $b_w(i)$-flag of w (or 0 if no such power of two exists). For an interval $I = [\![x, y]\!]$, if index $i - x + 1$ is a Blue ℓ-flag of w_I, then index i is a Blue ℓ-flag of w, and furthermore if index i is a Blue ℓ-flag of w and $x \leq i \leq y - \ell$, then index $i - x + 1$ is a Blue ℓ-flag of w_I as well. Hence, for all $x \leq i \leq y - \ell_0$ with $b_w(i) \leq \ell_0$, we have that $b_w(i) = b_{w_I}(i - x + 1)$. Thus, by the union bound,

$$\mathop{\mathbf{Pr}}_{i \in I}[b_w(i) \neq b_{w_I}(i - x + 1)] \leq \mathop{\mathbf{Pr}}_{i \in I}[i \geq y - \ell_0] + \mathop{\mathbf{Pr}}_{i \in I}[b_w(i) \geq \ell_0] \leq \frac{\ell_0}{|I|} + \beta, \tag{3.5}$$

We also have

$$\|p_{w,I} - p_{w_I}\|_1 = \sum_{\ell'} |p_{w,I}(\ell') - p_{w_I}(\ell')| = \sum_{\ell'} \frac{1}{|I|} \left| \sum_{i \in I} \mathbb{I}(b_w(i) = \ell') - \sum_{i \in I} \mathbb{I}(b_{w_I}(i - x + 1) = \ell') \right|$$

Thus, the expression $\|p_{w,I} - p_{w_I}\|_1$ is a $\frac{1}{|I|}$-Lipschitz function in the indicator functions $\mathbb{I}(b_w(i) = \ell')$ for $i \in I$ and ℓ' a power of two or 0. Changing the value of any single $b_w(i)$ changes the value of at most two such indicator functions, and furthermore we know that changing the value of $|I| \cdot \mathbf{Pr}_{i \in I}[b_w(i) \neq b_{w_I}(i - x + 1)]$ values of $b_w(i)$ makes the expression $\|p_{w,I} - p_{w_I}\|_1$ equal to 0, so we have that

$$\|p_{w,I} - p_{w_I}\|_1 \leq \frac{1}{|I|} \cdot 2 \cdot |I| \cdot \mathop{\mathbf{Pr}}_{i \in I}[b_w(i) \neq b_{w_I}(i - x + 1)]. \leq 2 \cdot \mathop{\mathbf{Pr}}_{i \in I}[b_w(i) \neq b_{w_I}(i - x + 1)] \tag{3.6}$$

Combining (3.5) and (3.6) gives the desired result. □

Lemma 3.17 argues about the Blue-flag-balance of intervals. Combining Lemma 3.17 with Lemma 3.18, we obtain the following result about the Blue-flag-balance of substrings.

Lemma 3.19 (Substring Blue-flag-balance)**.** *Let $\beta > 0$, and n be sufficiently large in terms of β. Let $w \in \{0,1\}^N$ be a string with $L = 2^n$ ones, and suppose that at most $\beta^2 L$ indices $i \in [L]$ satisfy $b_w(i) \geq$*

$2^{n-200\beta^{-3}\log n}$. *Then, except for at most* $32\beta^{-3}\log n$ *values of* $m \in [\![n-150\beta^{-3}\log n, n]\!]$ *the following holds: the number of* $i \in [2^{n-m}]$ *for which* $w_{m,i}$ *is not* 6β*-Blue-flag-balanced is less than* $3\beta \cdot 2^{n-m}$.

Proof. Let $m \in [\![n-150\beta^{-3}\log n, n]\!]$ be any value such that there are at most $\beta \cdot 2^{n-m}$ dyadic intervals $I_{m,i}$ that are not β-Blue-flag-balanced in w. By Lemma 3.17, all but $32\beta^{-3}\log n$ values of m have this property. We show that each such m satisfy the requirements of the lemma.

Call an index $i \in [2^{n-m}]$ *good* if (1) the dyadic interval $I_{m,i}$ is β-Blue-flag-balanced in w, and (2) for both intervals $I \in \{I_{m-1,2i-1}, I_{m-1,2i}\}$, there are at most $\beta|I|$ indices $i \in I$ with $b_w(i) \geq 2^{n-200\beta^{-3}\log n}$. By the choice of m, there are less than $\beta \cdot 2^{n-m}$ choices of $i \in [2^{n-m}]$ that violate requirement (1). Next, we use the assumption that in total at most $\beta^2 L$ indices $i \in [L]$ satisfy $b_w(i) \geq 2^{n-200\beta^{-3}\log n}$. Thus, at most $\beta \cdot 2^{n-m+1}$ of the dyadic intervals I of size 2^{m-1} contain at least $\beta|I|$ indices $i \in I$ satisfying $b_w(i) \geq 2^{n-200\beta^{-3}\log n}$. Hence, at most $2\beta \cdot 2^{n-m}$ choices of $i \in [2^{n-m}]$ violate requirement (2). We see that all but less than $3\beta \cdot 2^{n-m}$ choices of $i \in [2^{n-m}]$ are good.

We now show that for each good index $i \in [2^{n-m}]$, the substring $w_{m,i}$ is 6β-Blue-flag-balanced. By Lemma 3.18, for each good i and either interval $I \in \{I_{m-1,2i-1}, I_{m-1,2i}\}$, we have

$$\|p_{w,I} - p_{w_I}\|_1 \leq 2\left(\beta + \frac{2^{n-200\beta^{-3}\log n}}{2^m}\right) \leq 2\left(\beta + 2^{-50\beta^{-3}\log n}\right).$$

Since interval $I_{m,i}$ is β-Blue-flag-balanced with respect to w, we have by the triangle inequality

$$\begin{aligned}\|p_{w_{m-1,2i-1}} - p_{w_{m-1,2i}}\|_1 &\leq \|p_{w_{m-1,2i-1}} - p_{w,I_{m-1,2i-1}}\|_1 + \|p_{w_{m-1,2i}} - p_{w,I_{m-1,2i}}\|_1 + \|p_{w,I_{m-1,2i-1}} - p_{w,I_{m-1,2i}}\|_1 \\ &\leq 4(\beta + 2^{-50\beta^{-3}\log n}) + \beta < 6\beta,\end{aligned}$$

assuming n is sufficiently large. Hence, for each good i, the substring $w_{m,i}$ is 6β-Blue-flag-balanced, as desired. □

3.6 Green case

Call a pair of strings (s,t) a *Green pair* if

1. s and t have the same number L of ones, and
2. there exists an $1 \leq \ell \leq \varepsilon^5 L$ such that s and t are both type ℓ-Green.

In this section, we implement the Green strategy in the Overview (Section 3.2). Specifically, we show (Lemma 3.21) that when we have strings s and t and some scale m^* with many Green pairs $(s_{m^*,i}, t_{m^*,i})$, then we can find a common subsequence of s and t of length $(0.5+\delta)|s|$. At the highest level, we do this by finding common subsequences within the Green pairs, and matching ones elsewhere, using the Prefix/Suffix LCS Lemma (Lemma 3.9).

Within the Green pairs, we match ones everywhere, except that we switch to matching zeros at Green flags in s and t.

Lemma 3.20 (Green matching lemma)**.** *Let* L *be a power of two. Let* (s,t) *be a Green pair where strings* s

and t have L ones each. Then for Δ uniformly random in $[\![-L, L]\!]$, we have

$$\mathop{\mathbf{E}}_{\Delta}[\mathrm{LCS}(\mathrm{Trim}_{\Delta}(s), \mathrm{Trim}_{-\Delta}(t)) + |\Delta|] \geq L + \frac{\varepsilon^5}{8}L.$$

Proof. For a fixed Δ in $[\![-L, L]\!]$, let G'_Δ be the set of indices $i \in [L]$ such that i is Green ℓ-flag of s and $i + \Delta \in [L]$ and $i + \Delta$ is a Green ℓ-flag of t. Let $G_\Delta \subseteq G'_\Delta$ be a subset of size at least $|G'_\Delta|/\ell - 1$ such that any two distinct $i, i' \in G_\Delta$ satisfy $|i - i'| \geq \ell$, and furthermore every $i \in G_\Delta$ satisfies $i + \Delta + \ell \leq L$. Such a G_Δ can be greedily selected from G'_Δ: we can get $|G_\Delta|/\ell$ indices such that any two differ by at least ℓ, and, as any $i \in G'_\Delta$ satisfies $i + \Delta \leq L$, we can guarantee $i + \Delta + \ell \leq L$ for all $i \in G_\Delta$ by removing the largest index.

Since s and t are type ℓ-Green, both s and t have at least $\varepsilon^2 L$ Green ℓ-flags. Thus, for each Green ℓ-flag $i \in [L]$ of s, there are at least $\varepsilon^2 L$ values of Δ such that $i \in G'_\Delta$. Hence, $\mathbf{Pr}_{\Delta \in [\![-L,L]\!]}[i \in G'_\Delta] \geq \frac{\varepsilon^2 L}{2L+1} > \frac{\varepsilon^2}{3}$. By linearity of expectation, we have $\mathbf{E}_{\Delta \sim [\![-L,L]\!]}[|G'_\Delta|] \geq \frac{\varepsilon^2}{3} \cdot \varepsilon^2 L = \frac{\varepsilon^4}{3}L$. Hence, as $|G_\Delta| \geq |G'_\Delta|/\ell - 1$, we have

$$\mathop{\mathbf{E}}_{\Delta \sim [\![-L,L]\!]}[|G_\Delta|] \geq \frac{\varepsilon^4 L}{3\ell} - 1 > \frac{\varepsilon^4 L}{4\ell}. \tag{3.7}$$

Here we used that $\ell \leq \varepsilon^5 L$.

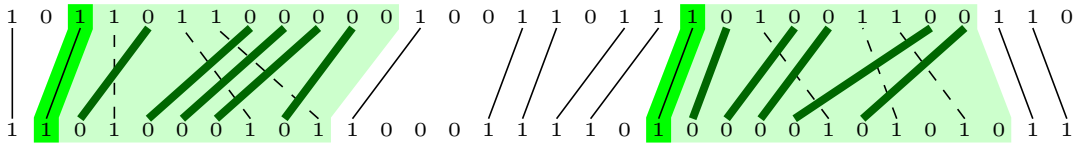


Figure 3.5: The matching strategy for Lemma 3.20. In this example $\ell = 4$ and $\Delta = 0$. The darker Green one-bits are Green ℓ-flags, and the light Green substrings following them are relatively zero-rich. The solid and dashed black lines indicate the initial matching of ones, and the thick dark-green lines indicate matchings of zeros that replace the matchings of ones. The matchings of ones that are replaced are dashed.

It suffices to construct a large matching between the bits of s and the bits of t such that only equal bits are matched and such that the matching is "non-crossing", meaning that earlier bits of s are matched with earlier bits of t. Indeed, the number of pairs in a non-crossing matching corresponds to the length of a common subsequence of s and t.

Consider the matching where we match the i-th one in s with the $(i + \Delta)$-th one in t. This matches $L - |\Delta|$ ones. In particular, for each $i \in G_\Delta$, the ones in substring $s_{[\![i+1, i+\ell-1]\!]}$ are exactly matched with the ones in substring $t_{[\![i+\Delta+1, i+\Delta+\ell-1]\!]}$. In this matching, for each $i \in G_\Delta$, replace the matching between the ones of $s_{[\![i+1, i+\ell-1]\!]}$ and the ones of $t_{[\![i+\Delta+1, i+\Delta+\ell-1]\!]}$ with a matching between the zeros of $s_{[\![i, i+\ell-1]\!]}$ and the zeros of $t_{[\![i+\Delta, i+\Delta+\ell-1]\!]}$. All of the zeros of $s_{[\![i, i+\ell-1]\!]}$ come after the i-th one of s, and all the zeros of $t_{[\![i+\Delta, i+\Delta+\ell-1]\!]}$ come after the $(i + \Delta)$-th one of t, which stays matched to the i-th one of s, so the matching stays non-crossing after this replacement. Furthermore, since any two $i \in G_\Delta$ differ by at least ℓ, the substrings $s_{[\![i, i+\ell-1]\!]}$ and $t_{[\![i+\Delta, i+\Delta+\ell-1]\!]}$ across $i \in G_\Delta$ are pairwise disjoint, so these $|G_\Delta|$ replacements can be done simultaneously while keeping the matching non-crossing and thus valid.

Substrings $s_{[\![i, i+\ell-1]\!]}$ and $t_{[\![i+\Delta, i+\Delta+\ell-1]\!]}$ each have at least $\max(1, (1 + \varepsilon)(\ell - 1))$ zeros because i is a Green ℓ-flag of s and $i + \Delta$ is a Green ℓ-flag of t. Thus, each replacement deletes $(\ell - 1)$ pairs of ones and

adds at least $\max(1, (1+\varepsilon)(\ell-1))$ pairs of zeros. If $\ell \geq 2$, each replacement increases the number of pairs matched by $\varepsilon(\ell-1) > \frac{\varepsilon}{2}\ell$, so the total number of bits in the new matching is at least $L - |\Delta| + \frac{\varepsilon}{2}\ell|G_\Delta|$. If $\ell = 1$, each replacement increases the number of pairs matched by at least 1, so the total number of pairs in the new matching is $L - |\Delta| + |G_\Delta|$ which is also at least $L - |\Delta| + \frac{\varepsilon}{2}\ell|G_\Delta|$. Thus, for all Δ, we have shown

$$\mathrm{LCS}(\mathrm{Trim}_\Delta(s), \mathrm{Trim}_{-\Delta}(t)) \geq L - |\Delta| + \frac{\varepsilon}{2}\ell|G_\Delta|.$$

Hence, we have by linearity of expectation and (3.7),

$$\mathop{\mathbf{E}}_{\Delta\sim[\![-L,L]\!]}[\mathrm{LCS}(\mathrm{Trim}_\Delta(s), \mathrm{Trim}_{-\Delta}(t)) + |\Delta|] \geq L + \frac{\varepsilon}{2}\ell \cdot \mathop{\mathbf{E}}_{\Delta\sim[\![-L,L]\!]}[|G_\Delta|] \geq L + \frac{\varepsilon^5}{8}L,$$

as desired. □

To find strings s and t that have LCS beating the 1/2 barrier, it is not enough to assume that s and t form a Green pair and use Lemma 3.20, because we lose by the size of the random shift $|\Delta|$. To remedy this, we break s and t into shorter substrings so that the loss from the random shift is at most the length of a single substring. If a substantial fraction of these substrings form Green pairs, then we can combine the gains using the Shifted LCS Lemma (Lemma 3.9) to get a large overall LCS, and the loss from the random shift is negligible. The following lemma implements this idea, showing that many Green pairs gives large overall LCS.

Lemma 3.21 (Many Green pairs implies large LCS)**.** *Let s and t be strings with the same length and the same number $L = 2^n$ of ones, and let $m^* \leq n - 10 - \log\varepsilon^{-5}$. Suppose there exists a set $Z \subset [2^{n-m^*}]$ such that $|Z| > 2^{n-m^*}/10$ and for all $i \in Z$, the pair of substrings $(s_{m^*,i}, t_{m^*,i})$ is a Green pair. Then*

$$\mathrm{LCS}(s,t) \geq \left(0.5 + \frac{\varepsilon^5}{5000}\right)|s|.$$

Proof. Let $L^* = 2^{m^*}$. We may assume $L \geq (0.5 - \frac{\varepsilon^5}{1000})|s|$ or else we are done by having an LCS of $(0.5 + \frac{\varepsilon^5}{1000})|s|$ zeros. By Lemma 3.20, we have, for all $i \in Z$,

$$\mathop{\mathbf{E}}_{\Delta\sim[\![-L^*,L^*]\!]}[\mathrm{LCS}\left(\mathrm{Trim}_\Delta(s_{m^*,i}), \mathrm{Trim}_{-\Delta}(t_{m^*,i})\right) + |\Delta|] \geq L^* + \frac{\varepsilon^5}{8}L^*.$$

We have, by linearity of expectation,

$$\begin{aligned}
&\mathop{\mathbf{E}}_{\Delta\sim[\![-L^*,L^*]\!]}\left[\sum_{i\in Z}\left(\mathrm{LCS}\left(\mathrm{Trim}_\Delta(s_{m^*,i}), \mathrm{Trim}_{-\Delta}(t_{m^*,i})\right) + |\Delta|\right)\right] \\
&\qquad = \sum_{i\in Z}\mathop{\mathbf{E}}_{\Delta\sim[\![-L^*,L^*]\!]}[\mathrm{LCS}\left(\mathrm{Trim}_\Delta(s_{m^*,i}), \mathrm{Trim}_{-\Delta}(t_{m^*,i})\right) + |\Delta|] \\
&\qquad \geq |Z| \cdot \left(L^* + \frac{\varepsilon^5}{8}L^*\right) .
\end{aligned}$$

Hence, we may fix a single Δ for which $|\Delta| \le L^*$ and

$$\sum_{i \in Z} \mathrm{LCS}\left(\mathrm{Trim}_{\Delta}(s_{m^*,i}), \mathrm{Trim}_{-\Delta}(t_{m^*,i})\right) \ge |Z| \cdot \left(L^* - |\Delta| + \frac{\varepsilon^5}{8} L^*\right).$$

Thus, the set Z satisfies the setup of Lemma 3.9 with $n' = n$ and $m' = m^*$ and $L' = L$ and $\delta' = \frac{\varepsilon^5}{8}$. Therefore we get

$$\mathrm{LCS}(s,t) \ge \left(1 + \frac{\varepsilon^5}{160}\right) L > \left(1 + \frac{\varepsilon^5}{160}\right)\left(0.5 - \frac{\varepsilon^5}{1000}\right)|s| > \left(0.5 + \frac{\varepsilon^5}{5000}\right)|s|. \qquad \square$$

3.7 Blue-Yellow case

In this section, we implement the Blue-Yellow strategy described in the Overview (Section 3.2), which is the most intricate part of our argument. Call a pair of strings (s,t) a *Blue-Yellow pair* if:

1. The strings s and t have the same length and the same number of ones.
2. There exists an m such that s and t are both of type m-Blue-Yellow.
3. Both s and t are 6γ-Blue-flag-balanced.

Note that (s,t) is a Blue-Yellow pair if and only if (t,s) is a Blue-Yellow pair.

We show that when we have strings s and t and some scale m^* with many Blue-Yellow pairs $(s_{m^*,i}, t_{m^*,i})$ among their substrings at scale m^*, then we can find a common subsequence of s and t of length $(1/2+\delta)|s|$. At the highest level (Lemma 3.24), we do this by finding common subsequences within the Blue-Yellow pairs, and matching ones elsewhere. To find large common subsequences within Blue-Yellow pairs, we use Lemma 3.22 and Lemma 3.23.

Lemma 3.22 shows that, for a Blue-Yellow pair (s_i, t_i), we can find substrings s_i' and t_i' of s_i and t_i which "gain in span," meaning that $\mathrm{LCS}(s_i', t_i') > (1/4+\delta)(|s_i'| + |t_i'|)$. To do this, we match ones of s_i with ones of t_i, and switch to matching zeros when we simultaneously encounter a Blue flag in s_i and Yellow flag in t_i of the appropriate lengths. However, despite gaining in span, the lengths of s_i and t_i may be quite different (this offset comes from the zeros of the Blue flags in s_i spanning fewer ones than the zeros of the Yellow flags in t_i), so repeatedly applying Lemma 3.22 is insufficient. Indeed, this would cause us to systematically match shorter substrings in s with longer substrings in t, leading the common subsequence obtained to reach the end of t well before reaching the end of s.

The subsequent lemma, Lemma 3.23, shows that if *two* Blue-Yellow pairs (s_i, t_i) and (s_{i+1}, t_{i+1}) occur consecutively in a string, then we can find a common subsequence that gains in span, like Lemma 3.22, but also uses the same number of ones in both strings, creating a balanced matching. Roughly, the proof follows by applying Lemma 3.22 twice, once on the pair (s_i, t_i), matching more bits in t_i, and once on the pair (t_{i+1}, s_{i+1}), matching more bits in s_{i+1}, so that together the number of bits used from s and t is equal. As a result, in total our common subsequence spans the same number of bits in s and t but still gains in span. We refer the reader to Figure 3.6 below for a visual illustration of this argument.

Thanks to Lemma 3.23, it follows that if we have many Blue-Yellow pairs, then we have many pairs of subsequences $(s_i s_{i+1}, t_i t_{i+1})$ of s and t where there is a common subsequence that gains in span. Thus, using

Lemma 3.9, we can piece these subsequences together, matching ones in between, giving a large LCS overall; this is the content of Lemma 3.24.

Lemma 3.22 (Blue-Yellow matching lemma)**.** *Let (s,t) be a Blue-Yellow pair where each string has $L = 2^n$ ones. Then, for Δ uniformly random in $[L/4]$,*

$$\Pr_{\Delta}\left[\mathrm{LCS}(s_{[\![1,(0.5+0.01\varepsilon)L]\!]}, t_{[\![1+\Delta,(0.5+0.3\varepsilon)L+\Delta]\!]}) \geq (0.5+0.24\varepsilon)L\right] \geq 0.9.$$

Proof. This proof is long, so we organize it into four parts. First, we use the assumptions to find many disjoint Blue flags in s. Next, we match the Blue flags in s with candidate Yellow flags in t. We then show that, on average, most of these candidate flags are in fact Yellow flags in t. Finally, with these ingredients in place, we show the desired lower bound on the expected LCS over the random offset Δ.

Finding many disjoint Blue flags in s. Since (s,t) is a Blue-Yellow pair, the strings s and t are both type m-Blue-Yellow for some integer m. Because s is type m-Blue-Yellow, the fraction of indices $i \in [L]$ with $b_s(i) \in [\![2^m, \gamma L]\!]$ is at least $\varepsilon^2 - \gamma$. As s is 6γ-Blue-flag-balanced, by Lemma 3.15, the fraction of indices $i \in [L/2]$ with $b_s(i) \in [\![2^m, \gamma L]\!]$ is at least $\varepsilon^2 - 4\gamma$. Let $1 \leq i_1 < i_2 < \cdots$ be such that $i_k \in [L/2]$ is the smallest index satisfying $b_s(i_k) \in [\![2^m, \gamma L]\!]$ and, if $k > 1$, $i_k \geq i_{k-1} + b_s(i_{k-1})$. Intuitively, $(i_k)_{k\geq 1}$ is a maximal sequence of non-overlapping Blue flags in $[L/2]$.

We claim that there is a smallest index d such that $\sum_{k=1}^{d} b_s(i_k) \geq 0.5(\varepsilon^2 - 4\gamma)L$. By the minimality of each i_k, the intervals $[\![i_k, i_k + b_s(i_k) - 1]\!]$ for $k = 1, \ldots, k'$ together contain all indices $i < i_{k'+1}$ with $b_s(i) \in [\![2^m, \gamma L]\!]$. Since there are at least $0.5(\varepsilon^2 - 4\gamma)L$ such indices $i \in [L/2]$, as long as $\sum_{k=1}^{k'} b_s(i_k) < 0.5(\varepsilon^2 - 4\gamma)L$, the index $i_{k'+1}$ is well defined, so in particular d is well defined. Furthermore, by minimality of d, we have $\sum_{k=1}^{d} b_s(i_k) \leq 0.5(\varepsilon^2 - 4\gamma)L + \gamma L < 0.5\varepsilon^2 L$. We thus have found indices $i_1, \ldots, i_d$ where

$$i_d \leq L/2,\ i_{k+1} \geq i_k + b_s(i_k) \text{ for each } k < d, \text{ and } \sum b_s(i_k) \in [0.5(\varepsilon^2 - 4\gamma)L, 0.5\varepsilon^2 L].$$

Matching Blue flags in s with candidate Yellow flags in t. Define $i_0 = 1$, $\ell_{s,0} = 0$, and $i_{d+1} = \lfloor (0.5 + 0.01\varepsilon)L \rfloor$. For $k \in [d]$, let $\ell_{s,k} \stackrel{\text{def}}{=} b_s(i_k) > 1$ for the lengths of these Blue flags in s, and let $\ell_{t,k} \stackrel{\text{def}}{=} \lceil 0.56\varepsilon^{-1}\ell_{s,k} \rceil$ for $k \in [0,d]$, corresponding to Yellow flag length we want to match in t. Here the constant $0.56\varepsilon^{-1}$ above is chosen so that $0.9(\ell_{t,k} - 1) \geq 0.5\varepsilon^{-1}\ell_{s,k}$, so that a Yellow $\ell_{t,k}$-flag guarantees roughly the same number of zeros in t as a Blue $\ell_{s,k}$-flag does in s.

For a subset $K \subset [0,d]$, define $\ell_{s,K} \stackrel{\text{def}}{=} \sum_{k\in K} \ell_{s,k}$, and define $\ell_{t,K}$ similarly. Recall the shift Δ is to be chosen uniformly from $[L/4]$. For $k \in [0, d+1]$, let $j_k \stackrel{\text{def}}{=} i_k + \Delta + \ell_{t,[k-1]} - \ell_{s,[k-1]}$. This j_k is the index in t that we would like to be a Yellow $\ell_{t,k}$-flag to be matched with the Blue $\ell_{s,k}$-flag at i_k. The indices i_k and j_k partition s and t into substrings as follows. Let

$$\begin{aligned}
s'_k &\stackrel{\text{def}}{=} s_{[\![i_k, i_k+\ell_{s,k}-1]\!]} && \text{for } k \in [\![1,d]\!] \\
s''_k &\stackrel{\text{def}}{=} s_{[\![i_k+\ell_{s,k}, i_{k+1}-1]\!]} && \text{for } k \in [\![0,d]\!] \\
t'_k &\stackrel{\text{def}}{=} t_{[\![j_k, j_k+\ell_{t,k}-1]\!]} && \text{for } k \in [\![1,d]\!] \\
t''_k &\stackrel{\text{def}}{=} t_{[\![j_k+\ell_{t,k}, j_{k+1}-1]\!]} && \text{for } k \in [\![0,d]\!].
\end{aligned}$$

By construction of $i_0, \ldots, i_d$, we have $i_k + \ell_{s,k} \leq i_{k+1}$ for $k \in [\![0, d-1]\!]$, and also $i_d + \ell_{s,d} \leq L/2 + \gamma L <$

i_{d+1}. Thus, the substrings defined above give a partition $s_{[\![1,(0.5+0.01\varepsilon)L]\!]} = s''_0 s'_1 s''_1 s'_2 s''_2 \dots s'_d s''_d$, where the substrings alternate between the regions s'_k corresponding to Blue flags and the regions s''_k in between. Similarly, for $k \in [\![0, d-1]\!]$ we have

$$j_k + \ell_{t,k} = i_k + \Delta + \ell_{t,k} - \ell_{s,[k-1]} = j_{k+1} - (i_{k+1} - i_k - \ell_{s,k}) \le j_{k+1},$$

and

$$\begin{aligned} j_{d+1} &= i_{d+1} + \ell_{t,[d]} - \ell_{s,[d]} + \Delta \\ &\le 1 + L/2 + \gamma L + 0.57\varepsilon^{-1} \cdot \ell_{s,[d]} + \Delta \\ &\le 1 + L/2 + 0.01\varepsilon L + 0.57\varepsilon^{-1} \cdot 0.5\varepsilon^2 L + \Delta \\ &\le 1 + (0.5 + 0.3\varepsilon)L + \Delta. \end{aligned}$$

Thus, we have that $t''_0 t'_1 t''_1 t'_2 t''_2 \dots t'_d t''_d$ is a prefix of $t_{[\![1+\Delta,(0.5+0.3\varepsilon)L+\Delta]\!]}$, alternating between the regions t'_k that we would like to be Yellow flags and the regions t''_k in between.

Showing many j_k's are Yellow flags of t. We next show that typically, most of the j_k's as defined above are Yellow flags. Let K be the set of $k \in [d]$ for which the j_k is a Yellow $\ell_{t,k}$-flag of string t. Call a shift $\Delta \in [L/4]$ *good* if $\ell_{t,K} \ge (1 - 24000\varepsilon)\ell_{t,[d]}$, i.e. that when weighted by flag length, the set K comprises most of $[d]$. Since string t has type m-Blue-Yellow and $\ell_{t,k} > \ell_{s,k} \ge 2^m$ for every $k \in [d]$, we know that for every $k \in [d]$, at most $600\varepsilon L$ of the indices in $[L]$ are not Yellow $\ell_{t,k}$-flags in string t. Hence, if Δ is chosen uniformly at random in $[L/4]$, for each $k \in [d]$, the probability that $k \notin K$ is at most $600\varepsilon L/(L/4) = 2400\varepsilon$. By linearity of expectation,

$$\mathop{\mathbf{E}}_{\Delta \sim [L/4]}[\ell_{t,[d]} - \ell_{t,K}] \le 2400\varepsilon \ell_{t,[d]},$$

As $\ell_{t,K} \le \ell_{t,[d]}$ always, we have by Markov's inequality on $\ell_{t,[d]} - \ell_{t,K}$ that

$$\mathbf{Pr}[\Delta \text{ is good}] = 1 - \mathbf{Pr}\left[\ell_{t,[d]} - \ell_{t,K} \ge (24000\varepsilon)\ell_{t,[d]}\right] \ge 1 - \frac{2400\varepsilon\ell_{t,[d]}}{24000\varepsilon\ell_{t,[d]}} = 0.9.$$

Lower bounding expected LCS. Since $s''_0 s'_1 s''_1 s'_2 s''_2 \dots s'_d s''_d = s_{[\![1,(0.5+0.01\varepsilon)L]\!]}$ and $t''_0 t'_1 t''_1 t'_2 t''_2 \dots t'_d t''_d$ is a prefix of $t_{[\![1+\Delta,(0.5+0.3\varepsilon)L+\Delta+1]\!]}$, it suffices to show that, when Δ is good,

$$\mathrm{LCS}(s''_0 s'_1 s''_1 s'_2 s''_2 \dots s'_d s''_d, t''_0 t'_1 t''_1 t'_2 t''_2 \dots t'_d t''_d) \ge (0.5 + 0.24\varepsilon)L.$$

As the index i_k is a Blue $\ell_{s,k}$-flag in s, substring s'_k has at least $\varepsilon^{-1}(\ell_{s,k} - 1) \ge 0.5\varepsilon^{-1}\ell_{s,k}$ zeros. If $k \in K$, then j_k is a Yellow $\ell_{t,k}$-flag in t, so substring t'_k has at least $0.9(\ell_{t,k} - 1) > 0.5\varepsilon^{-1}\ell_{s,k}$ zeros as well. Hence, for $k \in K$, we have

$$\mathrm{LCS}(s'_k, t'_k) \ge 0.5\varepsilon^{-1}\ell_{s,k}.$$

Furthermore, for $k \in [\![0, d]\!]$, we have that s''_k has $i_{k+1} - i_k - \ell_{s,k}$ ones, and the number of ones in t''_k is also

$$j_{k+1} - j_k - \ell_{t,k} = (i_{k+1} + \Delta + \ell_{t,k} - \ell_{s,[k]}) - (i_k + \Delta + \ell_{t,[k-1]} - \ell_{s,[k-1]}) - \ell_{t,k} = i_{k+1} - i_k - \ell_{s,k},$$

so for all $k \in [\![0, d]\!]$,

$$\mathrm{LCS}(s''_k, t''_k) \geq i_{k+1} - i_k - \ell_{s,k}.$$

Indeed, we now have that for any good Δ,

$$\begin{aligned}
\mathrm{LCS}(s_{[\![1,(0.5+0.01\varepsilon)L]\!]}, t_{[\![1+\Delta,(0.5+0.3\varepsilon)L+\Delta]\!]}) &\geq \sum_{k=1}^{d} \mathrm{LCS}(s'_k, t'_k) + \sum_{k=0}^{d} \mathrm{LCS}(s''_k, t''_k) \\
&\geq \sum_{k\in K} \mathrm{LCS}(s'_k, t'_k) + \sum_{k=0}^{d} (i_{k+1} - i_k - \ell_{s,k}) \\
&\geq 0.5\varepsilon^{-1}\ell_{s,K} + (i_{d+1} - i_0 - \ell_{s,[d]}) && (3.8) \\
&\geq 0.5\varepsilon^{-1}(1 - 24000\varepsilon)\ell_{s,[d]} + (0.5 + 0.01\varepsilon)L - 2 - \ell_{s,[d]} && (3.9) \\
&= 0.5\varepsilon^{-1}(1 - 24002\varepsilon)\ell_{s,[d]} + (0.5 + 0.01\varepsilon)L - 2 \\
&\geq 0.5\varepsilon^{-1}(1 - 24002\varepsilon)(0.5(\varepsilon^2 - 4\gamma)L) + (0.5 + 0.01\varepsilon)L && (3.10) \\
&\geq (0.5 + 0.24\varepsilon)L. && (3.11)
\end{aligned}$$

In (3.8), we used that $\ell_{s,0} = 0$. In (3.9), we used that $i_{d+1} - i_0 = \lfloor (0.5 + 0.01\varepsilon)L \rfloor - 1 > (0.05 + 0.01\varepsilon)L - 2$. In (3.10), we used that $\ell_{s,[d]} = \sum_{k=1}^{d} b_s(i_k) \geq 0.5(\varepsilon^2 - 4\gamma)L$ by construction of $i_1, \ldots, i_d$. In (3.11), we used that $\varepsilon \leq 10^{-6}$ and $\gamma \leq 0.001\varepsilon^2$. This completes the proof as the probability that $\Delta \in [L/4]$ is good is at least 0.9. □

The next lemma shows that when two Blue-Yellow pairs occur consecutively (for technical reasons, we require that the second Blue-Yellow pair occurs on the reversed strings), we get a large common subsequence that also spans the same number of ones in both s and t. This allows us to piece together many such matchings and apply the Prefix/Suffix LCS Lemma (Lemma 3.9), giving an overall gain in LCS over the 1/2 barrier.

Lemma 3.23 (Blue-Yellow balanced matching lemma)**.** *Let $s = s_1 s_2$ and $t = t_1 t_2$ such that substrings s_1, s_2, t_1, t_2 each have L ones and start with a one, and the pairs (s_1, t_1) and $(\mathrm{rev}(t_2), \mathrm{rev}(s_2))$ are Blue-Yellow pairs. Then for Δ uniformly random in $[L/4]$, we have*

$$\mathop{\mathbf{E}}_{\Delta\sim[L/4]} [\mathrm{LCS}(\mathrm{Trim}_\Delta(s), \mathrm{Trim}_{-\Delta}(t)) + \Delta] \geq 2L + 0.12\varepsilon L.$$

Proof. We have $\mathrm{Trim}_\Delta(s)$ and $\mathrm{Trim}_{-\Delta}(t)$ each have $2L - \Delta$ ones, so we always have

$$\mathrm{LCS}(\mathrm{Trim}_\Delta(s), \mathrm{Trim}_{-\Delta}(t)) + \Delta \geq 2L.$$

Thus, because $0.8 \cdot 0.16\varepsilon L > 0.12\varepsilon L$, it suffices to show that

$$\mathop{\mathbf{Pr}}_{\Delta\sim[L/4]} [\mathrm{LCS}(\mathrm{Trim}_\Delta(s), \mathrm{Trim}_{-\Delta}(t)) + \Delta \geq 2L + 0.16\varepsilon L] \geq 0.8.$$

Call Δ *good* if both of the following inequalities are true:

$$\mathrm{LCS}\left((s_1)_{[\![1,(0.5+0.01\varepsilon)L]\!]}, (t_1)_{[\![1+\Delta,(0.5+0.3\varepsilon)L+\Delta]\!]}\right) \geq (0.5+0.24\varepsilon)L \tag{3.12}$$

$$\mathrm{LCS}\left(\mathrm{rev}(t_2)_{[\![1,(0.5+0.01\varepsilon)L]\!]}, \mathrm{rev}(s_2)_{[\![1+\Delta,(0.5+0.3\varepsilon)L+\Delta]\!]}\right) \geq (0.5+0.24\varepsilon)L. \tag{3.13}$$

By Lemma 3.22, first applied to the Blue-Yellow pair (s_1, t_1), then applied to the Blue-Yellow pair $(\mathrm{rev}(t_2), \mathrm{rev}(s_2))$, each of (3.12) and (3.13) holds with probability at least 0.9 for Δ sampled uniformly from $[L/4]$, so a random Δ in $[L/4]$ is good with probability at least 0.8.

Fix a good Δ. We show that

$$\mathrm{LCS}(\mathrm{Trim}_\Delta(s), \mathrm{Trim}_{-\Delta}(t)) \geq 2L - \Delta + 0.16\varepsilon L. \tag{3.14}$$

By the second part of Lemma 3.3,

$$\begin{aligned}\mathrm{rev}(t_2)_{[\![1,(0.5+0.01\varepsilon)L]\!]} &= \mathrm{rev}\left((t_2)_{[\![1+(0.5-0.01\varepsilon)L,L]\!]}\right)\\ \mathrm{rev}(s_2)_{[\![1+\Delta,(0.5+0.3\varepsilon)L+\Delta]\!]} &= \mathrm{rev}\left((s_2)_{[\![1+(0.5-0.3\varepsilon)L-\Delta,L-\Delta]\!]}\right).\end{aligned}$$

Hence since $\mathrm{rev}(\cdot)$ preserves LCS (Lemma 3.5), we have

$$\begin{aligned}&\mathrm{LCS}\left((t_2)_{[\![1+(0.5-0.01\varepsilon)L,L]\!]}, (s_2)_{[\![1+(0.5-0.3\varepsilon)L-\Delta,L-\Delta]\!]}\right)\\ &= \mathrm{LCS}\left(\mathrm{rev}(t_2)_{[\![1,(0.5+0.01\varepsilon)L]\!]}, \mathrm{rev}(s_2)_{[\![1+\Delta,(0.5+0.3\varepsilon)L+\Delta]\!]}\right)\\ &\geq (0.5+0.24\varepsilon)L\ .\end{aligned}$$

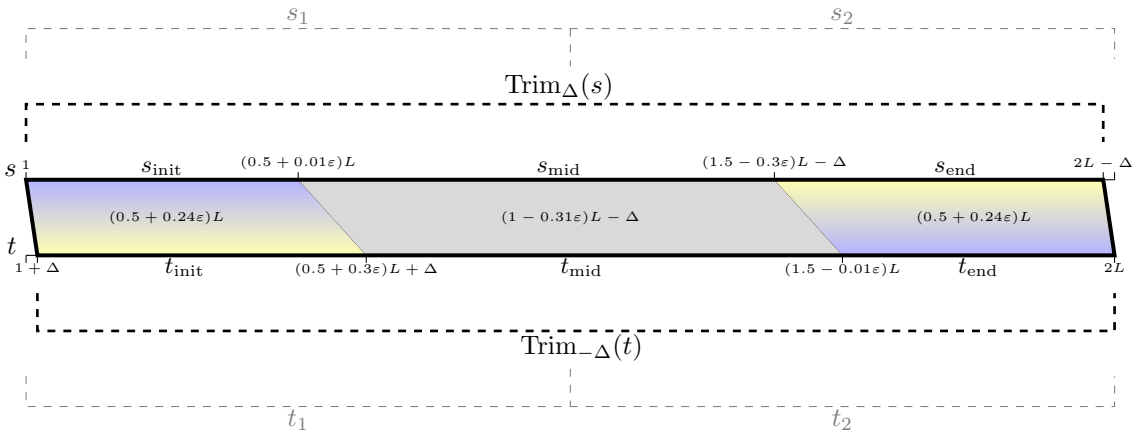


Figure 3.6: Lemma 3.23. We obtain two "trapezoids" with good LCS obtained from applying Lemma 3.22 to two Blue-Yellow pairs, (s_1, t_1) and $(\mathrm{rev}(s_2), \mathrm{rev}(t_2))$, and match ones in between (in the grey region), giving an improved LCS for $\mathrm{Trim}_\Delta(s)$ and $\mathrm{Trim}_{-\Delta}(t)$.

Writing $\mathrm{Trim}_{\Delta}(s) = s_{\mathrm{init}} s_{\mathrm{mid}} s_{\mathrm{end}}$ and $\mathrm{Trim}_{-\Delta}(t) = t_{\mathrm{init}} t_{\mathrm{mid}} t_{\mathrm{end}}$ where (see Figure 3.6)[5]

$$\begin{aligned}
s_{\mathrm{init}} &\stackrel{\mathrm{def}}{=} s_{[\![1,(0.5+0.01\varepsilon)L]\!]} && = (s_1)_{[\![1,(0.5+0.01\varepsilon)L]\!]} \\
s_{\mathrm{mid}} &\stackrel{\mathrm{def}}{=} s_{[\![(0.5+0.01\varepsilon)L,(1.5-0.3\varepsilon)L-\Delta+1]\!]} && \\
s_{\mathrm{end}} &\stackrel{\mathrm{def}}{=} s_{[\![(1.5-0.3\varepsilon)L-\Delta+1,2L-\Delta]\!]} && = (s_2)_{[\![(0.5-0.3\varepsilon)L-\Delta+1,L-\Delta]\!]} \\
t_{\mathrm{init}} &\stackrel{\mathrm{def}}{=} t_{[\![1+\Delta,(0.5+0.3\varepsilon)L+\Delta]\!]} && = (t_1)_{[\![1+\Delta,(0.5+0.3\varepsilon)L+\Delta]\!]} \\
t_{\mathrm{mid}} &\stackrel{\mathrm{def}}{=} t_{[\![(0.5+0.3\varepsilon)L+\Delta,(1.5-0.01\varepsilon)L+1]\!]} && \\
t_{\mathrm{end}} &\stackrel{\mathrm{def}}{=} t_{[\![(1.5-0.01\varepsilon)L+1,L]\!]} && = (t_2)_{[\![(0.5-0.01\varepsilon)L+1,L]\!]} \ .
\end{aligned}$$

We have s_{mid} and t_{mid} both have at least $(1-0.31\varepsilon)L - \Delta$ ones. Hence, we have

$$\begin{aligned}
\mathrm{LCS}(\mathrm{Trim}_{\Delta}(s), \mathrm{Trim}_{-\Delta}(t)) &\geq \mathrm{LCS}(s_{\mathrm{init}}, t_{\mathrm{init}}) + \mathrm{LCS}(s_{\mathrm{mid}}, t_{\mathrm{mid}}) + \mathrm{LCS}(s_{\mathrm{end}}, t_{\mathrm{end}}) \\
&\geq (0.5+0.24\varepsilon)L + (1-0.31\varepsilon)L - \Delta + (0.5+0.24\varepsilon)L \\
&> 2L - \Delta + 0.16\varepsilon L,
\end{aligned}$$

establishing (3.14) as desired. □

Combining Lemma 3.23 with the Prefix/Suffix LCS Lemma (Lemma 3.9) shows that when two strings have many Blue-Yellow pairs among their substrings at some scale, we get a large LCS.

Lemma 3.24 (Many Blue-Yellow pairs implies good LCS)**.** *Let s and t be strings with the same length and the same number $L = 2^n$ of ones, and let $m^* \leq n - 10 - \log \varepsilon^{-1}$. Suppose there exists a set $Z \subset [2^{n-m^*-1}]$ such that $|Z| > 2^{n-m^*-1}/10$ and for all $i \in Z$, the substring pairs $(s_{m^*,2i-1}, t_{m^*,2i-1})$ and $(\mathrm{rev}(s_{m^*,2i}), \mathrm{rev}(t_{m^*,2i}))$ are Blue-Yellow pairs. Then,*

$$\mathrm{LCS}(s,t) > (1 + 0.0001\varepsilon)\,|s|.$$

Proof. Let $L^* = 2^{m^*}$. We may assume $L \geq (0.5 - 0.001\varepsilon)|s|$ or else we are done by having an LCS of $(0.5+0.001\varepsilon)|s|$ zeros. For all $i \in Z$, substrings $s_{m^*+1,i} = s_{m^*,2i-1} s_{m^*,2i}$ and $t_{m^*+1,i} = t_{m^*,2i-1} t_{m^*,2i}$ satisfy the setup of Lemma 3.23 with $L = L^*$, so we have, for all $i \in Z$,

$$\mathop{\mathbf{E}}_{\Delta \sim [L^*/4]} \left[\mathrm{LCS}\left(\mathrm{Trim}_{\Delta}(s_{m^*+1,i}), \mathrm{Trim}_{-\Delta}(t_{m^*+1,i})\right) + \Delta \right] \geq 2L^* + 0.12\varepsilon L^* \ .$$

By linearity of expectation, we have

$$\mathop{\mathbf{E}}_{\Delta \sim [L^*/4]} \left[\sum_{i \in Z} \left[\mathrm{LCS}\left(\mathrm{Trim}_{\Delta}(s_{m^*+1,i}), \mathrm{Trim}_{-\Delta}(t_{m^*+1,i})\right) + \Delta \right] \right] \geq |Z| \cdot (2L^* + 0.12\varepsilon L^*) \ .$$

[5] A negligible detail: we do not need to shift the endpoints of substrings s_{mid} and t_{mid} by 1 because L is a power of two so εL is never an integer

Hence, we may fix a $\Delta \in [L^*/4]$ for which

$$\sum_{i \in Z} \mathrm{LCS}\left(\mathrm{Trim}_{\Delta}(s_{m^*+1,i}), \mathrm{Trim}_{-\Delta}(t_{m^*+1,i})\right) \geq |Z| \cdot (2L^* - \Delta + 0.12\varepsilon L^*) \ .$$

Thus, set Z satisfies the setup of Lemma 3.9 with $n' = n$ and $m' = m^* + 1$ and $L' = 2L^*$ and $\delta' = 0.06\varepsilon$. Hence, by Lemma 3.9,

$$\mathrm{LCS}(s,t) \geq \left(1 + \frac{0.06\varepsilon}{20}\right) L \geq (1 + 0.003\varepsilon)\,(0.5 - 0.001\varepsilon)|s| > (0.5 + 0.0001\varepsilon)\,|s| \ . \qquad \square$$

3.8 Putting it all together

3.8.1 Statistics

We now prove our main theorem. The first step is to define the *statistics* of a string w.

Definition 3.25 (Statistics)**.** Let w be a string with $L = 2^n$ ones, and let $n_0 \stackrel{\text{def}}{=} \max(0, n - 200\gamma^{-3}\log n)$. Let the *statistics* of string w be a table of the following data:

1. For all $m \geq n_0$ and $i \geq 1$, the number of zeros and the number of ones in $w_{m,i}$ (the number of ones is always 2^m).
2. For all $m \geq n_0$ and $i \geq 1$, the type (see Definition 3.12) of string $w_{m,i}$.
3. The set $\mathcal{I}^{n_0}(w)$ of pairwise disjoint intervals $I \subset [L]$ that each have size $|I| \geq 2^{n_0}$ such that w_I is imbalanced for each $I \in \mathcal{I}^{n_0}(w)$, and the sum $\sum_{I \in \mathcal{I}^{n_0}(w)} |I|$ is maximized (if multiple such $\mathcal{I}^{n_0}(w)$ exist, break the tie arbitrarily).
4. For each $I \in \mathcal{I}^{n_0}(w)$, the indices x and y such that substring w_I starts at the x-th bit and ends at the y-th bit of s.

We say two strings s and t *agree on statistics* if their tables of statistics are identical.

This next lemma shows that it is possible to pigeonhole strings by their statistics, by showing that there are not too many possible statistics for a string.

Lemma 3.26. *There are at most* $2^{\mathrm{poly}\log N}$ *possible tables of statistics for a string of length* N*.*

Proof. A string w of length N has at most

$$\sum_{m=n_0}^{n} 2^{n-m} = \mathrm{poly}\log N$$

substrings $w_{m,i}$ that are considered in its table of statistics. Furthermore, for each substring $w_{m,i}$, there are at most N choices for the number of zeros and the number of ones, and at most $O(N)$ choices for the type of $w_{m,i}$ (there is one Imbalanced type, $O(N)$ Green types, and $O(\log N)$ Blue-Yellow types), so there are at most $2^{\mathrm{poly}\log N}$ choices for all the types and zero/one-counts in the table. Lastly, $\mathcal{I}^{n_0}(w)$ has at most $2^{n-n_0} = \log^{O_\varepsilon(1)} N$ intervals (because intervals have length at least 2^{n_0} and are disjoint), so there are at most $2^{\mathrm{poly}\log N}$ choices for $\mathcal{I}^{n_0}(w)$ and the locations of the endpoints of the intervals in $\mathcal{I}^{n_0}(w)$. $\square$

3.8.2 The Imbalanced case

This next lemma covers an easy case, when s has many large imbalanced intervals. In this case, our pigeonholing by statistics guarantees that s and t are imbalanced in common locations, allowing us to apply the imbalanced strategy to find a large LCS.

Lemma 3.27. *Let s and t be strings that each start with a one, agree on statistics, and have $L = 2^n$ ones where n is sufficiently large. Suppose there exists a set $\mathcal{I}$ of pairwise disjoint intervals I that each satisfy $|I| \geq 2^{n-200\gamma^{-3}\log n}$ such that s_I is imbalanced for each $I \in \mathcal{I}$ and $\sum_{I\in\mathcal{I}} |I| \geq \frac{\varepsilon^5}{10}L$. Then*

$$\mathrm{LCS}(s,t) \geq \left(0.5 + \frac{\varepsilon^6}{150}\right)|s|.$$

Proof. Let $n_0 = n - 200\gamma^{-3}\log n$. We may assume that $L \geq \frac{|s|}{3}$, or else s and t each have $\frac{2|s|}{3}$ zeros and $\mathrm{LCS}(s,t) \geq \frac{2}{3}|s| > (0.5 + \frac{\varepsilon^6}{150})|s|$. Since s and t agree on statistics, we have $\mathcal{I}^{n_0}(s) = \mathcal{I}^{n_0}(t)$. Furthermore, as $\mathcal{I}^{n_0}(s)$ maximizes the sum $\sum_{I\in\mathcal{I}^{n_0}(s)} |I|$, we have $\sum_{I\in\mathcal{I}^{n_0}(s)} |I| \geq \sum_{I\in\mathcal{I}} |I| \geq \frac{\varepsilon^5}{10}L$. Let $\mathcal{I}^{n_0}(s) = \mathcal{I}^{n_0}(t) = \{I_1, \ldots, I_k\}$, with $I_1 < I_2 < \cdots < I_k$ (these intervals are pairwise disjoint so the order is the obvious one). We thus may write

$$s = s''_0 s'_1 s''_1 s'_2 \cdots s'_k s''_k, \qquad t = t''_0 t'_1 t''_1 t'_2 \cdots t'_k t''_k$$

where $s'_j = s_{I_j}$ for $j \in [1,k]$, and substring s''_j consists of the bits between the end of substring s'_j (or the beginning of string s if $j = 0$) and the beginning of substring s'_{j+1} (or the end of string s if $j = k$), and the partition of t is defined analogously. By the definitions of s_{I_j} and t_{I_j}, for $j \in [\![1,k]\!]$, s'_j and t'_j have the same number $|I_j|$ of ones, and for $j \in [\![0,k]\!]$, s''_j and t''_j have the same number of ones as well. Further, since s and t agree on statistics, for all $j \in [\![1,k]\!]$, substrings s_{I_j} and t_{I_j} start and end in the same positions in their respective strings s and t. In particular, s'_j and t'_j have the same length for all $j \in [\![1,k]\!]$, and s''_j and t''_j have the same length for all $j \in [\![0,k]\!]$.

For each $j \in [\![0,k]\!]$, the substrings s''_j and t''_j have the same length and the same number of ones, so $\mathrm{LCS}(s''_j, t''_j) \geq |s''_j|/2$. Additionally, for $j \in [\![1,k]\!]$ the number of zeros in substrings s'_j and t'_j are equal and not in $(1 \pm \varepsilon)|s'_j|$. Hence, by Lemma 3.8, we have $\mathrm{LCS}(s'_j, t'_j) \geq (1/2 + \varepsilon/5)|s'_j|$. Therefore we have

$$\begin{aligned}\mathrm{LCS}(s,t) &\geq \sum_{j=1}^{k} \mathrm{LCS}(s'_j, t'_j) + \sum_{j=0}^{k} \mathrm{LCS}(s''_j, t''_j) \;\geq \sum_{j=1}^{k} \left(\frac{1}{2} + \frac{\varepsilon}{5}\right)|s'_j| + \sum_{j=0}^{k} \frac{|s''_j|}{2} \\ &= \frac{|s|}{2} + \frac{\varepsilon}{5}\sum_{j=1}^{k} |s'_j| \;\geq \frac{|s|}{2} + \frac{\varepsilon^6 L}{50} \;\geq \left(\frac{1}{2} + \frac{\varepsilon^6}{150}\right)|s|,\end{aligned}$$

where the last two inequalities used that $|s'_j| \geq |I_j|$, $\sum_{j=1}^{k} |I_j| \geq \frac{\varepsilon^5 L}{10}$, and $L \geq \frac{|s|}{3}$. □

3.8.3 Combining the arguments for the Imbalanced, Green, and Blue-Yellow cases

We now prove the main technical lemma, which shows that two strings that agree on statistics have LCS beating the 1/2 barrier. This establishes Theorem 3.1 up to a pigeonhole argument and an assumption

about the number of ones being a power of two. The proof consists of piecing together (1) the Imbalanced case, when s and t have many substrings of Imbalanced type, Lemma 3.27, (2) the Green case, when s and t have many substrings of Green type, and (3) the Blue-Yellow case, when s and t have many substrings of Blue-Yellow type. These three cases correspond to the three matching strategies stated in the Overview (Section 3.2).

Lemma 3.28. *There exists an absolute constant $\delta = \frac{\varepsilon^6}{150}$ such that the following holds for n sufficiently large. Let s and t be strings that each start with a one, have $L = 2^n$ ones each, such that s and t agree on statistics, and $\mathrm{rev}(s)$ and $\mathrm{rev}(t)$ agree on statistics. Then $\mathrm{LCS}(s,t) \geq (0.5+\delta)|s|$.*

Proof. First, suppose that at least $\gamma^2 L$ values of $i \in [L]$ satisfy $b_s(i) \geq 2^{n-200\gamma^{-3}\log n}$. Define $1 \leq i_1 < \cdots < i_d$ such that i_k is the smallest index such that $b_s(i_k) \geq 2^{n-200\gamma^{-3}\log n}$ and $i_k \geq i_{k-1} + b_s(i_{k-1})$ if $k > 1$, and d is the largest index such that i_d is well-defined. By the definition of $b_s(i_k)$, index i_k is a Blue $b_s(i_k)$-flag in s, so for the interval $I_k \stackrel{\text{def}}{=} [\![i_k, \min(i_k + b_s(i_k) - 1, L)]\!]$, substring s_{I_k} is imbalanced by Lemma 3.7. Furthermore, since $i_k + b_s(i_k) \leq i_{k+1}$ for $k = 1, \ldots, d-1$, we have $I_1, \ldots, I_d$ are pairwise disjoint. Lastly, by minimality of each i_k, each index $i \in [L]$ with $b_s(i) \geq 2^{n-200\gamma^{-3}\log n}$ is in some interval I_k. Thus, $\sum |I_k| \geq \gamma^2 L > \frac{\varepsilon^5}{10}$. Thus, we may apply Lemma 3.27 to strings s and t, giving $\mathrm{LCS}(s,t) \geq (0.5+\delta)|s|$. Hence, we may assume for the rest of the argument that

- At most $\gamma^2 L$ values of $i \in [L]$ satisfy $b_s(i) \geq 2^{n-200\gamma^{-3}\log n}$.

Similarly, we may assume

- At most $\gamma^2 L$ values of $i \in [L]$ satisfy $b_{\mathrm{rev}(s)}(i) \geq 2^{n-200\gamma^{-3}\log n}$ (applying Lemma 3.27 to $\mathrm{rev}(s)$ and $\mathrm{rev}(t)$),
- At most $\gamma^2 L$ values of $i \in [L]$ satisfy $b_t(i) \geq 2^{n-200\gamma^{-3}\log n}$ (applying Lemma 3.27 to t and s), and
- At most $\gamma^2 L$ values of $i \in [L]$ satisfy $b_{\mathrm{rev}(t)}(i) \geq 2^{n-200\gamma^{-3}\log n}$ (applying Lemma 3.27 to $\mathrm{rev}(t)$ and $\mathrm{rev}(s)$).

Hence, we may apply the Substring Blue-flag-balance Lemma, Lemma 3.19, to each of the strings s, $\mathrm{rev}(s)$, t, and $\mathrm{rev}(t)$ with $\beta = \gamma$. This shows the existence of some scale m^* with $n - 10 - \log \delta^{-1} \geq m^* \geq n - 150\gamma^{-3}\log n$ such that the following hold:

- For at least $(1-3\gamma)\cdot 2^{n-m^*}$ values of $i \in [2^{n-m^*}]$, the substring $s_{m^*,i}$ is 6γ-Blue-flag-balanced,
- For at least $(1-3\gamma)\cdot 2^{n-m^*}$ values of $i \in [2^{n-m^*}]$, the substring $\mathrm{rev}(s_{m^*,i})$ is 6γ-Blue-flag-balanced,
- For at least $(1-3\gamma)\cdot 2^{n-m^*}$ values of $i \in [2^{n-m^*}]$, the substring $t_{m^*,i}$ is 6γ-Blue-flag-balanced, and
- For at least $(1-3\gamma)\cdot 2^{n-m^*}$ values of $i \in [2^{n-m^*}]$, the substring $\mathrm{rev}(t_{m^*,i})$ is 6γ-Blue-flag-balanced.

Indeed, for each bullet, Lemma 3.19 implies there are at most $32\gamma^{-3}\log n$ values of m^* for which the condition does not hold, so among $150\gamma^{-3}\log n - 10 - \log\delta^{-1} > 128\gamma^{-3}\log n$ values of m^*, some m^* allows all four conditions to hold. Fix this m^* and let

$$L^* \stackrel{\text{def}}{=} 2^{m^*}\ .$$

Since the number of ones in s and t are a power of two, s and t agree on statistics, and $m^* > n_0$, where n_0 is from Definition 3.25, we have $s = s_{m^*,1}\cdots s_{m^*,2^{n-m^*}}$ and $t = t_{m^*,1}, \ldots, t_{m^*,2^{n-m^*}}$, where, for all

$i = 1, \ldots, 2^{n-m^*}$, substrings $s_{m^*,i}$ and $t_{m^*,i}$ have the same number L^* of ones and the also the same number of zeros, and thus have the same length. Similarly, we may write $\text{rev}(s) = \text{rev}(s_{m^*,2^{n-m^*}}) \cdots \text{rev}(s_{m^*,1})$ and $\text{rev}(t) = \text{rev}(t_{m^*,2^{n-m^*}}) \cdots \text{rev}(t_{m^*,1})$.

Since n is sufficiently large, for all $i \in [2^{n-m^*}]$, substrings $s_{m^*,i}$ and $\text{rev}(s_{m^*,i})$ have types by Lemma 3.13. Let Z_1 (resp. $\bar{Z}_1$) be the set of $i \in [2^{n-m^*}]$ such that substring $s_{m^*,i}$ (resp. $\text{rev}(s_{m^*,i})$) is Imbalanced. Let Z_2 (resp. $\bar{Z}_2$) be the set of $i \in [2^{n-m^*}]$ such that substring $s_{m^*,i}$ (resp. $\text{rev}(s_{m^*,i})$) is ℓ-Green for some ℓ. Let Z_3 (resp. $\bar{Z}_3$) be the set of $i \in [2^{n-m^*}]$ such that substring $s_{m^*,i}$ (resp. $\text{rev}(s_{m^*,i})$) is m-Blue-Yellow for some m. Since $|Z_1| + |Z_2| + |Z_3| = |\bar{Z}_1| + |\bar{Z}_2| + |\bar{Z}_3| = 2^{n-m^*}$, we have the following cases covering all possibilities.

Case 1a. $|Z_1| \geq 2^{n-m^*}/10$. In this case, for each $i \in Z_1$, because $s_{m^*,i}$ is type Imbalanced, there exists some interval $J_i \subset I_{m^*,i}$ with $|J_i| \geq \varepsilon^5 L^* > 2^{n-200\gamma^{-3}\log n}$ such that s_{J_i} is imbalanced. Since the intervals $I_{m^*,i}$ are pairwise disjoint, the intervals J_i are pairwise disjoint. Then setting $\mathcal{I}' = \{J_i : i \in Z_1\}$, we have that $\sum_{J \in \mathcal{I}'} |J| \geq \varepsilon^5 L^* \cdot |Z_1| = \frac{\varepsilon^5}{10} \cdot L$. Hence, by Lemma 3.27, we have $\text{LCS}(s,t) \geq (0.5 + \frac{\varepsilon^6}{150})|s| = (0.5+\delta)|s|$.

Case 1b. $|\bar{Z}_1| \geq 2^{n-m^*}/10$. By an identical argument to Case 1a, we can show $\text{LCS}(\text{rev}(s), \text{rev}(t)) \geq (0.5 + \delta)|s|$, which implies that $\text{LCS}(s,t) \geq (0.5 + \delta)|s|$.

Case 2a. $|Z_2| \geq 2^{n-m^*}/10$. Since $m^* \leq n - 10 - \log \varepsilon^{-5}$ by definition of m^*, we may apply Lemma 3.21 to strings s and t with subset $Z \subset [2^{n-m^*}]$. By Lemma 3.21, we have $\text{LCS}(s,t) \geq (0.5 + \frac{\varepsilon^5}{5000})|s| > (0.5+\delta)|s|$.

Case 2b. $|\bar{Z}_2| \geq 2^{n-m^*}/10$. By an identical argument to Case 2a, we can show $\text{LCS}(\text{rev}(s), \text{rev}(t)) \geq (0.5 + \delta)|s|$, which implies that $\text{LCS}(s,t) \geq (0.5 + \delta)|s|$.

Case 3. $|Z_3| \geq \frac{4}{5} \cdot 2^{n-m^*}$ and $|\bar{Z}_3| \geq \frac{4}{5} \cdot 2^{n-m^*}$. Let Z_3' be the set of $i \in [2^{n-m^*-1}]$ such that the following hold:

- Substring $s_{m^*,2i-1}$ is 6γ-Blue-flag-balanced.
- Substring $t_{m^*,2i-1}$ is 6γ-Blue-flag-balanced.
- Substring $\text{rev}(s_{m^*,2i})$ is 6γ-Blue-flag-balanced.
- Substring $\text{rev}(t_{m^*,2i})$ is 6γ-Blue-flag-balanced.
- We have $2i - 1 \in Z_3$.
- We have $2i \in \bar{Z}_3$.

By choice of m^*, the first four conditions above each fail for at most $3\gamma \cdot 2^{n-m^*}$ values of $i \in [2^{n-m^*-1}]$. Since $|Z_3| \geq \frac{4}{5} \cdot 2^{n-m^*}$, the fifth condition fails for at most $\frac{1}{5} \cdot 2^{n-m^*}$ values of i, and similarly the last condition fails for at most $\frac{1}{5} \cdot 2^{n-m^*}$ values of i. Since there are $\frac{1}{2} \cdot 2^{n-m^*}$ values of $i \in [2^{n-m^*-1}]$, we have that Z_3' has size at least $(\frac{1}{2} - 12\gamma - \frac{2}{5}) \cdot 2^{n-m^*} > \frac{1}{10} \cdot 2^{n-m^*-1}$ (recall $\gamma = 10^{-15}$ is very small).

Fix $i \in Z_3'$. We claim that strings s and t with parameter m^* and set Z satisfy the setup of Lemma 3.24. The bound on $m^* \leq n - 10 - \log \varepsilon^{-1}$ follows from the definition of m^*. We thus need to show that, for all $i \in Z_3'$, the pairs $(s_{m^*,2i-1}, t_{m^*,2i-1})$ and $(\text{rev}(s_{m^*,2i}), \text{rev}(t_{m^*,2i}))$ are a Blue-Yellow pairs. Since $2i - 1 \in Z_3$, there exists some integer m such that substring $s_{m^*,2i-1}$ is type m-Blue-Yellow, and since s and t agree on statistics, substring $t_{m^*,2i-1}$ is also type m-Blue-Yellow. Since $i \in Z_3'$, we have that strings $s_{m^*,2i-1}$ and $t_{m^*,2i-1}$ are both 6γ-Blue-flag-balanced, so $(s_{m^*,2i-1}, t_{m^*,2i-1})$ form a Blue-Yellow pair.

Similarly, since $2i \in \bar{Z}_3$ and $\mathrm{rev}(s)$ and $\mathrm{rev}(t)$ agree on statistics, there exists some integer m' such that substrings $\mathrm{rev}(s_{m^*,2i}) = \mathrm{rev}(s)_{m^*,2^{n-m^*}-(2i-1)}$ and $\mathrm{rev}(t_{m^*,2i})$ have type m'-Blue-Yellow. Since $i \in Z'_3$, we have that strings $\mathrm{rev}(s_{m^*,2i})$ and $\mathrm{rev}(t_{m^*,2i})$ are both 6γ-Blue-flag-balanced, so $(\mathrm{rev}(s_{m^*,2i}), \mathrm{rev}(t_{m^*,2i}))$ form a Blue-Yellow pair, as desired. Thus, by Lemma 3.24, we have

$$\mathrm{LCS}(s,t) \geq (0.5 + 0.0001\varepsilon)\,|s| > (0.5+\delta)\,|s|.$$

In all cases we have shown that $\mathrm{LCS}(s,t) \geq (0.5+\delta)|s|$, proving the lemma. □

3.8.4 Finishing the proof

To prove the main theorem, we need to find two strings that agree on statistics, and remove the assumption that the number of ones is a power of two.

Theorem 3.29. *There exists absolute constants $A > 0$ and $\delta = \frac{\varepsilon^6}{900}$ such that the following holds for N sufficiently large. Let $\mathcal{C} \subset \{0,1\}^N$ be a code with at least $2^{\log^A N}$ strings. Then $\mathcal{C}$ contains two strings s and t such that $\mathrm{LCS}(s,t) \geq (0.5+\delta)N$.*

Proof. By Lemma 3.26, there exists a constant A' such that the number of possible tables of statistics for a string of length N is at most $2^{O(\log^{A'} N)}$. We pick $A = A'+1$. By removing at most half of the elements of $\mathcal{C}$, we may assume that every string in $\mathcal{C}$ starts with the same bit, and without loss of generality we may assume that every string starts with a one. Let 2^n be the largest power of two less than $N/3$, so that $2^n \geq N/6$. If there exist two strings s and t in $\mathcal{C}$ with less than 2^n ones, then we have $\mathrm{LCS}(s,t) \geq 2N/3$, so we may assume that all but at most one string in $\mathcal{C}$ has at least 2^n ones.

By the pigeonhole principle, there exist two strings $s, t \in \mathcal{C}$ such that

- s and t have the same number $L \geq 2^n$ of ones,
- $s_{n,1}$ and $t_{n,1}$ agree on statistics, and
- $\mathrm{rev}(s_{n,1})$ and $\mathrm{rev}(t_{n,1})$ agree on statistics.

Indeed, the total number of tables of statistics for each of substrings $s_{n,1}$ and $\mathrm{rev}(s_{n,1})$ is at most $2^{O(\log^{A'} N)}$, so the total number of pigeonholes here is at most $2^{O(\log^{A'} N)} < 2^{\log^A N} = |\mathcal{C}|$ if N is sufficiently large.

Let $s = s_{n,1}s'$ and $t = t_{n,1}t'$. Since strings s and t have the same length and same number of ones, and substrings $s_{n,1}$ and $t_{n,1}$ agree on statistics, we have that the suffixes s' and t' have the same length and the same number of ones as well. Thus, $\mathrm{LCS}(s',t') \geq |s'|/2$. By applying Lemma 3.28 to strings $s_{n,1}$ and $t_{n,1}$, we have

$$\mathrm{LCS}(s_{n,1}, t_{n,1}) \geq \left(0.5 + \frac{\varepsilon^6}{150}\right)|s_{n,1}|.$$

Hence, we have

$$\mathrm{LCS}(s,t) \geq \mathrm{LCS}(s_{n,1},t_{n,1}) + \mathrm{LCS}(s',t') \geq \left(0.5+\frac{\varepsilon^6}{150}\right)|s_{n,1}| + 0.5|s'| = 0.5N + \frac{\varepsilon^6}{150}|s_{n,1}| \geq (0.5+\delta)N,$$

as desired. In the last inequality, we used that $|s_{n,1}| \geq 2^n \geq N/6$. □

Acknowledgments.

This chapter appeared as [50] and is joint work with Venkatesan Guruswami and Xiaoyu He. We thank Jacob Fox for helpful conversations and the reference [5].

Chapter 4

List decoding: binary alphabet

4.1 Introduction

In this chapter, we consider list-decoding over binary alphabets. Recall that a binary code $\mathcal{C} \subset \mathbb{F}_2^n$ is (p, L)-list-decodable if any Hamming ball of radius pn in $\mathbb{F}_2^n$ contains at most L points of $\mathcal{C}$: that is, if for all $x \in \mathbb{F}_2^n$, $|\mathcal{B}(x, pn) \cap \mathcal{C}| \leq L$, where $\mathcal{B}(x, pn)$ is the Hamming ball of radius pn centered at x. Recall the *list-decoding capacity theorem (Theorem 2.9):*

Theorem 4.1 (List decoding capacity theorem)**.** *Let $p \in (0, 1/2)$ and $\varepsilon > 0$.*

1. *There exist binary codes of rate $1 - h(p) - \varepsilon$ that are $(p, \lceil 1/\varepsilon \rceil)$-list decodable.*
2. *Any binary code of rate $1 - h(p) + \varepsilon$ that is (p, L)-list decodable up to distance p must have $L \geq 2^{\Omega(\varepsilon n)}$.*

As shown in Chapter 2, the existential part of Theorem 4.1 is proved by showing that a uniformly random subset of $\mathbb{F}_2^n$ is $(p, 1/\varepsilon)$-list decodable with high probability. For a long time, uniformly random codes were the only example of binary codes known to come close to this bound, and today we still do not have many other options. There are explicit constructions of capacity-achieving list-decodable codes over large alphabets (either growing with n or else large-but-constant) [33, 71, 72], but over binary alphabets we still do not have any explicit constructions; we refer the reader to the survey [46] for an overview of progress in this area.

Because it is a major open problem to construct explicit binary codes of rate $1 - h(p) - \varepsilon$ with constant (or even poly(n)) list-sizes, one natural line of work has been to study structured random approaches, in particular *random linear codes.* A random linear code $\mathcal{C} \subset \mathbb{F}_2^n$ is simply a random subspace of $\mathbb{F}_2^n$, and the list-decodability of these codes has been well-studied [129, 49, 48, 23, 126, 108, 111]. There are several reasons to study the list-decodability of random linear codes. Not only is it a natural question in its own right as well as a natural stepping stone in the quest to obtain explicit binary list-decodable codes, but also the list-decodabilility of random linear codes is useful in other coding-theoretic applications. One example of this is in concatenated codes and related constructions [54, 63, 80, 79], where a random linear code is used as a short inner code. Here, the linearity is useful because (a) a linear code can be efficiently described; (b) it is sometimes desirable to obtain a linear code at the end of the day, hence all components of the construction must be linear; and (c) as in [80] sometimes the linearity is required for the construction to work.

To this end, the line of work mentioned above has aimed to establish that random linear codes are "as list-decodable" as uniformly random codes. That is, uniformly random codes are viewed (as is often the

case in coding theory) as the optimal construction, and we try to approximate this optimality with random linear codes, despite the additional structure.

Our contributions. In this paper, we give an improved analysis of the list-decodability of random linear binary codes. More precisely, our contributions are as follows.

- **A unified analysis.** As we discuss more below, previous work on the list-decodability of random linear binary codes either work only in certain (non-overlapping) parameter regimes [48, 126], or else get substantially sub-optimal bounds on the list-size [111]. Our argument obtains improved list size bounds over all these results and works in all parameter regimes.

  Our approach is surprisingly simple: we adapt an existential argument of Guruswami, Håstad, Sudan and Zuckerman [49] to hold with high probability. Extending the argument in this way was asked as an open question in [49] and had been open until now.

- **Improved list-size for random linear codes.** Not only does our result imply that random linear codes of rate $1 - h(p) - \varepsilon$ are (p, L)-list-decodable with list-size $L = O(1/\varepsilon)$, in fact we show that $L \leq h(p)/\varepsilon + 2$. In particular, the leading constant is small and—to the best of our knowledge—is the best known, even existentially, for any list-decodable code.

- **Better lower bounds on the list size for list-decoding random linear codes.** We show a matching lower bound to our upper bound: if a binary random linear code of rate $1 - h(p) - \varepsilon$ is list-decodable with high probability up to radius p with an output list size of L, then we must have $L \geq \lfloor \frac{h(p)}{\varepsilon} - 0.01 \rfloor$.

  Previous work [61] has established that $L = \Omega(1/\varepsilon)$, but to the best of our knowledge this is the first work that pins down the leading constant. In particular, [61] shows that, in the situation above, we have $L \geq c_p/\varepsilon$, where c_p is a constant that goes to zero as p goes to $1/2$. In contrast, we show below that the leading constant is at least $h(p)$, which goes to 1 as p goes to $1/2$.

4.2 Previous Work and Our Results

Below, we survey related work on the list-decodability of random linear binary codes, state our results, describe a few generalizations and variations provable with our techniques, and highlight some related work.

4.2.1 Random Linear Codes

The list-decodability of random linear binary codes has been well studied. Here we survey the results that are most relevant for this chapter. As this chapter focuses on binary codes, we focus this survey on results for binary codes, even though many of the works mentioned also apply to general q-ary codes. We additionally remark that, in contrast to the large alphabet setting [64], capacity achieving binary codes have no known explicit constructions.

A modification of the proof of the list decoding capacity theorem shows that a random linear code of rate $1 - h(p) - \varepsilon$ is $(p, \exp(O(\frac{1}{\varepsilon})))$-list decodable [129]. However, whether or not random linear codes of this rate with list-sizes that do not depend exponentially on ε remained open for decades: this question was explicitly asked in [35].

A first step was given in the work of Guruswami, Håstad, Sudan and Zuckerman [49], who proved via a beautiful potential-function-based-argument that there *exist* binary linear codes or rate $1 - h(p) - \varepsilon$ which are $(p, 1/\varepsilon)$-list-decodable. However, this result did not hold with high probability. Our approach relies heavily on the approach of [49], and we return to their argument in §4.4.

Over the next 15 years, a line of work [48, 23, 126, 108, 110, 111] has focused on the list-decodability (and related properties) of random linear codes, which should hold with high probability. The works most relevant to ours are [48, 126], which together more or less settle the question. We state these results here for binary alphabets, although both works address larger alphabets as well.

The first result, of [48], establishes a result for a constant p, bounded away from $1/2$.

Theorem 4.2 (Theorem 2 of [48])**.** *Let $p \in (0, 1/2)$. Then there exist constants $C_p, \delta > 0$ such that for all $\varepsilon > 0$ and sufficiently large n, for all $R \leq 1 - h(p) - \varepsilon$, if $\mathcal{C} \subseteq \mathbb{F}_2^n$ is a random linear code of rate R, then $\mathcal{C}$ is $(p, C_p/\varepsilon)$-list decodable with probability at least $1 - 2^{-\delta n}$.*

However, C_p is not small and tends to ∞ as p approaches $1/2$. The following result of [126] fills in the gap when p is quite close to $1/2$.

Theorem 4.3 (Theorem 2 of [126])**.** *There exist constants C_1, C_2 so that for all sufficiently small $\varepsilon > 0$ and sufficiently large n, for $p = 1/2 - C_1\sqrt{\varepsilon}$ and for all $R \leq 1 - h(p) - \varepsilon$, if $\mathcal{C} \subseteq \mathbb{F}_2^n$ is a random linear code of rate R, then $\mathcal{C}$ is $(p, C_2/\varepsilon)$-list decodable with probability at least $1 - o(1)$.*

4.2.2 Our list-size upper bounds for random linear codes

We show that high probability a random linear binary code of rate $1 - h(p) - \varepsilon$ is (p, L)-list-decodable with $L \sim h(p)/\varepsilon$. More precisely, the upper bound is as follows (proved in §4.4).

Theorem 4.4. *Let $\varepsilon > 0$ and $p \in (0, 1/2)$. A random linear code of rate $1 - h(p) - \varepsilon$ is $(p, h(p)/\varepsilon + 1)$-list decodable with probability $1 - \exp(-\Omega_\varepsilon(n))$.*

Theorem 4.4 improves upon the picture given by Theorems 4.2 and 4.3 in two ways. First, the leading constant on the list size, which is $h(p)$, improves over both the constant C_p from Theorem 4.2 (which blows up as $p \to 1/2$) and on the constant C_2 from Theorem 4.3 (which the authors do not see how to make less than 2). Moreover, when $p \to 1/2$, Theorem 4.4 improves on Theorem 4.3 in that it decouples p from ε: in Theorem 4.3, we must take $p = 1/2 - O(\sqrt{\varepsilon})$ and $R = 1 - h(p) - \varepsilon$, while in Theorem 4.4, p and ε may be chosen independently. Thus, Theorem 4.4 offers the first true "list-decoding capacity theorem for binary linear codes," in that it precisely mirrors the quantifiers in Theorem 4.1.

The list size of $h(p)/\varepsilon + 1$ is smaller than the list size of $1/\varepsilon$ given by the classical list decoding capacity theorem for uniformly random codes. Guruswami and Narayanan [61] showed that uniformly random codes have list-size $\Omega(1/\varepsilon)$. By tightening their second moment method proof (see Appendix A of [93]), we can in fact show that the list the list size of $1/\varepsilon$ given by uniformly random binary codes in the list decoding capacity theorem is tight, even in the leading constant of 1. That is, the "correct" list size of a uniformly random code is tightly concentrated between $\lfloor 1/\varepsilon \rfloor \pm 1$ for small ε. Thus, for all $p \in (0, 1/2)$ and sufficiently small ε, random linear codes of rate $1 - h(p) - \varepsilon$ with high probability can be list decoded up to distance p with smaller list sizes than uniformly random codes.

We are unaware of results in the literature that give even the existence of binary codes list decodable with list size *better* than $h(p)/\varepsilon$. We remark that the Lovasz Local Lemma also gives the *existence* of $(p, h(p)/\varepsilon)$ list decodable codes, matching our high probability result for random linear codes (see Appendix C of [93]).

4.2.3 Our lower bounds: Pinning down the output list size.

Pinning down the output list size L is an important problem. For example, for many of the algorithmic applications within coding theory, the list size represents a bottleneck on the running time of an algorithm that must check each item in the list before pruning it down [54, 33, 71, 72, 55]. For applications in pseudorandomness, for example to expanders or extractors, the list size corresponds to the expansion or to the amount of entropy in the input, respectively, and it is of interest to precisely pin down these quantities.

Motivated by pinning down the list size, we also prove an essentially tight lower bound on the list-size of random linear binary codes.

Theorem 4.5. *Fix $p \in (0, 1/2)$, and fix $\delta \in (0, 1)$. There exists $\varepsilon_{p,\delta} > 0$ such that for all $\varepsilon \in (0, \varepsilon_{p,\delta})$ and n sufficiently large, a random linear code in $\mathbb{F}_2^n$ of rate $1 - h(p) - \varepsilon$ is* not *$\left(p, \lfloor \frac{h(p)}{\varepsilon} - \delta \rfloor - 1\right)$-list-decodable with probability $1 - 2^{-\Omega(n)}$.*

Previously, Guruswami and Narayanan [61] showed that lists of size c_p/ε are necessary, and we improve this lower bound to essentially $h(p)/\varepsilon$. The constant c_p is not explicitly computed, but one can deduce from the proof that if p tends to $1/2$ then c_p will tend to 0. Their lower bound follows from a second moment method argument, i.e., they consider a certain random variable X whose positivity is equivalent to the failure of a random linear code to be list-decodable, and then show that $\mathbf{Var}\, X = o(\mathbf{E}\, X)^2$. In this sense our approach is similar to theirs, because we rely on results from [101] which themselves are proved using a second moment method. However, we are able to get stronger results (in the sense that our leading constant does not decay as $p \to 1/2$, and moreover is optimal for binary codes). One of the reasons may be the notion of "implicit rareness" from [101], which provides a useful characterization of the lists contained in a random linear code.

4.2.4 Generalizations and variations

The techniques in Theorem 4.4 and Theorem 4.5 can be applied to obtain variations and generalizations, which appear in [93, 59]. We informally state the variations and generalizations here, and refer the reader to [93, 59] for the proofs.

Larger alphabets. One naturally might wonder whether the upper and lower bounds in this chapter also extend beyond binary codes to codes over all constant-sized fields. We do not know how to extend the list-size upper bounds in Theorem 4.4 to nonbinary codes, but the list-size lower bounds in Theorem 4.5 generalize to similar lower bounds over nonbinary fields. Over alphabets of size q, the list-decoding capacity is $1 - h_q(p)$, where $h_q(x) \stackrel{\text{def}}{=} x \log_q(q-1) - x \log_q x - (1-x) \log_q(1-x)$ is the *q-ary entropy*, and we show random-linear codes of rate $1 - h_q(p) - \varepsilon$ cannot have list size better $\frac{h_q(p)}{\varepsilon}$ (see [59]). We conjecture that for any finite field size q the list size of $\frac{h_q(p)}{\varepsilon}$ is optimal, as we show it is for $q = 2$.

Average radius list-decodability. Our upper bound can be generalized to hold for a strengthening of list-decoding known as *average-radius list-decoding*. A code $\mathcal{C}$ is *(p, L)-average-radius list-decodable* if for any

set $\Lambda \subseteq \mathcal{C}$ of size $L+1$ and $z \in \mathbb{F}_q^n$,

$$\frac{1}{L+1}\sum_{c\in\Lambda} \Delta_H(c,z) > pn.$$

It is not hard to see that (p, L)-average-radius list-decodability implies (p, L)-list-decodability, and this stronger formulation has led to stronger lower bounds than are achievable otherwise [61]. In addition to stronger lower bounds, average-radius list-decoding—essentially replacing a maximum with an average in the definition of list-decoding—is a natural concept, and it has helped establish connections between list-decoding and compressed sensing [23].

A modification of our list-size upper bound in Theorem 4.4 can give Theorem 4.4 for average-radius list-decodability. That is, we can show that a random linear code of rate $1 - h(p) - \varepsilon$ is $(p, h(p)/\varepsilon + 1)$-*average-radius*-list-decodable with probability $1 - \exp(-\Omega_\varepsilon(n))$. See [59] for proof.[1]

List-recovery. *List-recovery* is a variation of list-decoding where the "noise" is replaced by uncertainty about each symbol of the received word z. Formally, we say that a code $\mathcal{C}$ is (ℓ, L)-*list-recoverable* if for any sets $S_1, \ldots, S_n \subseteq \mathbb{F}_q$ with $|S_i| \leq \ell$ for all i,

$$|\{c \in \mathcal{C} \,\mathrm{s.t.}\, c_i \in S_i \,\forall i\}| < L.$$

List-recovery was originally used as a stepping-stone to list-decoding and unique-decoding (e.g., [51, 52, 53, 54]) but it has since become a useful primitive in its own right, with applications beyond coding theory [81, 103, 39, 76, 31].

Our lower bound techniques in Theorem 4.5 also give list-recovery lower bounds for random linear codes. In the list-recovery setting, we can show a random linear code of rate $1 - \log_q(\ell) - \varepsilon$ requires output list size $L \geq \ell^{\Omega(1/\varepsilon)}$ for list-recovery from input list size ℓ, which is in surprising contrast to completely random codes, where output list size $L = O(\ell/\varepsilon)$ suffices with high probability. See [59].

Rank metric codes. Rank metric codes, introduced by Delsarte in [27], are codes $\mathcal{C} \subseteq \mathbb{F}_q^{m\times n}$; that is, the codewords are $m \times n$ matrices, where typically $m \geq n$. The distance between two codewords X and Y is given by the rank of their difference: $\Delta_R(X, Y) \stackrel{\text{def}}{=} \frac{1}{n}\operatorname{rank}(X - Y)$, where Δ_R is called the *rank metric*.

Our upper bound argument can be adapted to random linear rank-metric codes (see [93]). As with standard (Hamming-metric) codes, recent work aimed to show that random linear rank-metric codes are nearly as list-decodable as uniformly random codes [30, 62]. Our approach establishes that in fact, random linear binary rank-metric codes are more list-decodable than their uniformly random counterparts in certain parameter regimes, in the sense that the list sizes near capacity are strictly smaller. Along the way, we show that low-rate random linear binary rank-metric codes are list-decodable to capacity, answering a question of [62].

4.2.5 Related work

We now highlight some related work.

[1] We note that in the definition of list-decoding in [59], Hamming balls have strictly less than L codewords, rather than at most L codewords, so the list sizes in [59] differ by 1 from our list sizes.

Lower bounds for list sizes of arbitrary codes. It is known that a typical (i.e., uniformly random) list-decodable code of rate $1 - h(p) - \varepsilon$ has list size $L = \Theta(1/\varepsilon)$, and a natural question to ask is whether *every* code requires a list of size $L = \Omega(1/\varepsilon)$. Blinovsky ([11, 10]) showed that lists of size $\Omega_p(\log(1/\varepsilon))$ are necessary for list-decoding a code of rate $1 - h(p) - \varepsilon$. Later, Guruswami and Vadhan [69] considered the high-noise regime where $p = 1 - 1/q - \eta$ and showed that lists of size $\Omega_q(1/\eta^2)$ are necessary. Finally, Guruswami and Narayanan [61] showed that for *average-radius* list-decoding, the list size must be $\Omega_p(1/\sqrt{\varepsilon})$.

Relevant results for other ensembles of codes. Lastly, we discuss some other results concerning other code ensembles. First of all, uniformly random codes are known to achieve list-decoding capacity with high probability, with list size roughly $1/\varepsilon$, and this list size is tight for binary codes [93]. Comparing with the above results, random linear binary codes have smaller list sizes than random codes by a factor of roughly $h(p)$, while for larger alphabet sizes, the best list size bounds for random linear codes are a constant factor worse than the $1/\varepsilon$ list size of random codes. Random codes exhibit sharp threshold behavior for a number of "symmetric" properties including list-decoding, list average-radius list-decoding, list-recovery, and perfect hashing [60].

Recent work of [101] shows that a random code from Gallager's ensemble of LDPC codes [38] achieves list-decoding capacity with high probability. More generally, they show that random LDPC codes inherit combinatorial properties from random linear codes, including list-decoding, average-radius list-decoding, and list-recovery. Consequently, our positive results for binary random linear codes also apply to binary LDPC codes. Furthermore, as part of their approach, they develop techniques to characterize the lists that appear in a random linear code with high probability, which we utilize for this chapter.

Finally, we note that there are no known explicit constructions of list-decodable codes of rate $1 - h(p) - \varepsilon$ which achieve a list size even of $O(1/\varepsilon)$. Over large alphabets, the best explicit constructions of capacity-achieving list-decodable or list-recoverable codes have list sizes at least $(1/\varepsilon)^{\Omega(1/\varepsilon)}$ (e.g., [89, 88]). Further, if one insists on *binary* codes, or even codes over alphabets of size independent of ε, we do not know of *any* explicit constructions of list-decodable codes with rate approaching $1 - h(p)$.

Two-point concentration. We showed that the optimal list size L of a random linear code is concentrated on at most three values for both list-decoding and average-radius list-decoding: $\lfloor h_2(p)/\varepsilon \rfloor + 1, \lfloor h_2(p)/\varepsilon \rfloor$, and, if the value is different, $\lfloor h_2(p)/\varepsilon - 0.01 \rfloor$.

In [93, Theorem 2.5], it was also shown that the optimal list size of a *completely* random binary code is concentrated on two or three values for list-decoding. This type of concentration is also well studied in graph theory, where it is known that in Erdős-Rényi graphs, a number of graph parameters are concentrated on two values. Examples include the clique number (size of the largest clique) [98, 12], the chromatic number [94, 2, 1], and the diameter [107].

4.3 Preliminaries

We define a *random linear binary code of rate* R to be the span of $k = Rn$ independently random vectors $b_1, \ldots, b_k \in \mathbb{F}_2^n$. One can easily check the vectors $b_1, \ldots, b_k$ are linearly independent with probability at least $1 - 2^{-(n-k)}$, so the dimension of this code is indeed k with high probability. Given a code $\mathcal{C}$ and $p \in (0, 1/2)$, we define the list size of a point $x \in \mathbb{F}_2^n$ to be $L_{\mathcal{C}}(x) \stackrel{\text{def}}{=} |\mathcal{B}(x, pn) \cap \mathcal{C}|$.

For a vector $x \in \mathbb{F}_2^L$ and $I \subset [L]$, we use $x_I \in \mathbb{F}_2^{|I|}$ to denote the vector $(x_i)_{i \in I}$ with coordinates from I in increasing order. We use $\mathbf{1}_L$ to denote the all ones vector of length L. For two sets $A, B \subseteq \mathbb{F}_2^n$, define the sumset $A + B = \{a + b : a \in A, b \in B\}$. When $b \in \mathbb{F}_2^n$, let $A + b$ denote $A + \{b\}$.

We use several notions from probability and information theory. We let Bernoulli(p) be the distribution that returns 0 with probability $1 - p$ and 1 with probability p. For a random variable X with domain $\mathcal{X}$, we use $H(X)$ to denote the *entropy* of X:

$$H(X) = -\sum_{x \in \mathcal{X}} \mathop{\mathbf{Pr}}_{X}(x) \log(\mathop{\mathbf{Pr}}_{X}(x)).$$

For a probability distribution τ, we may also use $H(\tau)$ to denote the entropy of a random variable with distribution τ. We use $\mathrm{supp}(\tau)$ to denote the set of values on which τ has positive probability mass.

Let X be a random variable supported on $\mathcal{X}$ and Y be a random variable supported on $\mathcal{Y}$. We define the *conditional entropy* of Y given X as

$$H(Y|X) = -\sum_{x \in \mathcal{X}, y \in \mathcal{Y}} p(x, y) \log \frac{p(x, y)}{p(x)}.$$

It is easy to check that $H(X) - H(X|Y) = H(Y) - H(Y|X)$ and we call this the *mutual information* $I(X;Y)$:

$$I(X;Y) = H(X) - H(X|Y) = H(Y) - H(Y|X)$$

For random variables X, Y, Z, we define the *conditional mutual information* $I(X;Y|Z)$ by

$$I(X;Y|Z) = H(X|Z) - H(X|Y,Z) = H(Y|Z) - H(Y|X,Z)$$

Conditional entropy, mutual information, and conditional mutual information satisfy the *data processing inequality:* for any function f supported on the domain of Y, we have

$$H(X|f(Y)) \geq H(X|Y) \quad \text{and} \quad I(X;Y) \geq I(X;f(Y)) \quad \text{and} \quad I(X;Y|Z) \geq I(X;f(Y)|Z).$$

We also use *Fano's inequality*, which states that if X is a random variable supported on $\mathcal{X}$ and Y is a random variable supported on $\mathcal{Y}$, and if $f : \mathcal{Y} \to \mathcal{X}$ is a function and $p_{err} = \mathbf{Pr}_{X,Y}[f(Y) \neq X]$

$$H(X|Y) \leq h(p_{err}) + p_{err} \cdot \log(|\mathcal{X}| - 1)$$

We also use the following binomial estimate. For a distribution τ whose support elements have probability masses $p_1, \ldots, p_\ell$ summing to 1, we have

$$2^{(H(\tau) - o(1))n} \leq \binom{n}{p_1 n, \ldots, p_\ell n} \leq 2^{H(\tau) n}. \tag{4.1}$$

4.4 Upper bound proof

In this section, we prove Theorem 4.4, which we restate here.

Theorem (Theorem 4.4, restated)**.** Let $\varepsilon > 0$ and $p \in (0, 1/2)$. A random linear code of rate $1 - h(p) - \varepsilon$ is $(p, h(p)/\varepsilon + 1)$-list decodable with probability $1 - \exp(-\Omega_\varepsilon(n))$.

Before it was known that a typical random linear code is $(p, O(1/\varepsilon))$-list decodable, Guruswami, Håstad, Sudan and Zuckerman [49] proved the *existence* of binary linear codes of rate $1-h(p)-\varepsilon$ that are $(p, 1/\varepsilon)$-list decodable. However, their argument did not work with high probability, and the authors explicitly stated this as a drawback of their proof. This section shows how to make the argument in [49] work with high probability. We start by reviewing the approach of [49], which is the basis of our proof.

4.4.1 The approach of [49]

The approach of [49] followed from a beautiful potential-function argument, which is the basis of our approach and which we describe here.

Let $k \stackrel{\text{def}}{=} Rn = (1 - h(p) - \varepsilon)n$. We choose vectors $b_1, \ldots, b_k$ one at a time, so that the code $\mathcal{C}_i \stackrel{\text{def}}{=} \text{span}(b_1, \ldots, b_i)$ remains "nice": formally, so that a potential function $\tilde{S}_{\mathcal{C}_i}$ remains small. Once we have picked all k vectors, we set $\mathcal{C} = \mathcal{C}_k$, and the fact that $\tilde{S}_{\mathcal{C}_k}$ is small implies list-decodability.

Recall that for a code $\mathcal{C}$ and $x \in \mathbb{F}_2^n$, we set $L_{\mathcal{C}}(x) = |\mathcal{B}(x, pn) \cap \mathcal{C}|$. Define

$$\tilde{S}_{\mathcal{C}} \stackrel{\text{def}}{=} \frac{1}{2^n} \sum_{x \in \mathbb{F}_2^n} 2^{\varepsilon n L_{\mathcal{C}}(x)}.$$

It is not hard to show that for any vectors $b_1, \ldots, b_i \in \mathbb{F}_2^n$,

$$\mathop{\mathbf{E}}_{b_{i+1} \sim \mathbb{F}_2^n} \left[\tilde{S}_{\mathcal{C}_i + \{0, b_{i+1}\}} | b_1, \ldots, b_i \right] \leq \tilde{S}^2_{\mathcal{C}_i}. \tag{4.2}$$

That is, when a uniformly random vector b_{i+1} is added to the basis $\{b_1, \ldots, b_i\}$, we expect the potential function not to grow too much. Hence, there exists a choice of vectors $b_1, \ldots, b_k$ so that $\tilde{S}_{\mathcal{C}_{i+1}} \leq \tilde{S}^2_{\mathcal{C}_i}$ for $i = 0, 1, \ldots, k-1$.[2]

As $\mathcal{C}_0 = \{0\}$, we have $\tilde{S}_{\mathcal{C}_0} \leq 1 + 2^{-n(1-h(p)-\varepsilon)}$. Setting $\mathcal{C} = \mathcal{C}_k = \text{span}(b_1, \ldots, b_k)$, we have

$$\tilde{S}_{\mathcal{C}} \leq \tilde{S}^{2^k}_{\mathcal{C}_0} \leq \left(1 + 2^{-n(1-h(p)-\varepsilon)}\right)^{2^k} \leq e^{2^{k - n(1-h(p)-\varepsilon)}} = e$$

by our choice of k. This implies that $\sum_x 2^{\varepsilon n L_{\mathcal{C}}(x)} \leq e \cdot 2^n$, and in particular, for all $x \in \mathbb{F}_2^n$, we have $2^{\varepsilon n L_{\mathcal{C}}(x)} \leq e \cdot 2^n$. Thus, for all x, $L_{\mathcal{C}}(x) \leq \frac{1}{\varepsilon} + o(1)$, as desired.

The approach of [49] is extremely clever, but these ideas have not, to the best of our knowledge, been used in subsequent work on the list-decodability of random linear codes. One reason is that the crux of the argument, which is (4.2), holds in expectation, and it was not clear how to show that it holds with high probability; thus, the result remained existential, and other techniques were introduced to study typical random codes [48, 23, 126, 108, 111].

[2] As a technical detail, one needs to be careful that $b_{i+1} \notin \mathcal{C}_i$. One can guarantee $b_{i+1} \notin \mathcal{C}_i$ by carefully examining the proof of (4.2), or use (4.2) to get a similar equation where we additionally condition $b_{i+1} \notin \mathcal{C}_i$.

4.4.2 Proof of Theorem 4.4

We improve the argument of [49] in two ways. First, we show that in fact, (4.2) essentially holds with high probability over the choice of b_{i+1}, which allows us to use the approach sketched above for random linear codes. Second, we introduce one additional trick which takes advantage of the linearity of the code in order to reduce the constant in the list size from 1 to $h(p)$. Before diving into the details, we briefly describe the main ideas.

The first improvement follows from looking at the potential function in the right way. In this paragraph, all $o(1)$ terms are exponentially small in n. Our goal is $\tilde{S}_{\mathcal{C}_k} \leq O(1)$. Write $\tilde{S}_{\mathcal{C}_i} = 1 + \tilde{T}_{\mathcal{C}_i}$. By above, $\tilde{T}_{\mathcal{C}_0} = \tilde{S}_{\mathcal{C}_0} - 1 = o(1)$. We show that with high probability, for all $i \leq k$, we have $\tilde{T}_{\mathcal{C}_i} = o(1)$. In the [49] argument we have

$$\mathbf{E}\,\tilde{S}_{\mathcal{C}_{i+1}} \leq \tilde{S}^2_{\mathcal{C}_i} = (1 + \tilde{T}_{\mathcal{C}_i})^2 = 1 + 2\tilde{T}_{\mathcal{C}_i}(1 + o(1)),$$

and so $\mathbf{E}\,\tilde{T}_{\mathcal{C}_{i+1}} = 2\tilde{T}_{\mathcal{C}_i}(1 + o(1))$. Crucially, one can show that, always, $2\tilde{T}_{\mathcal{C}_i} \leq \tilde{T}_{\mathcal{C}_{i+1}}$. Thus, by Markov's inequality, $\tilde{T}_{\mathcal{C}_{i+1}} = 2\tilde{T}_{\mathcal{C}_i}(1 + o(1))$ with probability $1 - o(1)$, for appropriately chosen $o(1)$ terms. Union bounding over the $o(1)$ failure probabilities in the k steps, we conclude that $\tilde{T}_{\mathcal{C}_i}$ grows roughly as slowly as in the existential argument, giving the desired list decodability.

The second improvement follows from the linearity of the code. In the last step of the [49] argument, we replace the summation "$\sum_x$" in $\sum_x 2^{\varepsilon n L_{\mathcal{C}}(x)} \leq e \cdot 2^n$ with a "$\forall x$." We can save a bit because, by linearity, the contribution $2^{\varepsilon n L_{\mathcal{C}}(x)}$ is the same for all x in a coset $y + \mathcal{C}$.

Now we go through the details. With hindsight, let $\eta = \lfloor \frac{h(p)}{\varepsilon} + 2 \rfloor - (\frac{h(p)}{\varepsilon} + 1) > 0$. It is convenient to change the definition of the potential function very slightly: losing the tilde, define, for a code $\mathcal{C} \subset \mathbb{F}_2^n$,

$$A_{\mathcal{C}}(x) \stackrel{\text{def}}{=} 2^{\frac{\varepsilon n L_{\mathcal{C}}(x)}{1+\varepsilon\eta}} \qquad \text{and} \qquad S_{\mathcal{C}} \stackrel{\text{def}}{=} \mathop{\mathbf{E}}_{x \sim \mathbb{F}_2^n}[A_{\mathcal{C}}(x)] \qquad \text{and} \qquad T_{\mathcal{C}} \stackrel{\text{def}}{=} S_{\mathcal{C}} - 1.$$

As noted above, it is helpful to think of $T_{\mathcal{C}}$ as a very small term; we would like to show—in accordance with (4.2)—that $T_{\mathcal{C}}$ approximately doubles each time we add a basis vector. The term $S_{\mathcal{C}}$ differs from the term $\tilde{S}_{\mathcal{C}}$ above in that $A_{\mathcal{C}}(x)$ has an extra factor of $\frac{1}{1+\varepsilon\eta}$ in the exponent. This is an extra "slack" term that helps guarantee a high probability result under the same parameters. However, this definition does not change how the potential function behaves. In particular, we still have the following lemma:

Lemma 4.6 (Following [49])**.** *For all linear $\mathcal{C} \subseteq \mathbb{F}_2^n$ and all $b \in \mathbb{F}_2^n$,*

$$L_{\mathcal{C}+\{0,b\}}(x) \leq L_{\mathcal{C}}(x) + L_{\mathcal{C}}(x+b) \tag{4.3}$$

$$A_{\mathcal{C}+\{0,b\}}(x) \leq A_{\mathcal{C}}(x) \cdot A_{\mathcal{C}}(x+b), \tag{4.4}$$

with equality if and only if $b \notin \mathcal{C}$.

Proof. To see (4.3), notice that

$$\begin{aligned} L_{\mathcal{C}+\{0,b\}}(x) &= |\mathcal{B}(x, pn) \cap (\mathcal{C} \cup (\mathcal{C} + b))| \\ &\leq |\mathcal{B}(x, pn) \cap \mathcal{C}| + |\mathcal{B}(x, pn) \cap (\mathcal{C} + b)| \\ &= |\mathcal{B}(x, pn) \cap \mathcal{C}| + |\mathcal{B}(x + b, pn) \cap \mathcal{C}| \\ &= L_{\mathcal{C}}(x) + L_{\mathcal{C}}(x + b), \end{aligned}$$

with equality in the second line if and only if $b \notin \mathcal{C}$. Inequality (4.4) follows as a consequence of (4.3), and this proves the lemma. □

The following lemma is the key step of our proof, establishing that when $T_{\mathcal{C}}$ is small, it essentially doubles with high probability each time we add a basis vector.

Lemma 4.7. *If $\mathcal{C}$ is a fixed linear code,*

$$\Pr_{b \sim \mathbb{F}_2^n}\left[S_{\mathcal{C}+\{0,b\}} \geq 1 + 2T_{\mathcal{C}} + T_{\mathcal{C}}^{1.5}\right] \leq T_{\mathcal{C}}^{0.5}.$$

Proof. By Lemma 4.6, for all b,

$$\begin{aligned} S_{\mathcal{C}+\{0,b\}} &= \mathop{\mathbf{E}}_{x}\left[A_{\mathcal{C}+\{0,b\}}(x)\right] \\ &\leq \mathop{\mathbf{E}}_{x}\left[A_{\mathcal{C}}(x)A_{\mathcal{C}}(x+b)\right] \\ &= \mathop{\mathbf{E}}_{x}\left[-1 + A_{\mathcal{C}}(x) + A_{\mathcal{C}}(x+b) + (A_{\mathcal{C}}(x)-1)(A_{\mathcal{C}}(x+b)-1)\right] \\ &= 1 + 2T_{\mathcal{C}} + \mathop{\mathbf{E}}_{x}\left[(A_{\mathcal{C}}(x)-1)(A_{\mathcal{C}}(x+b)-1)\right]. \end{aligned}$$

Over the randomness of b and x, we have x and $x+b$ are independently uniform over $\mathbb{F}_2^n$, so

$$\mathop{\mathbf{E}}_{b}\mathop{\mathbf{E}}_{x}\left[(A_{\mathcal{C}}(x)-1)(A_{\mathcal{C}}(x+b)-1)\right] = \mathop{\mathbf{E}}_{b,x}\left[A_{\mathcal{C}}(x)-1\right] \cdot \mathop{\mathbf{E}}_{b,x}\left[A_{\mathcal{C}}(x+b)-1\right] = T_{\mathcal{C}}^2. \tag{4.5}$$

As $A_{\mathcal{C}}(x) - 1$ is always nonnegative, we have, by Markov's inequality,

$$\Pr_{b}\left[S_{\mathcal{C}+\{0,b\}} \geq 1 + 2T_{\mathcal{C}} + T_{\mathcal{C}}^{1.5}\right] \leq \Pr_{b}\left[\mathop{\mathbf{E}}_{x}\left[(A_{\mathcal{C}}(x)-1)(A_{\mathcal{C}}(x+b)-1)\right] \geq T_{\mathcal{C}}^{1.5}\right] \leq \frac{T_{\mathcal{C}}^2}{T_{\mathcal{C}}^{1.5}} = T_{\mathcal{C}}^{0.5}. \qquad \square$$

Iterating Lemma 4.7 gives the following.

Lemma 4.8. *Let $p \in (0, 1/2)$ and $\varepsilon \in (0, 1-h(p))$. Let $\mathcal{C} \subset \mathbb{F}_2^n$ be a random linear code of rate $1 - h(p) - \varepsilon$. Then, with probability $1 - \exp(-\Omega_{\varepsilon,p}(n))$, we have $S_{\mathcal{C}} < 2$.*

Proof. It suffices to prove this when n is sufficiently large in terms of ε. As in §4.4.1, let $b_1, b_2, \ldots, b_k \in \mathbb{F}_2^n$ be independently and uniformly chosen, and let $\mathcal{C}_i = \operatorname{span}\{b_1, \ldots, b_i\}$. Consider the sequence

$$\begin{aligned} \delta_0 &\stackrel{\text{def}}{=} 2^{-n(1-h(p)-\frac{\varepsilon}{1+\varepsilon\eta})} \\ \delta_i &\stackrel{\text{def}}{=} 2\delta_{i-1} + \delta_{i-1}^{1.5}. \end{aligned}$$

We can verify by induction that for $i \leq n(1-h(p)-\varepsilon)$, we have $\delta_i < 2^{i+1}\delta_0 < 2^{-\frac{\varepsilon^2\eta n}{2}}$. To see this, notice that the base case is trivial, and if $\delta_j < 2^{j+1}\delta_0$ for $j < i$, we have

$$\delta_i = 2\delta_{i-1}(1+\delta_{i-1}^{0.5}) = 2^i\delta_0 \cdot \prod_{j=0}^{i-1}(1+\delta_j^{0.5}) \leq 2^i\delta_0 \cdot e^{\sum_{j=0}^{i-1}\delta_j^{0.5}} < 2^{i+1}\delta_0.$$

In the first two equalities, we applied the definitions of δ_i and $\delta_{i-1}, \ldots, \delta_1$, respectively. In the first inequality, we used the estimate $1 + z \leq e^z$, and in the second we used the inductive hypothesis $\delta_j < 2^{-\frac{\varepsilon^2\eta n}{2}}$ for $j < i$

and that n is sufficiently large. By this induction, we conclude that, if $k = n(1-h(p)-\varepsilon)$, then $\delta_k < 2^{-\frac{\varepsilon^2\eta n}{2}}$.

Let $b_1, \ldots, b_k \in \mathbb{F}_2^n$ be random vectors, and let $\mathcal{C}_i = \operatorname{span}(b_1, \ldots, b_i)$ with $\mathcal{C}_k = \mathcal{C}$. Call $\mathcal{C}_i$ *good* if $T_{\mathcal{C}_i} < \delta_i$. We have

$$T_{\mathcal{C}_0} \;=\; S_{\{0\}} - 1 \;=\; \left(2^{\frac{\varepsilon n}{1+\varepsilon\eta}} - 1\right) \cdot \frac{\operatorname{Vol}(n, pn)}{2^n} \;<\; 2^{\frac{\varepsilon n}{1+\varepsilon\eta}} \cdot \frac{2^{h(p)n}}{2^n} \;=\; \delta_0,$$

so $\mathcal{C}_0$ is always good. On the other hand, by the definition of δ_i and Lemma 4.7, if $\mathcal{C}_i$ is good,

$$\mathbf{Pr}\left[\mathcal{C}_{i+1} \text{ not good}\right] \;=\; \mathbf{Pr}\left[T_{\mathcal{C}_{i+1}} \geq \delta_{i+1}\right] \;\leq\; \mathbf{Pr}\left[T_{\mathcal{C}_{i+1}} \geq 2T_{\mathcal{C}_i} + T_{\mathcal{C}_i}^{1.5}\right] \;\leq\; T_{\mathcal{C}_i}^{0.5} \;<\; \delta_i^{0.5}.$$

Thus, with probability at least

$$1 - \left(\delta_0^{0.5} + \delta_1^{0.5} + \cdots + \delta_{k-1}^{0.5}\right) \;>\; 1 - k2^{-\varepsilon^2\eta n/4} \;\geq\; 1 - 2^{-\Omega_{\varepsilon,p}(n)}$$

we have $T_{\mathcal{C}_i} < \delta_i$ for all $i = 0, \ldots, k$. In particular, $T_{\mathcal{C}} = T_{\mathcal{C}_k} < \delta_k < 2^{-\frac{\varepsilon^2\eta n}{2}}$. Thus, $S_{\mathcal{C}} = 1 + T_{\mathcal{C}} < 2$ with probability $1 - \exp(-\Omega_{\varepsilon,p}(n))$, completing the proof of Lemma 4.8. □

Finally, we prove the following lemma, which implies Theorem 4.4.

Lemma 4.9. *Any linear code $\mathcal{C} \subseteq \mathbb{F}_2^n$ of rate $1 - h(p) - \varepsilon$ with $S_{\mathcal{C}} < 2$ is $(p, \frac{h(p)}{\varepsilon} + 1)$-list decodable.*

Proof. Suppose for sake of contradiction that there exists $x^* \in \mathbb{F}_2^n$ such that $|\mathcal{B}(x^*, pn) \cap \mathcal{C}| \geq \lfloor \frac{h(p)}{\varepsilon} + 2 \rfloor$. By linearity of $\mathcal{C}$, for all $x \in \mathbb{F}_2^n$ and $c \in \mathcal{C}$, we have

$$|\mathcal{B}(x + c, pn) \cap \mathcal{C}| = |\mathcal{B}(x, pn) \cap (\mathcal{C} - c)| = |\mathcal{B}(x, pn) \cap \mathcal{C}|,$$

so $|\mathcal{B}(x^* + c, pn) \cap \mathcal{C}| \geq \lfloor \frac{h(p)}{\varepsilon} + 2 \rfloor = \frac{h(p)}{\varepsilon} + 1 + \eta$ for all $c \in \mathcal{C}$. If $S_{\mathcal{C}} < 2$, then we have

$$\begin{aligned}
2^{n+1} \;>\; 2^n S_{\mathcal{C}} \;&=\; \sum_{x \in \mathbb{F}_2^n} \exp\left(n \cdot \frac{\varepsilon}{1+\varepsilon\eta} \cdot |\mathcal{B}(x, pn) \cap \mathcal{C}|\right) \\
&\geq\; \sum_{c \in \mathcal{C}} \exp\left(n \cdot \frac{\varepsilon}{1+\varepsilon\eta} \cdot |\mathcal{B}(x^* + c, pn) \cap \mathcal{C}|\right) \\
&\geq\; \exp\left(n(1 - h(p) - \varepsilon)\right) \cdot \exp\left(n \cdot \frac{\varepsilon}{1+\varepsilon\eta} \cdot \left(\frac{h(p)}{\varepsilon} + 1 + \eta\right)\right) \\
&=\; \exp\left(n\left(1 + \frac{\varepsilon\eta(1 - h(p) - \varepsilon)}{1+\varepsilon\eta}\right)\right).
\end{aligned}$$

which is a contradiction for large enough n. □

Proof of Theorem 4.4. Apply Lemma 4.8 and then Lemma 4.9 for $R = 1 - h(p) - \varepsilon$. □

Remark 4.10. We do not see how to extend this proof to larger alphabets. If, for example, $q = 3$, Lemma 4.7 would need to say $\mathbf{Pr}[S_{\mathcal{C}+\{0,b,2b\}} > 1 + 3T_{\mathcal{C}} + o(T_{\mathcal{C}})] < o(1)$. However, the current proof would fail to show

this, as we could not separate the expectation in (4.5); that is, we cannot say

$$\begin{aligned}&\mathop{\mathbf{E}}_{b,x}\left[(A_{\mathcal{C}}(x)-1)(A_{\mathcal{C}}(x+b)-1)(A_{\mathcal{C}}(x+2b)-1)\right]\\&\quad=\mathop{\mathbf{E}}_{b,x}\left[A_{\mathcal{C}}(x)-1\right]\cdot\mathop{\mathbf{E}}_{b,x}\left[A_{\mathcal{C}}(x+b)-1\right]\cdot\mathop{\mathbf{E}}_{b,x}\left[A_{\mathcal{C}}(x+2b)-1\right]\end{aligned}$$

4.5 Lower bound proof

4.5.1 Techniques.

To illustrate the techniques for our lower bounds, we warm with some calculations that suggest why the "right" list-size is $\frac{h(p)}{\varepsilon}$.

Consider a random linear code $\mathcal{C}\subset\mathbb{F}_q^n$, of rate $R=1-h(p)-\varepsilon$, where $p\in(0,\frac{1}{2})$. Let $L=\lfloor h(p)/\varepsilon-0.01\rfloor$. We claim that $\mathcal{C}$ is unlikely to be (p,L)-list-decodable.

Define a set of matrices $\mathcal{M}$ as follows: Let $u\in\mathbb{F}_q^L$ be a random vector with independent Bernoulli(p) entries, namely, each entry is 0 with probability $1-p$ and 1 with probability p. Let x be uniformly sampled from $\mathbb{F}_2$. Let τ denote the distribution (over $\mathbb{F}_2^L$) of the random vector $u+x\cdot 1_L$. Finally, define $\mathcal{M}$ to be the set of all matrices $M\in\mathbb{F}_q^{n\times L}$, such that a uniformly sampled row of M has the distribution τ. So, for example, every $M\in\mathcal{M}$ has $\frac{1}{2}((1-p)^L+p^L)n$ rows with all ones.

We will show that $\mathcal{M}$ is *bad* and *abundant*. By *bad* we mean that a linear code containing a matrix from $\mathcal{M}$ cannot be (p,L)-list-decodable. Given a matrix $M\in\mathbb{F}_q^{n\times L}$, we write $M\subseteq\mathcal{C}$ ("$\mathcal{C}$ contains M") to mean that each of the columns of M is a codeword in $\mathcal{C}$. We say that $\mathcal{M}$ is *abundant* (for the rate R) if a random linear code of rate R is likely to contain at least one matrix from $\mathcal{M}$. Clearly, the combination of these properties means that $\mathcal{C}$ is unlikely to be (p,L)-list-decodable.

We first prove that $\mathcal{M}$ is bad. Assume that $\mathcal{C}$ contains some matrix $M\in\mathcal{M}$. For $j=1,\ldots,n$ we may write the jth row of M as $u_j+x_j\cdot 1_L$, so that, for a random $j\in[n]$, the pair (u_j,x_j) is distributed as Bernoulli$(p)^L\times$ Bernoulli$(1/2)$. Then every column of M has Hamming distance exactly pn from the vector $x=(x_1,x_2,\ldots,x_n)$, so then $\mathcal{B}(x,pn)$ contains all the column vectors of M. Thus, $\mathcal{C}$ cannot be (p,L)-list-decodable, as the set of L column vectors of M is a "bad list" for list-decodability with these parameters.

Showing that $\mathcal{M}$ is abundant is harder, and at this stage we only provide some intuition for this fact. Let us compute the expected number of matrices $M\in\mathcal{M}$ that are contained in $\mathcal{C}$. First, we estimate the cardinality of $\mathcal{M}$. Let M be a matrix in $\mathcal{M}$. For each $u\in\mathbb{F}_2^L$, let p_u denote the probability that a sample from τ is u, so that $\sum_u p_u=1$. For example $p_{1_L}=\frac{1}{2}((1-p)^L+p^L)$. Then $\mathcal{M}$ is the set of all matrices with with exactly $p_u n$ rows equal to u for all u. Thus, we have by (4.1)

$$|\mathcal{M}|=\binom{n}{p_v n : v\in\mathbb{F}_2^L}\approx 2^{H(p_u:u\in\mathbb{F}_2^L)n}=2^{H(\tau)n}.$$

We can estimate $H(\tau)$, and thus $|\mathcal{M}|$, as follows: τ is the distribution $u+x\cdot 1_L$ for $u\sim$ Bernoulli$(p)^L$ and $x\sim$ Bernoulli$(1/2)$. Note that, if L is large, then for a random sample $v\sim\tau$, we can with high probability recover the pair (u,x) used to generate v: let x be the majority bit of v and let $u=v-x\cdot 1_L$. Thus, $H(\tau)$

is $H(u,x) = L \cdot h(p) + 1$. Therefore,

$$|\mathcal{M}| \approx 2^{n(Lh(p)+1)}. \tag{4.6}$$

and

$$\mathbb{E}\,|\{M \in \mathcal{M}\,\mathrm{s.t.}\,M \subseteq \mathcal{C}\}| = 2^{RLn} \cdot |\mathcal{M}|2^{-Ln} \approx 2^{n-\varepsilon Ln} \geq 2^{n(1-h(p))} \tag{4.7}$$

where, the approximation comes from (4.6) and substituting $1 - h(p) - \varepsilon$ for R. Thus, in expectation, $\mathcal{C}$ contains many "bad" lists for list-decoding.

Of course, this back-of-the-envelope calculation does *not* yield the result advertised above. It might be the case that, even though the expected number of $M \in \mathcal{M}$ so that $M \subseteq \mathcal{C}$ is large, the probability that such an M exists is still small. In fact, as [101] shows, there are simple examples where this does happen. The above example also illustrates this: even if we chose $L \sim 1/\varepsilon$, rather than $L \sim h(p)/\varepsilon$, the expected number of bad lists would still be exponentially large in (4.7). But of course if $L = 1/\varepsilon$, we cannot have many bad lists with high probability, because earlier in this chapter (Theorem 4.4) we showed that random linear codes *are* (p, L) list decodable with $L = h(p)/\varepsilon + 1$. Thus, proving that $\mathcal{M}$ is abundant requires more work.

A standard approach to show that $\mathcal{M}$ is abundant would be via the second-moment method. Recently, [101] gave a general theorem which encompasses second-moment calculations in this context. In particular, they showed that there is essentially only one reason that a set $\mathcal{M}$ might not be abundant: there exists some matrix $A \in \mathbb{F}_2^{L \times L'}$, such that the set $\{MA \mid M \in \mathcal{M}\}$ is small. If this occurs, we say that $\mathcal{M}$ is *implicitly rare*.[3] They used this result to study the list-decodability of random Low-Density Parity-Check codes, but we can use their result to do our second moment calculation. We show that our example of $\mathcal{M}$ above is not implicitly rare, by showing that there is no such linear map A. Appealing to the machinery of [101], rather than applying the second moment method from scratch, allows us to get tighter constants with slightly less work, and gives a more principled approach to our lower bound.

With the above outline, we can sketch why $L = \lfloor h(p)/\varepsilon - 0.01 \rfloor$ is the "correct" list size from our lower bound. By above, we can show a list-size lower bound of L if the expected occurrences of matrices of $\mathcal{M}_A \stackrel{\text{def}}{=} \{MA | M \in \mathcal{M}\}$ is large for any $A \in \mathbb{F}_2^{L \times L'}$. For our chosen $\mathcal{M}$, the matrix A with the smallest expected occurrences of $\mathcal{M}_A$ turns out to be the matrix

$$A = \left[\begin{array}{c} I \\ \hline 1 \quad \cdots \quad 1 \end{array}\right] \in \mathbb{F}_2^{L \times (L-1)}.$$

In this case, the rows of MA are distributed as $\tau' \sim (u_1 + u_L, u_2 + u_L, \ldots, u_{L-1} + u_L)$ where $u_1, \ldots, u_L \sim$ Bernoulli(p). For L large, we can recover $u_1, \ldots, u_L$ from a sample $v \sim \tau'$ with high probability by setting u_L as the majority bit and $u_i = v_i - u_L$ for $i = 1, \ldots, L-1$. Thus, we have roughly $H(\tau') \sim H(u_1, \ldots, u_L) =$

[3] The term "implicitly rare" is used by the first version of [101], available at https://arxiv.org/abs/1909.06430v1.

$L \cdot h(p)$. This implies that

$$\mathbb{E}\,|\{M \in \mathcal{M} \,\mathrm{s.t.}\, AM \subseteq \mathcal{C}\}| = 2^{R(L-1)n} \cdot |\mathcal{M}| 2^{-(L-1)n} \approx 2^{h(p)n - \varepsilon L n}.$$

Thus, the expected occurrences of $\mathcal{M}_A$ is larger than 1 only for $L < \frac{h(p)}{\varepsilon}$, hence the choice of L.

4.5.2 Tools from [101]

As discussed in Section 4.5.1, for our lower bounds we use tools from the recent work [101].

For a distribution τ on $\mathbb{F}_2^L$ and a matrix $A \in \mathbb{F}_2^{L' \times L}$, we define the distribution $A\tau$ on $\mathbb{F}_2^{L'}$ in the natural way by

$$\mathop{\mathbf{Pr}}_{A\tau}(x) = \sum_{\{y \in \mathbb{F}_2^L \,\mathrm{s.t.}\, Ay = x\}} \mathop{\mathbf{Pr}}_{\tau}(y),$$

namely, $A\tau$ is the distribution of the random vector Ay, where $y \sim \tau$.

We work with matrices $M \in \mathbb{F}_2^{n \times L}$ ($L \in \mathbb{N}$), where we view the columns of M as potential codewords in $\mathcal{C}$. We use the notation "$M \subseteq \mathcal{C}$" to mean that the columns of M are all contained in $\mathcal{C}$.

We group together sets of such matrices M according to their row distribution.

Definition 4.11 (τ_M, $\dim(\tau)$, $\mathcal{M}_{n,\tau}$)**.** Given a matrix $M \in \mathbb{F}_2^{n \times L}$, the empirical row distribution defined by the rows of M over $\mathbb{F}_2^L$ is called the type τ_M of M. That is, τ_M is the distribution so that for $v \in \mathbb{F}_2^L$,

$$\mathop{\mathbf{Pr}}_{\tau_M}(v) = \frac{|\{i \,\mathrm{s.t.}\, \text{the } i\text{'th row of } M \text{ is equal to } v\}|}{n}.$$

For a distribution τ on $\mathbb{F}_2^L$, we use $\dim(\tau)$ to refer to $\dim(\mathrm{span}(\mathrm{supp}(\tau)))$. We use $\mathcal{M}_{n,\tau}$ to refer to the set of all matrices in $\mathbb{F}_2^{n \times L}$ which have empirical row distribution τ.

Remark 4.12. We remark that for some distributions τ over $\mathbb{F}_2^L$, the set $\mathcal{M}_{n,\tau}$ may be empty due to $n \cdot \mathbf{Pr}_\tau(v)$ not being an integer. For such τ we can define $\mathcal{M}_{n,\tau}$ to consist of matrices M with either $\lfloor n \cdot \mathbf{Pr}_\tau(v) \rfloor$ or $\lceil n \cdot \mathbf{Pr}_\tau(v) \rceil$ copies of v. This has a negligible effect on the analysis as we always take n to be sufficiently large compared to other parameters, so for clarity of exposition we ignore this technicality.

Given $M \in \mathcal{M}_{n,\tau}$, note that $\mathcal{M}_{n,\tau}$ consists exactly of those matrices obtained by permuting the rows of M. In particular, since the *random linear code* model is invariant to such permutations, all of the matrices in $\mathcal{M}_{n,\tau}$ have the same probability of being contained in $\mathcal{C}$.

As discussed in Section 4.5.1, we prove a lower bound by exhibiting a distribution τ such that the corresponding set $\mathcal{M}_{n,\tau}$ is both *bad* and *abundant*. When $\mathcal{M}_{n,\tau}$ satisfies these properties, we say that τ itself is, respectively, *bad* and *abundant*.

The work [101] characterizes which distributions τ satisfy the *abundance* property, namely, which classes $\mathcal{M}_{n,\tau}$ are likely to have at least one of their elements appear (as a matrix) in a random linear code of a given rate. To motivate the definition below, suppose that the distribution τ has low entropy: $H(\tau) < \gamma \cdot \dim(\tau)$ for some $\gamma \in (0,1)$. This implies that the class $\mathcal{M}_{n,\tau}$ is not too big: more precisely, it is not hard to see that $|\mathcal{M}_{n,\tau}| \le 2^{H(\tau) \cdot n} \le 2^{\gamma \dim(\tau) n}$. Using a calculation like we did in Section 4.5.1, we see that, since $\mathcal{M}_{n,\tau}$ is not very large, it is unlikely for a random linear code of rate less than $1 - \gamma$ to contain a matrix from $\mathcal{M}_{n,\tau}$.

However, this is not the only reason that $\mathcal{M}_{n,\tau}$ might be unlikely to appear in a random linear code.

As is shown in [101], it could also be because a random output of τ, subject to some linear transformation (perhaps to a space of smaller dimension), has low entropy. We call such distributions *implicitly rare:*

Definition 4.13 (γ-implicitly rare)**.** We say that a distribution τover $\mathbb{F}_2^L$ is γ-*implicitly rare* if there exists a full-rank linear transformation $A : \mathbb{F}_2^L \to \mathbb{F}_2^{L'}$ where $L' \leq L$ such that

$$H(A\tau) < \gamma \cdot \dim(A\tau)$$

Observe that by taking A to be the identity map, we recover the case where τ itself has low entropy. Furthermore, note that every matrix in $\mathcal{M}_{n,A\tau}$ has all of its columns contained in the column-span of some matrix in $\mathcal{M}_{n,\tau}$. Hence, for a linear code to contain the latter matrix, it must also contain the former. Thus, abundance of the distribution τ implies abundance of $A\tau$. Definition 4.13 is motivated by the contrapositive of this statement, namely, non-abundance of $A\tau$ implies non-abundance of τ.

For an illustrative example of an implicitly rare distribution, we refer the reader to [101, Example 2.5]. Specifically, the example provides a case where for some full-rank matrix A, we have $H(A\tau)/\dim(A\tau) > H(\tau)/\dim(\tau)$.

Essentially, [101] shows that a row distribution τ is likely to appear in a random linear code (namely, τ satisfies the abundance property) if and only if it is *not* implicitly rare. The following theorem follows from Lemma 2.7 in [101].[4]

Theorem 4.14 (Follows from Lemma 2.7 in [101])**.** *Let $R \in (0,1)$ and fix $\eta > 0$. Let τ be a $(1-R-\eta)$-implicitly rare distribution over $\mathbb{F}_2^L$ ($L \in \mathbb{N}$), and let $\mathcal{C}$ be a random linear code of rate R. Then*

$$\mathbf{Pr}[\exists M \in \mathcal{M}_{n,\tau} : M \subseteq \mathcal{C}] \leq 2^{-\eta n}$$

Conversely, suppose that τ is not $(1-R+\eta)$-implicitly rare. Then

$$\mathbf{Pr}[\exists M \in \mathcal{M}_{n,\tau} : M \subseteq \mathcal{C}] \geq 1 - n^{O_L(1)} \cdot 2^{-\eta n} \ .$$

The first part of the theorem follows from a natural first-moment method argument, while the second part follows from the analogous second-moment argument. The $O_L(1)$ term in the second part is $2^{O(L)}$. We emphasize that it is important that we allow arbitrary full-rank linear transformations $A : \mathbb{F}_2^L \to \mathbb{F}_2^{L'}$ in Definition 4.13: if we only allowed A to be the identity map, the second part of the theorem would be false.

4.5.3 List decoding RLC lower bound

In this section, we prove Theorem 4.5, which we restate here for convenience.

Theorem (Theorem 4.5, restated)**.** Fix $p \in (0, 1/2)$, and fix $\delta \in (0,1)$. There exists $\varepsilon_{p,\delta} > 0$ such that for all $\varepsilon \in (0, \varepsilon_{p,\delta})$ and n sufficiently large, a random linear code in $\mathbb{F}_2^n$ of rate $1 - h(p) - \varepsilon$ is *not* $\left(p, \lfloor \frac{h(p)}{\varepsilon} - \delta \rfloor - 1\right)$-list-decodable with probability $1 - 2^{-\Omega(n)}$.

We first define a bad distribution τ in Definition 4.15; then we will show that it is bad in Proposition 4.16; then we will show that it is not implicitly rare (and hence abundant by Theorem 4.14) in Lemma 4.18. Finally we will prove Theorem 4.5 from these pieces.

[4] This is also given as Theorem 2.2 in the first version of [101], available at https://arxiv.org/abs/1909.06430v1.

Definition 4.15 (The bad distribution τ for list-decoding lower bounds)**.** Let $p \in (0, 1/2)$ and $\delta > 0$. Choose $L > 0$. Define the distribution τ on $\mathbb{F}_2^L$ as the distribution of the random vector $u + \alpha \mathbf{1}_L$, where $u \sim \text{Bernoulli}(p)^L$, and α is sampled independently and uniformly from $\mathbb{F}_2$.

First, we observe that τ is indeed bad, in the sense that it provides a counter-example to list-decodability.

Proposition 4.16 (τ is bad)**.** *Let τ be as in Definition 4.15. Let $\mathcal{C} \subseteq \mathbb{F}_2^n$ and let $M \in \mathcal{M}_{n,\tau}$. If $M \subseteq \mathcal{C}$, then $\mathcal{C}$ is not $(p, L-1)$-list-decodable.*

Proof. Let $M \in \mathcal{M}_{n,\tau}$. We want to show that the columns of M all lie in a single ball of radius pn.

By definition of τ and $\mathcal{M}_{n,\tau}$, we may write the j-th row of M as $u^{(j)} + \alpha_j \mathbf{1}_L$, so that the empirical distribution of the pairs $(u^{(j)}, \alpha_j)_{1 \le j \le n}$ is $\text{Bernoulli}(p)^L \times \text{Bernoulli}(1/2)$.[5]

For any $i \in [L]$, the number of $j \in [n]$ such that $M_{i,j} = u_i^{(j)} + \alpha_j \neq \alpha_j$ is exactly the number of times $u_i^{(j)} \neq 0$, which is pn, since $u_i^{(j)}$ is distributed as $\text{Bernoulli}(p)$. Thus, each column $M_{i,*}$ of M has distance at most pn from the word $(\alpha_1, \ldots, \alpha_n)$, so that any code containing M has L codewords in a ball of radius pn and hence is not $(p, L-1)$-list-decodable. □

Next, we show that τ is not implicitly rare for large enough L by showing that $A\tau$ has high entropy for any matrix A. We first show this is true when A is either the $L \times L$ identity I_L or an $L \times (L+1)$ matrix with the identity and an additional column with all nonzero entries.

Lemma 4.17. *Let $p \in (0, 1/2)$, $p' \in [p, 1/2]$, and $\delta > 0$. There exists $L_{p,\delta}$ such that, for all $L \ge L_{p,\delta}$ and $0 \le d \le L$, the following holds. Let w be a fixed vector in $\mathbb{F}_2^d$ all of whose entries are nonzero. Let v be a vector sampled from $\text{Bernoulli}(p)^d$ and let α be sampled from $\text{Bernoulli}(p')$. Then*

$$H(v + \alpha w) \ge d \cdot \left(h(p) + \frac{h(p)}{L + \delta} \right). \tag{4.8}$$

Proof. If $d = 0$, the assertion is trivial, so assume $d \ge 1$. As a guide to the reader, we emphasize that throughout the proof the vector v and the field element α are random variables, while the vector w is fixed.

We will bound $H(v + \alpha w)$ in two cases, one when d is small (relative to L) and one when d is large. (The precise definitions of "small" and "large" will be determined below.)

First we consider the case where d is small. We have (for any d) that

$$\begin{aligned} H(v + \alpha w) &= H(v_1 + \alpha w_1, v_2 + \alpha w_2, \ldots, v_d + \alpha w_d) \\ &= H(v_2 + \alpha w_2, \ldots, v_d + \alpha w_d | v_1 + \alpha w_1) + H(v_1 + \alpha w_1) \\ &\ge H(v_2 + \alpha w_2, \ldots, v_d + \alpha w_d | v_1, \alpha) + H(v_1 + \alpha w_1) \\ &= H(v_2, \ldots, v_d) + H(v_1 + \alpha w_1) \end{aligned} \tag{4.9}$$

The second equality uses the definition of conditional entropy. The inequality follows from the data processing inequality. The last equality uses the fact that w is a fixed vector so once α is known, $\alpha w_2, \ldots, \alpha w_d$ are also known, along with the assumption that the v_1 is independent of $v_2, \ldots, v_L$.

[5] This is without loss of generality: if not, as per Remark 4.12, we can associate pairs with rows so that the empirical distribution is close to $\text{Bernoulli}(p)^L \times \text{Uniform}(\mathbb{F}_2)$ up to an additive factors that are $o(1)$ as $n \to \infty$. After adjusting parameters, this has a negligible effect on the analysis and final result.

Now, $v_1 + \alpha w_1$ is nonzero if $v_1 = 0$ and $\alpha \neq 0$, if $v_1 \neq 0$ and $\alpha = 0$, or if $v_1, \alpha \neq 0$ and $v_1 + \alpha w_1 \neq 0$. This happens with probability $p^* = (1-p)p' + (1-p')p$. In the case that $v_1 + \alpha w_1$ is nonzero, each nonzero element of $\mathbb{F}_2$ has equal probability by symmetry. Thus $v_1 + \alpha w_1$ is distributed as Bernoulli(p^*). One can check that $H(\text{Bernoulli}(p^*)) = h(p^*)$, so from (4.9) we have

$$H(v + \alpha w) \geq H(v_2, \ldots, v_d) + h(p^*) = (d-1) \cdot h(p) + h(p^*). \tag{4.10}$$

Since $p < 2p(1-p) \leq p^* \leq 1/2$ and $h(\cdot)$ is strictly increasing on $(0, 1/2)$, we have $h(p^*) \geq h(p) + \varepsilon_p$ for some $\varepsilon_p > 0$. Thus, when $d \leq \varepsilon_p L$, (4.10) implies that

$$H(v + \alpha w) \geq d \cdot h(p) + \varepsilon_p \geq d \cdot \left(h(p) + \frac{1}{L} \right) > d \cdot \left(h(p) + \frac{h(p)}{L + \delta} \right),$$

where in the last inequality we have used that $\delta > 0$ and $h(p) < 1$. This lower bounds $H(v + \alpha w)$ in the case when d is "small," specifically when $d < \varepsilon_p \cdot L$.

Next we handle the case when d is "large." We have (for any d) that

$$\begin{aligned} H(v + \alpha w) &= H(v + \alpha w | \alpha) + H(\alpha) - H(\alpha | v + \alpha w) \\ &= d \cdot h(p) + h(p') - H(\alpha | v + \alpha w) \\ &\geq d \cdot h(p) + h(p) - H(\alpha | v + \alpha w). \end{aligned}$$

It thus suffices to show that $H(\alpha | v + \alpha w)$ is "small," which we do with Fano's inequality.

Let $\hat{\alpha}$ be the element of $\mathbb{F}_2$ that minimizes the Hamming distance $\Delta_H(\hat{\alpha} w, v + \alpha w)$, breaking ties arbitrarily. In expectation a $p < 1/2$ fraction of the d coordinates of v are nonzero. Similarly, for any vector $w' \in \mathbb{F}_2^d$ with all nonzero entries, in expectation a $1 - p > 1/2$ fraction of the coordinates of v disagree with w'.

By Hoeffding's inequality, for any nonzero $\zeta \in \mathbb{F}_2$,

$$\mathbf{Pr}\left[\Delta_H(v, \zeta w) \geq \frac{d}{2} \right] \leq 2\exp\left(-2d \left(\frac{1}{2} - p \right) \right) = \exp(-\Omega_p(d)) \tag{4.11}$$

and similarly

$$\mathbf{Pr}\left[\Delta_H(v, \mathbf{0}) \leq \frac{d}{2} \right] \leq 2\exp\left(-2d \left(\frac{1}{2} - p \right) \right) = \exp(-\Omega_p(d)). \tag{4.12}$$

If none of the events in (4.11) and (4.12) hold, then we have $\Delta_H(\alpha' w, v + \alpha w) < d/2$ for all $\alpha' \neq \alpha$ and $\Delta_H(\alpha w, v + \alpha w) > d/2$, in which case $\hat{\alpha} = \alpha$. Thus, by the union bound over the two events in (4.11) and (4.12), the probability that $\alpha \neq \hat{\alpha}$ is at most

$$p_{err} \stackrel{\text{def}}{=} \mathbf{Pr}[\hat{\alpha} \neq \alpha] \leq \exp(-\Omega_p(d)).$$

By Fano's inequality, as α takes at most 2 values and as $\hat{\alpha}$ is a function only of $v + \alpha w$, we have

$$H(\alpha | v + \alpha w) \leq h(p_{err}) \leq \exp(-\Omega_p(d)).$$

Thus, there exists some $d_{p,\delta}$ such that, for $d \geq d_{p,\delta}$, we have $H(\alpha|v+\alpha w) \leq \frac{\delta h(p)}{d+\delta}$, in which case

$$H(v+\alpha w) \geq d \cdot h(p) + h(p) - H(\alpha|v+\alpha w) \geq d \cdot h(p) + \frac{d}{d+\delta} \cdot h(p) \geq d \cdot \left(h(p) + \frac{h(p)}{L+\delta}\right).$$

This completes the case where d is "large."

We have shown that (4.8) holds when $d \leq \varepsilon_p \cdot L$ and when $d \geq d_{p,\delta}$. Thus, for $L \geq d_{p,\delta}/\varepsilon_p \stackrel{\text{def}}{=} L_{p,\delta}$, we have (4.8) always holds, as desired. □

Using Lemma 4.17, we may now prove that τ is not implicitly rare.

Lemma 4.18. *Let $p \in (0, 1/2)$ and let $\delta > 0$. There exists $L_{p,\delta}$ such that, for $L \geq L_{p,\delta}$, the distribution τ given in Definition 4.15 is* not *$(h(p) + \frac{h(p)}{L+\delta})$-implicitly rare.*

Proof. Let $L_{p,\delta}$ be as in Lemma 4.17. Let $L \geq L_{p,\delta}$, and let τ be the corresponding distribution in the lemma statement.[6] Fix a full-rank matrix A of rank L'. As τ is supported on $\mathbb{F}_2^L$, the rank of $A\tau$ is L'. We show that $H(A\tau) \geq L' \cdot (h(p) + \frac{h(p)}{L+\delta})$. At a high level, our strategy is to decompose the distribution $A\tau$ into several distributions that each have the set up of Lemma 4.17. Furthermore, this decomposition has enough conditional independence that the entropy of $A\tau$ can be lower bounded by the sum of the entropies of the smaller distributions, which we can lower bound by Lemma 4.17.

As A is full-rank it must have exactly L' rows. Since permuting the coordinates of τ yields the same distribution τ, permuting the columns of A does not change the entropy $H(A\tau)$; thus, we may assume that the first L' columns are linearly independent. Furthermore, if B is invertible, $H(BA\tau) = H(A\tau)$. Thus, by running Gaussian elimination on the rows of A, we may assume without loss of generality that

$$A = \left[\begin{array}{c|cccc} & | & | & \cdots & | \\ I_{L'} & w^{(1)} & w^{(2)} & \cdots & w^{(k)} \\ & | & | & \cdots & | \end{array}\right]$$

where $w^{(1)}, \ldots, w^{(k)} \in \mathbb{F}_2^{L'}$ and $k = L - L'$. Let a sample from τ be given by

$$\begin{bmatrix} v_1 \\ \vdots \\ v_{L'} \\ \alpha_1 \\ \vdots \\ \alpha_k \end{bmatrix} + \begin{bmatrix} 1 \\ \vdots \\ \\ \\ \\ 1 \end{bmatrix} \cdot \alpha_{k+1}.$$

where $v_1, \ldots, v_{L'}, \alpha_1, \ldots, \alpha_k \sim \text{Bernoulli}(p)$ and $\alpha_{k+1} \sim \text{Bernoulli}(1/2)$. (Note that this means that α_{k+1} is uniform on $\mathbb{F}_2$.) Then $A\tau$ is given by

$$\begin{bmatrix} v_1 \\ \vdots \\ v_{L'} \end{bmatrix} + \sum_{i=1}^{k+1} \begin{bmatrix} | \\ w^{(i)} \\ | \end{bmatrix} \cdot \alpha_i \tag{4.13}$$

[6] We treat the output of τ as a column vector.

where we let $w^{(k+1)}$ be the product $A \cdot \mathbf{1}_L \in \mathbb{F}_2^{L'}$. We emphasize that $v_1, \ldots, v_{L'}, \alpha_1, \ldots, \alpha_{k+1}$ are independent random variables, while A and $w^{(1)}, \ldots, w^{(k)}$ are fixed.

By definition of A and $w^{(k+1)}$, for any coordinate $i \notin \bigcup_{j=1}^{k} \mathrm{supp}(w^{(j)})$, we have $i \in \mathrm{supp}(w^{(k+1)})$. Thus, $\bigcup_{i=1}^{k+1} \mathrm{supp}(w^{(i)}) = [L']$. For $i = 1, \ldots, k+1$, let $J_i = \mathrm{supp}(w^{(i)}) \setminus (\bigcup_{j=1}^{i-1} J_j)$ (when $i = 1$ the union is the empty set), so that $J_1, \ldots, J_{k+1}$ form a partition of $[L']$. Recall that the notation $v_J \in \mathbb{F}_2^{|J|}$ denotes the vector $(v_i)_{i \in J}$ with coordinates from J in increasing order. We have

$$\begin{aligned} H(A\tau) &= H(A\tau | v_{J_{k+1}}, \alpha_{k+1}) + I(A\tau; v_{J_{k+1}}, \alpha_{k+1}) \\ &= H(A\tau | v_{J_k}, v_{J_{k+1}}, \alpha_k, \alpha_{k+1}) + I(A\tau; v_{J_k}, \alpha_k | v_{J_{k+1}}, \alpha_{k+1}) + I(A\tau; v_{J_{k+1}}, \alpha_{k+1}) \end{aligned}$$

Continuing, we have

$$\begin{aligned} H(A\tau) &= H(A\tau | v_{J_1}, \ldots, v_{J_{k+1}}, \alpha_1, \ldots, \alpha_{k+1}) + \sum_{i=1}^{k+1} I(A\tau; v_{J_i}, \alpha_i | v_{J_{i+1}}, \ldots, v_{J_{k+1}}, \alpha_{i+1}, \ldots, \alpha_{k+1}) \\ &= \sum_{i=1}^{k+1} I(A\tau; v_{J_i}, \alpha_i | v_{J_{i+1}}, \ldots, v_{J_{k+1}}, \alpha_{i+1}, \ldots, \alpha_{k+1}), \end{aligned} \tag{4.14}$$

where the second equality uses that $J_1, \ldots, J_{k+1}$ form a partition of $[L']$, so $A\tau$ is completely determined by $v_{J_1}, \ldots, v_{J_{k+1}}, \alpha_1, \ldots, \alpha_{k+1}$, and thus $H(A\tau | v_{J_1}, \ldots, v_{J_{k+1}}, \alpha_1, \ldots, \alpha_{k+1}) = 0$. For clarity, we note that the summand above when $i = k+1$ is simply $I(A\tau; v_{J_{k+1}}, \alpha_{k+1})$. We thus have

$$\begin{aligned} H(A\tau) &\geq \sum_{i=1}^{k+1} I((A\tau)_{J_i}; v_{J_i}, \alpha_i | v_{J_{i+1}}, \ldots, v_{J_{k+1}}, \alpha_{i+1}, \ldots, \alpha_{k+1}) \\ &= \sum_{i=1}^{k+1} I\left(v_{J_i} + \sum_{j \geq i} w_{J_i}^{(j)} \cdot \alpha_j ; v_{J_i}, \alpha_i \middle| v_{J_{i+1}}, \ldots, v_{J_{k+1}}, \alpha_{i+1}, \ldots, \alpha_{k+1} \right) \\ &= \sum_{i=1}^{k+1} I\left(v_{J_i} + w_{J_i}^{(i)} \cdot \alpha_i ; v_{J_i}, \alpha_i \middle| v_{J_{i+1}}, \ldots, v_{J_{k+1}}, \alpha_{i+1}, \ldots, \alpha_{k+1} \right) \\ &= \sum_{i=1}^{k+1} I\left(v_{J_i} + w_{J_i}^{(i)} \cdot \alpha_i ; v_{J_i}, \alpha_i \right) \\ &= \sum_{i=1}^{k+1} H\left(v_{J_i} + w_{J_i}^{(i)} \cdot \alpha_i \right) \end{aligned}$$

The inequality applies the data processing inequality to (4.14), using that $(A\tau)_{J_i}$ is a function of $A\tau$. The first equality uses (4.13) and that $w^{(1)}, \ldots, w^{(i-1)}$ have no support in J_i by definition of J_i. The second equality uses that $\alpha_{i+1}, \ldots, \alpha_{k+1}$ are being conditioned on. The third equality uses that the v_i's and α_i's are all independent and that the J_i are pairwise disjoint, so changing $v_{J_{i+1}}, \ldots, v_{J_{k+1}}, \alpha_{i+1}, \ldots, \alpha_{k+1}$ does not affect $v_{J_i} + w_{J_i}^{(i)} \cdot \alpha_i$. The last equality uses that $H(v_{J_i} + w_{J_i}^{(i)} \cdot \alpha_i | v_{J_i}, \alpha_i) = 0$. As $L \geq L_{p,\delta}$ and as $w_{J_i}^{(i)}$ has all nonzero entries by definition of J_i, we may apply Lemma 4.17 with $v = v_{J_i}$ and $\alpha = \alpha_i$ and $w = w_{J_i}^{(i)}$

and $d = |J_i|$. This gives

$$\begin{aligned} H(A\tau) &\geq \sum_{i=1}^{k+1} H\left(v_{J_i} + w_{J_i}^{(i)} \cdot \alpha_i\right) \\ &\geq \sum_{i=1}^{k+1} |J_i| \cdot \left(h(p) + \frac{h(p)}{L+\delta}\right) \\ &= L' \cdot \left(h(p) + \frac{h(p)}{L+\delta}\right), \end{aligned}$$

as desired. The last equality uses that $J_1, \ldots, J_{k+1}$ partition $[L']$. □

We now finish the proof of Theorem 4.5.

Proof of Theorem 4.5. Let $L_{p,\delta/2}$ be as in Lemma 4.18 and choose $\varepsilon_{p,\delta} \stackrel{\text{def}}{=} \frac{h(p)}{L_{p,\delta/2}+1}$. Fix $\varepsilon < \varepsilon_{p,\delta}$. Let $L = \lfloor \frac{h(p)}{\varepsilon} - \delta \rfloor$. Let τ be as in Definition 4.15 with this choice of L. By Lemma 4.18, as $L \geq L_{p,\delta/2}$, τ is not $\left(h(p) + \frac{h(p)}{L+\delta/2}\right)$-implicitly rare. Thus, as $\varepsilon \leq \frac{h(p)}{L+\delta} < \frac{h(p)}{L+\delta/2}$, there is some constant $c_{p,\varepsilon} > 0$ so that τ is not $(h(p) + \varepsilon + c_{p,\varepsilon})$-implicitly rare.

Then Theorem 4.14 with $\eta = c_{p,\varepsilon}$ tells us that, for n sufficiently large, a random linear code of rate $1 - (h(p) + \varepsilon + c_{p,\varepsilon}) + c_{p,\varepsilon} = 1 - h(p) - \varepsilon$ contains L codewords given by some matrix $M \in \mathcal{M}_{n,\tau}$ with probability at least $1 - 2^{-\Omega_{p,\varepsilon}(n)}$.

Finally, Proposition 4.16 implies that $\mathcal{C}$ is not $(p, L-1)$-list-decodable. Our choice of L proves the theorem. □

Acknowledgements

The list size upper bounds in this chapter appeared as [92] (conference version) and [93] (journal version) and is joint work with Mary Wootters. The list size lower bounds in this chapter appeared as [58] (conference version) and [59] (journal version) and is joint work with Venkatesan Guruswami, Jonathan Mosheiff, Nicolas Resch, Shashwat Silas, and Mary Wootters.

Chapter 5

List decoding: large alphabet

5.1 Introduction

Recall that the $[n, k]$-*Reed–Solomon code* over $\mathbb{F}_q$ with evaluation points $(\alpha_1, \ldots, \alpha_n)$ is defined as the set

$$\big\{\big(f(\alpha_1), \ldots, f(\alpha_n)\big) \,:\, f \in \mathbb{F}_q[x],\ \deg(f) < k\big\}\,.$$

Because (recall Theorem 2.7) RS codes attain the optimal rate versus unique-decoding-radius trade-off (and also because they admit efficient algorithms), they have been well-studied since their introduction in the 1960's [106]. However, perhaps surprisingly, there is still much about them that we do not know. One notable example is their *list-decodability* and more generally their *list-recoverability*. We discuss list-decodability first, and discuss list-recoverability after that.

List-Decodability of RS Codes. Recall that a code $\mathcal{C} \subset \mathbb{F}_q^n$ is (p, L)-list-decodable if for any $y \in \mathbb{F}_q^n$,

$$|\{c \in \mathcal{C} \,:\, d(c, y) \le p\}| \le L.$$

In particular, $(p, 1)$-list-decodability is the same as having distance greater than $2p$. List-decodability was introduced by Elias and Wozencraft in the 1950's [34, 127]. By now it is an important primitive in both coding theory and theoretical computer science more broadly. In general, larger *list sizes* (the parameter L) allow for a larger *list-decoding radius* (the parameter p). In this chapter, we are interested in the case when $p = 1 - \varepsilon$ is large.

The list-decodability of Reed–Solomon codes is of interest for several reasons. First, both list-decodability and Reed–Solomon codes are central notions in coding theory, and we believe that question is interesting in its own right. Moreover, the list-decodability of Reed–Solomon codes has found applications in complexity theory and pseudorandomness [17, 120, 96].

Until recently, the best bounds available on the list-decodability of RS codes were bounds that hold generically for any code. Recall (Theorem 2.12) the *Johnson bound* states that any p-unique-decodable code is $(1 - \sqrt{1 - 2p}, O(nq))$-list-decodable over an alphabet of size q ([82], see also [65, Theorem 7.3.3]). This implies that, for any $\varepsilon \in (0, 1]$, there are RS codes that are list-decodable up to radius $1 - \varepsilon$ (with polynomial list sizes) that have rate $\Omega(\varepsilon^2)$. The celebrated Guruswami–Sudan algorithm [66] gives an efficient algorithm

to list-decode RS codes up to the Johnson bound, but it breaks down at this point. Meanwhile, the *list-decoding capacity theorem* implies that no code (and in particular, no RS code) that is list-decodable up to radius $1-\varepsilon$ can have rate bounded above ε, unless the list sizes are exponential.

There have been several works over the past decade aimed at closing the gap between the Johnson bound (rate $\Theta(\varepsilon^2)$) and the list-decoding capacity theorem (rate $\Theta(\varepsilon)$). On the negative side, it is known that *some* RS codes (that is, some way of choosing the evaluation points $\alpha_1, \ldots, \alpha_n$), are not list-decodable substantially beyond the Johnson bound [7]. On the positive side, Rudra and Wootters [109] showed that a random choice of evaluation points will, with high probability, yield a code that is list-decodable up to radius $1-\varepsilon$ with rate $O\left(\frac{\varepsilon}{\log^5(1/\varepsilon)\log q}\right)$. Unfortunately, while the dependence on ε in the rate is nearly optimal (the "correct" dependence should be linear in ε, according to the list-decoding capacity theorem), the $\log q$ term in the denominator means that the rate necessarily goes to zero as n grows, as we must have $q \geq n$ for RS codes. Working in a different parameter regime, Shangguan and Tamo showed that over a large alphabet, there exist RS codes of rate larger than $1/9$ that can also be list-decoded beyond the Johnson bound (and in fact, optimally) [113]. However, this result only holds for small list sizes ($L = 2, 3$), and in particular, for such small list sizes one cannot hope to list-decode up to a radius $1-\varepsilon$ that approaches 1. Thus, there was still a substantial gap between capacity and the best known trade-offs for list-decoding RS codes.

List-Recoverability of RS Codes. The gap between capacity and the best known trade-offs for RS codes is even more pronounced for *list recovery,* a generalization of list decoding. We say that a code $\mathcal{C} \subset \mathbb{F}_q^n$ is (p, ℓ, L)-list-recoverable if for any $S_1, S_2, \ldots, S_n \subset \mathbb{F}_q$ with $|S_i| = \ell$,

$$|\{c \in \mathcal{C} : d(c, S_1 \times S_2 \times \cdots \times S_n) \leq p\}| \leq L.$$

Here, we extend the definition of Hamming distance to sets by denoting

$$d(c, S_1 \times \cdots \times S_n) = \frac{1}{n}|\{i \in [n] : c_i \notin S_i\}|.$$

The parameter ℓ is called the *input list size.* List-decoding is the special case of list-recovery for $\ell = 1$. List-recovery first arose in the context of list-decoding (for example, the Guruswami–Sudan algorithm mentioned above is in fact a list-recovery algorithm), but has since found applications beyond that, for example in pseudorandomness [68] and algorithm design [32].

Both the Johnson bound and the list-decoding capacity theorem have analogs for list-recovery. The list-recovery Johnson bound [67] implies that there are RS codes of rate $\Omega(\varepsilon^2/\ell)$ that are list-recoverable up to radius $1-\varepsilon$ with input list size ℓ and polynomial output list size. However, the list-recovery capacity theorem implies that there are codes of rate $\Omega(\varepsilon)$ (with *no* dependence on ℓ) that achieve the same guarantee, provided that the alphabet size q is sufficiently large.

Thus the gap for list-recovery (between rate $\Theta(\varepsilon^2/\ell)$ and $\Theta(\varepsilon)$) is even larger than that for list-decoding, and in particular the dependence on ℓ becomes important. To the best of our knowledge, before our work there were *no* results known for RS codes that established list-recovery up to arbitrarily large radius $1-\varepsilon$ with a better dependence on ℓ than $1/\ell$.

Motivating question. Given this state of affairs, our motivating question is whether or not RS codes can be list-decoded or list-recovered up to radius $1-\varepsilon$ with rates $\Omega(\varepsilon)$ (in particular, with a linear dependence

on ε and no dependence on the alphabet size q or the input list size ℓ). As outlined below, we nearly resolve this question for list-decoding and make substantial progress for list-recovery.

Subsequent work. After this work first appeared as [43], and inspired by the techniques in [43] and in [113], Ferber, Kwan, and Sauermann showed that there exist $(1-\epsilon, O(1/\epsilon))$-list-decodable RS codes with rate $\Omega(\epsilon)$ over a field size polynomial in the block length, improving our result for list-decoding [36]. In a very recent work, Goldberg, Shangguan, and Tamo further improved the rate of [36] by showing the existence of $(1-\epsilon, O(1/\epsilon))$-list-decodable RS codes with rate approaching $\frac{\epsilon}{2-\epsilon}$ [41]. See Section 5.1.2 for more details.

5.1.1 Contributions

Our main result establishes the list-recoverability (and in particular, the list-decodability), of Reed–Solomon codes up to radius $1-\varepsilon$, representing a significant improvement over previous work. Our techniques build on the approach of [113]; the main new technical contribution is a novel connection between list-decoding RS codes and the Nash-Williams–Tutte theorem in graph theory, which may be of independent interest. We outline our contributions below.

Existence of RS codes that are near-optimally list-decodable. Our main theorem for list-decoding is as follows.

Theorem 5.1 (RS codes with near-optimal list-decoding)**.** *There is a constant $c \geq 1$ so that the following statement holds. For any $\varepsilon \in (0,1]$ and any sufficiently large n, there exist RS codes of rate $R \geq \frac{\varepsilon}{c(\log(1/\varepsilon)+1)}$ over a large enough finite field (as a function of n and ε), that are $(1-\varepsilon, c/\varepsilon)$-list-decodable.*

As discussed above, Theorem 5.1 is stronger than the result of Rudra and Wootters [109], in that the result of [109] requires that the rate tend to zero as n grows, while ours holds for constant-rate codes. On the other hand, our result requires the field size q to be quite large (see Table 5.1), which [109] did not require.

Our result also differs from the result of Shangguan and Tamo [113] discussed above. Because that work focuses on small list sizes, it does not apply to list-decoding radii approaching 1. In contrast, we are able to list-decode up to radius $1-\varepsilon$. We note that [113] is able to show that RS codes are exactly optimal, while we are off by logarithmic factors. Both our work and that of [113] require large field sizes.

Generalization to list-recovery. Theorem 5.1 follows from a more general result about list-recovery. Our main result is the following (see Theorem 5.15 for a more detailed version).

Theorem 5.2 (RS codes with list-recovery beyond the Johnson bound)**.** *There is a constant $c \geq 1$ such that the following statement holds. For any $\epsilon \in (0,1]$, any positive integer ℓ, and any sufficiently large n, there exist RS codes with rate $R \geq \frac{\varepsilon}{c\sqrt{\ell}(\log(1/\epsilon)+1)}$ over a large enough (as a function of n, ϵ, and ℓ) finite field, that are $(1-\varepsilon, \ell, c\ell/\varepsilon)$-list-recoverable.*

Theorem 5.2 establishes list-recoverability for RS codes well beyond the Johnson bound, and in particular breaks the $1/\ell$ barrier. To the best of our knowledge, this is the first result to do so for radius arbitrarily close to 1, although we note that work of Lund and Potukuchi achieved a similar rate for small error radius [96]. We discuss related work below in Section 5.1.2 and summarize quantitative results in Table 5.1.

Table 5.1: Prior work on list-decoding and list-recovery of RS codes. Above, C refers to an absolute constant. The "Capacity" results refer to the list-decoding and list-recovery capacity theorems, respectively, and are impossibility results. Above, we assume that $q \geq n$ and that $n \to \infty$ is growing relative to $1/\varepsilon$ and ℓ, and that n is sufficiently large.

	Radius p	List size L	Rate R	Field size q
List-Decoding:				
Capacity	$1-\varepsilon$	-	$\leq \varepsilon$	-
Johnson bound	$1-\varepsilon$	$\text{poly}(n)$	$C\varepsilon^2$	$q \geq n$
[109]	$1-\varepsilon$	C/ε	$\frac{C\varepsilon}{\log^5(1/\varepsilon)\log(q)}$	$q \geq Cn\log^C(n/\varepsilon)/\varepsilon$
[113]	$\frac{L}{L+1}(1-R)$	$L = 2, 3$	R	$q = 2^{Cn}$
This chapter (Thm. 5.1)	$1-\varepsilon$	C/ε	$\frac{C\varepsilon}{\log(1/\varepsilon)}$	$q = \left(\frac{1}{\varepsilon}\right)^{Cn}$
List-Recovery:				
Capacity	$1-\varepsilon$	-	$\leq \varepsilon$	-
Johnson bound	$1-\varepsilon$	$\text{poly}(n)$	$\frac{C\varepsilon^2}{\ell}$	$q \geq n$
[96]	$p \leq 1-1/\sqrt{2}$	$C\ell$	$\frac{C}{\sqrt{\ell}\cdot\log q}$	$q \geq Cn\sqrt{\ell}\cdot\log n$
This chapter (Thm. 5.2)	$1-\varepsilon$	$\frac{C\ell}{\varepsilon}$	$\frac{C\varepsilon}{\sqrt{\ell}\cdot\log(1/\varepsilon)}$	$q = \left(\frac{\ell}{\varepsilon}\right)^{Cn}$

A new connection to the Nash-Williams–Tutte theorem, and a new hypergraph Nash-Williams—Tutte conjecture. In order to derive our results, we build on the framework of [113]. That work developed a framework to view the list-decodability of Reed–Solomon codes in terms of the singularity of *intersection matrices* (which we define in Section 5.2). The main new technical contribution of this chapter is to connect the singularity of these matrices to tree-packings in particular graphs. This connection allows us to use the Nash-Williams–Tutte theorem from graph theory to obtain our results. The Nash-Williams–Tutte theorem gives sufficient conditions for the existence of a large *tree packing* (that is, a collection of pairwise edge-disjoint spanning trees) in a graph.

We think that this connection is a contribution in its own right, and it is our hope that it will lead to further improvements to our results on Reed–Solomon codes. In particular, we hope that it will help establish the following conjecture of [113]:

Conjecture 5.3 (Conjecture 1.5 of [113])**.** *For any $\epsilon > 0$ and integers $1 \leq k < n$ with $\epsilon n \in \mathbb{Z}$, there exist RS codes with rate $R = \frac{k}{n}$ over a large enough (as a function of n and ϵ) finite field, that are list-decodable from radius $1 - R - \epsilon$ and list size at most $\lceil \frac{1-R-\epsilon}{\epsilon} \rceil$.*

Conjecture 5.3 is stronger than our Theorem 5.1 about list-decoding. In particular, our theorem is near-optimal, but it is interesting mostly in the low-rate/high-noise parameter regime. In contrast, Conjecture 5.3 conjectures that there exist *exactly* optimal RS codes, in any parameter regime.

To encourage others to use our new connection and make progress on Conjecture 5.3, we propose a method of attack in Section 5.5. This outline exploits our connection to the Nash-Williams–Tutte theorem, and proceeds via a conjectured generalization of the Nash-Williams–Tutte theorem to hypergraphs: we show that establishing this hypergraph conjecture (which is stated as Conjecture 5.25 in Section 5.5) would in fact establish Conjecture 5.3. In Section 5.5, we give evidence for our hypergraph Nash-Williams–Tutte

conjecture, Conjecture 5.25, observing that the "easy direction" of the conjecture follows from the Nash-Williams–Tutte theorem, and also that a quantitative relaxation of the conjecture follows from existing work [25, 18]. As further evidence of the viability of this approach, this quantitative relaxation implies a second proof of our main list-decoding result, Theorem 5.1, and we also sketch this proof in Section 5.5.[1]

5.1.2 Related Work

We briefly review related work. See Table 5.1 for a quantitative comparison to prior work.

List-decoding of RS codes. Ever since the Guruswami–Sudan algorithm [66], which efficiently list-decodes RS codes up to the Johnson bound, it has been open to understand the extent to which RS codes are list-decodable *beyond* the Johnson bound, and in particular if there are RS codes that are list-decodable all the way up to the list-decoding capacity theorem, matching the performance of completely random codes. There have been negative results that show that *some* RS codes are not list-decodable to capacity [7], and others that show that even if they were, in some parameter regimes we are unlikely to find an efficient list-decoding algorithm [22]. The work of Rudra and Wootters, mentioned above, showed that for any code with suitably good distance, a random puncturing of that code was likely to be near-optimally list-decodable; this implies that an RS code with random evaluation points is likely to be list-decodable. Unfortunately, as discussed above, this result requires a constant alphabet size q in order to yield a constant-rate code, while RS codes necessarily have $q \geq n$.

Recently, Shangguan and Tamo [113] studied the list-decodability of RS codes in a different parameter regime, namely when the list size L is very small, either 2 or 3. They were able to get extremely precise bounds on the rate (showing that there are RS codes that are exactly optimal), but unfortunately for such small list sizes, it is impossible for any code to be list-decodable up to radius $1 - \varepsilon$ for small ε, which is our parameter regime of interest. Unlike the approach of [109], which applies to random puncturings of any code, the work of [113] targeted RS codes specifically and developed an approach via studying *intersection matrices*. The reason that their approach stopped at $L = 3$ was the difficulty of analyzing these intersection matrices. We build on their approach and use techniques from graph theory—in particular, the Nash-Williams–Tutte theorem—to analyze the relevant intersection matrices beyond what [113] were able to do. We discuss our approach more below in Section 5.1.3.

Subsequent work on list-decoding of RS codes. After the results of this chapter first appeared as [43], and inspired by our approach, Ferber, Kwan, and Sauermann [36] gave a beautiful proof establishing the existence of RS codes with rate $\Omega(\epsilon)$ that are list-decodable from radius $1-\epsilon$ with list size $O(1/\epsilon)$, over a polynomially (in the codes length) large finite field.[2] Compared with our result on the list-decodability of RS codes, their result removes the logarithmic factor in $1/\varepsilon$, and allows for smaller alphabet sizes; additionally, their proof is much shorter. In further follow-up work, Goldberg, Shangguan, and Tamo [41] further improved the rate from $\Omega(\varepsilon)$ to a rate approaching $\frac{\varepsilon}{2-\varepsilon}$.

However, we believe that there are still some advantages to our approach (beyond inspiring that of [36] and [41]). First, the result of [36] does not apply to list-recovery, and while [41] does apply to list-recovery,

[1] This second proof does not immediately establish list-recoverability, which is why we focus on our first proof.

[2] In fact, they show something more general: if one begins with any code of sufficiently large distance over a sufficiently large alphabet, and randomly punctures it to rate $\Omega(\varepsilon)$, the resulting code is with high probability $(1-\varepsilon, O(1/\varepsilon))$ list-decodable.

they do not surpass the $1/\ell$ barrier in the rate. Second, neither [36] nor [41] fully resolve Conjecture 5.3 about optimal list-decodability of RS codes. We believe that the framework and tools developed in this chapter together with Conjecture 5.25 provide a plausible attack method to resolve Conjecture 5.3.

List-recovery of RS codes. While the Guruswami–Sudan algorithm is in fact a list-recovery algorithm, much less was known about the list-recovery of RS codes beyond the Johnson bound than was known about list-decoding. (There is a natural extension of the Johnson bound for list-recovery, see [67]; for RS codes, it implies that an RS code of rate about ε^2/ℓ is list-recoverable up to radius $1-\varepsilon$ with input list sizes ℓ and polynomial output list size). As with list-decoding, it is known that *some* RS codes are not list-recoverable beyond the Johnson bound [44]. However, much less was known on the positive front. In particular, neither of the works [109, 113] discussed above work for list-recovery. In a recent work, Lund and Potuchuki [96] have proved an analogous statement to that of [109]: any code of decent distance, when randomly punctured to an appropriate length, yields with high probability a good list-recoverable code. This implies the existence of RS codes that are list-recoverable beyond the Johnson bound. However, in [96] there is again a dependence on $\log(q)$ in the rate bound, meaning that for RS codes, the rate must be sub-constant. Further, the work of [96] only applies up to radius $p = 1 - 1/\sqrt{2}$, and in particular does not apply to radii $p = 1-\varepsilon$, as we study in this chapter. Our results also work in the constant-p setting of [96], and in that regime we show that RS codes of rate $\Omega(1/\sqrt{\ell})$ are $(p, \ell, O(\ell))$ list-recoverable, which improves over the result of [96] by a factor of $\log q$ in the rate. However, we do require the field size to be much larger than that is required by [96] (see Table 5.1).

Subsequent work on list-recovery of RS codes. The recent work of Goldberg, Shangguan, and Tamo [41] mentioned above builds on [36], and shows that there are RS codes of rate approaching $\frac{\varepsilon}{1+\ell-\varepsilon}$ that are $(1-\varepsilon, \ell, L_{\varepsilon,\ell})$-list-recoverable, for a constant $L_{\varepsilon,\ell}$ that depends only on ε and ℓ. Compared to our work, while [41] improves the dependence on ε in the rate by a factor of $\log(1/\varepsilon)$, it has a worse dependence on ℓ, and in particular does not break the $1/\ell$ barrier that is present in the Johnson bound.

List-decoding and list-recovery of RS-like codes. There are constructions—for example, of *folded RS codes* and *univariate multiplicity codes* [64, 45, 87, 86]—of codes that are based on RS codes and that are known to achieve list-decoding (and list-recovery) capacity, with efficient algorithms. Our goal in this chapter is to study Reed–Solomon codes themselves.

5.1.3 Technical Overview

Intersection matrices. Our approach is centered around *intersection matrices,* introduced in [113]. Intersection matrices and their nonsingularity are defined formally below in Definition 5.7, but we give a brief informal introduction here. A t-wise intersection matrix, M, is defined by a collection of sets $I_1, I_2, \ldots, I_t \subseteq [n]$, and has entries that are monomials in $\mathbb{F}_q[x_1, x_2, \ldots, x_n]$. It was shown in [113] that if there is a counterexample to the list-decodability of a Reed–Solomon code with evaluation points $(\alpha_1, \ldots, \alpha_n)$—that is, if there exist polynomials $f_1, f_2, \ldots, f_{L+1}$ that all agree with some other polynomial $g : \mathbb{F}_q \to \mathbb{F}_q$ at many points α_i—then there is a $(L+1)$-wise intersection matrix that becomes singular when α_i is plugged in for x_i for all $i \in [n]$.

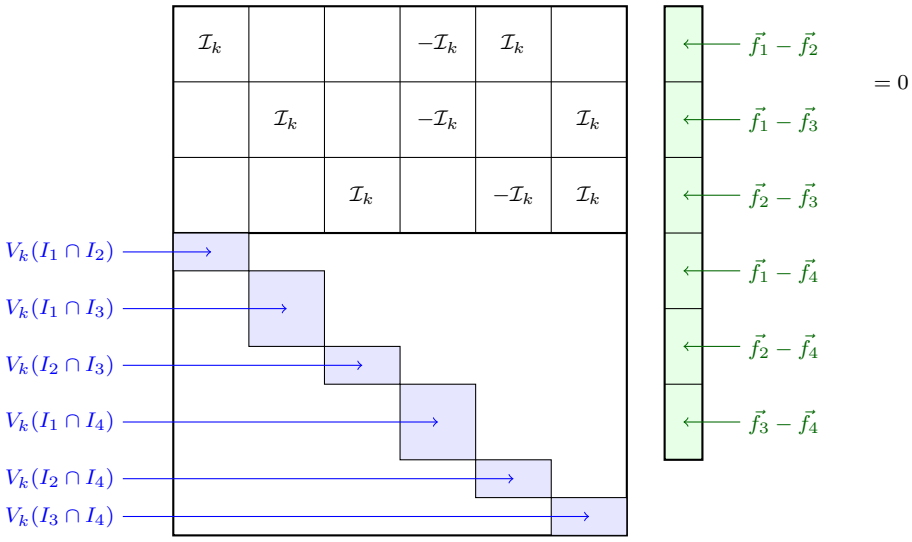


Figure 5.1: Let $f_1, f_2, f_3, f_4 \in \mathbb{F}_q[x]$ have degree $k-1$ and suppose that $I_j = \{s : f_j(\alpha_s) = g(\alpha_s)\}$. (In particular, f_i and f_j agree on $I_i \cap I_j$). Then the matrix-vector product depicted above is zero, where the vector $\vec{f_i}$ refers to the k coefficients of the polynomial f_i, and the j-th coordinate of this vector is the coefficient of x^{j-1} in f. Here, $V_k(I_i \cap I_j) \in \mathbb{F}_q^{|I_i \cap I_j| \times k}$ denotes the Vandermonde matrix whose rows are $[\alpha_s^0, \alpha_s^1, \ldots, \alpha_s^{k-1}]$ for $s \in I_i \cap I_j$. The notation $\mathcal{I}_k$ denotes the $k \times k$ identity matrix.

The set-up (both the definition of an intersection matrix and the connection to list-decoding) is most easily explained by an example. Suppose that we are interested in list-decoding for $L = 3$, and suppose that we are interested in a RS code with evaluation points $\alpha_1, \alpha_2, \ldots, \alpha_n$. Let f_1, f_2, f_3, f_4 and g be a counter-example to list-decoding, as above, and for $1 \leq j \leq 4$, let $I_j = \{i \in [n] : f_j(\alpha_i) = g(\alpha_i)\}$. Now consider the product shown in Figure 5.1 (see the caption for notation).

An inspection of Figure 5.1 shows that the matrix-vector product depicted is zero. Indeed, the top part is zero for any choice of the f_i, and the bottom part is zero since f_i and f_j are assumed to agree on $\{\alpha_s : s \in I_i \cap I_j\}$. The matrix shown is the 4-wise intersection matrix for the sets I_1, I_2, I_3, I_4, evaluated at $\alpha_1, \ldots, \alpha_n$. If the f_i's agree too much with the function g (i.e., if they are a counter-example to list-decodability for some given radius), then the sets $I_i \cap I_j$ are going to be larger, and this matrix will have more rows. In particular, the more the f_i's agree with g, the harder it is for this matrix to be singular. Intuitively, this sets us up for a proof by contradiction: if f_1, f_2, f_3, f_4 agree too much with g, then this matrix is nonsingular (at least for a non-pathological choice of α_i's); but Figure 5.1 displays a kernel vector!

A t-wise intersection matrix (for sets $I_1, \ldots, I_t$) generalizes a 4-wise intersection matrix shown in Figure 5.1. The bottom part looks exactly the same—a block-diagonal matrix with Vandermonde blocks—and the top part is an appropriate generalization that causes the analogous $k \cdot \binom{t}{2}$-long vector corresponding to the f_i's to vanish.

A conjecture about t-wise intersection matrices. With the motivation in Figure 5.1, the strategy of [113] was to study t-wise intersection matrices M for $t = L + 1$, and to show that for every appropriate choice of $I_1, \ldots, I_t$, the polynomial $\det(M) \in \mathbb{F}_q[x_1, x_2, \ldots, x_n]$ is not identically zero. The list-decodability of RS codes would then follow from the DeMillo–Lipton–Schwartz-Zippel lemma along with a counting argument. In particular, they made the following conjecture, and showed that it implies Conjecture 5.3

about list-decoding. Below, the *weight* of a family of subsets $I_1, \dots, I_t$ of $[n]$ is defined to be

$$\mathrm{wt}(I_1, \dots, I_t) = \sum_{i=1}^{t} |I_i| - \left| \bigcup_{i=1}^{t} I_i \right|, \tag{5.1}$$

and for a set J of indices, we use the shorthand $\mathrm{wt}(I_J) := \mathrm{wt}(I_j : j \in J)$.

Conjecture 5.4 (Conjecture 5.7 of [113])**.** *Let $t \geq 3$ be an integer and $I_1, \dots, I_t \subset [n]$ be subsets satisfying*

(i) $\mathrm{wt}(I_J) \leq (|J| - 1)k$ *for all nonempty* $J \subseteq [t]$,

(ii) *Equality holds for* $J = [t]$, *i.e.,* $\mathrm{wt}(I_{[t]}) = (t-1)k$.

Then the t-wise intersection matrix $M_{k,(I_1,\dots,I_t)}$ is nonsingular over any finite field.

The conditions (i) and (ii) above turn out to be the right way of quantifying "the f_i's agree enough with g." That is, if the f_i's agree too much with g (in the sense of going beyond Conjecture 5.3 about list-decoding), then it is possible to find sets I_j so that (i) and (ii) hold.

Unfortunately, the work of [113] was only able to establish Conjecture 5.4 for $t = 3, 4$ (corresponding to $L = 2, 3$), and it seemed challenging to extend their techniques directly to much larger values of L.

Establishing the conjecture under an additional assumption, and using that to establish our main results. In this chapter, we use a novel connection to the Nash-Williams–Tutte theorem, which establishes the existence of pairwise edge-disjoint spanning trees in a graph, to extend the results of [113] to larger L, at the cost of an additional assumption. More precisely, we are able to show in Theorem 5.9 (stated and proved in Section 5.3) that Conjecture 5.4 holds, *provided* that all three-wise intersections $I_i \cap I_j \cap I_\ell$ of the sets I_j are empty.

The connection to the Nash-Williams–Tutte theorem is explained in Section 5.3. Briefly, we consider each term in the expression

$$\det(M) = \sum_{\sigma \in S_n} (-1)^{\mathrm{sgn}(\sigma)} \prod_{i=1}^{n} M_{i,\sigma(i)}.$$

We show that $\prod_{i=1}^{n} M_{i,\sigma(i)}$ is a nonzero monomial in $x_1, \dots, x_n$ if and only if σ picks out a tree packing of a graph[3] that is determined by the sets $I_1, \dots, I_t$. It turns out that the requirements of (i) and (ii) in Conjecture 5.4 translate exactly into the requirements needed to apply the Nash-Williams–Tutte theorem to this graph. Thus, if (i) and (ii) hold, then there exists a tree packing in this graph and hence a nonzero term in $\det(M)$.

If the sets $I_i \cap I_j$ and $I_{i'} \cap I_{j'}$ that appear in the lower part of the t-wise intersection matrix do not intersect (that is, if there are no three-wise intersections among the sets I_j), then the reasoning above is enough to establish the conclusion of Conjecture 5.4, because all of the terms that appear in the expansion of the determinant are distinct monomials, and they cannot cancel. This is why Theorem 5.9 has this assumption.

In order to apply Theorem 5.9 to list-decoding, we back off from Conjecture 5.4 a bit. First, we allow a factor of $\Theta(\log t)$ slack on the right-hand sides of (i) and (ii). Second, rather than showing that the t-wise intersection matrix $M_{k,(I_1,\dots,I_t)}$ is nonsingular, we show that there exists a t'-wise intersection matrix that

[3] Throughout this chapter, a tree packing of a graph G means a collection of pairwise edge-disjoint spanning trees of G.

is nonsingular for some $t' < t$. Following the connection of [113] illustrated in Figure 5.1, this turns out to be enough to establish our main theorem on list-decoding/recovery.

We choose this smaller intersection matrix in Lemma 5.17 by carefully choosing a random subset J of $[t]$. By greedily removing elements from the sets $\{I_j : j \in J\}$, we can obtain subsets $I'_j \subset I_j$ with empty three-wise intersections $I'_j \cap I'_{j'} \cap I'_{j''} = \emptyset$. Furthermore, by the careful random choice of J, and since we allowed a $\Theta(\log t)$ slack in the initial weight bounds, we can show this step does not delete too many elements. This is the key step of Lemma 5.17. Using some of the sets $\{I_j : j \in J\}$, we can find a smaller intersection matrix obeying the setup of Conjecture 5.4 with the additional guarantee that all three-wise intersections are empty. We provide a more detailed summary of the proof in Section 5.4.1.

Another avenue to list-decoding: a hypergraph Nash-Williams–Tutte conjecture. Extending our connection of list-decoding RS codes to the Nash-Williams–Tutte theorem, we show that a suitable *hypergraph* generalization of the Nash-Williams–Tutte theorem would imply Conjecture 5.4 about the nonsingularity of intersection matrices, without any need for an additional assumption about three-wise intersections of the sets I_j.

We conjecture that such a generalization is true, and we state it in Section 5.5 as Conjecture 5.25. It requires a bit of notation to set up, so we do that in Section 5.5 rather than here; however, the reader interested in the hypergraph conjecture can at this point jump straight to Section 5.5 without missing anything.

We show that if our hypergraph conjecture were true, it would imply Conjecture 5.4, on the nonsingularity of intersection matrices (Theorem 5.26). This in turn would imply Conjecture 5.3, establishing the existence of RS codes with optimal list-decodability. This suggests a plan of attack towards Conjecture 5.3.

While we are unable to establish this challenging conjecture in full, we give some evidence for it. First, we show that the "easy part" of the conjecture follows from the Nash-Williams–Tutte theorem. Second, we observe that a quantitative relaxation of the conjecture follows from known results on Steiner tree packings [25] and disjoint bases of polymatroids [18]. This relaxation can be combined with the connection of hypergraph packings and intersection matrices established in Theorem 5.26, and the connection between intersection matrices and list decoding RS codes, to give a second proof of Theorem 5.1, that there are *near*-optimally list-decodable RS codes.

In addition to implying the optimal list-decodability of RS codes, Conjecture 5.25 may be of independent interest. A hypergraph generalization of Nash-Williams–Tutte is known for *partition-connected* hypergraphs [37] (see Section 5.5 for definition), a well studied notion. However, for a different notion called *weak-partition-connectivity*, less seems to be known, and Conjecture 5.25 poses a Nash-Williams–Tutte generalization for weakly-partition-connected hypergraphs.

Organization. A graphical overview of our results can be found in Figure 5.2. We begin in Section 5.2 with the needed notation and definitions, including the definition of t-wise intersection matrices.

In Sections 5.3, and 5.4, we prove Theorem 5.1 and Theorem 5.2 using our proof of Conjecture 5.4 under the additional assumption of no three-wise intersections. More precisely, in Section 5.3, we show how to use the Nash-Williams–Tutte theorem from graph theory to prove Theorem 5.9, which establishes Conjecture 5.4 under the assumption of no three-wise intersections. In Section 5.4 we prove Theorem 5.2 (and thus Theorem 5.1) on the list-recoverability (list-decodability) of RS codes.

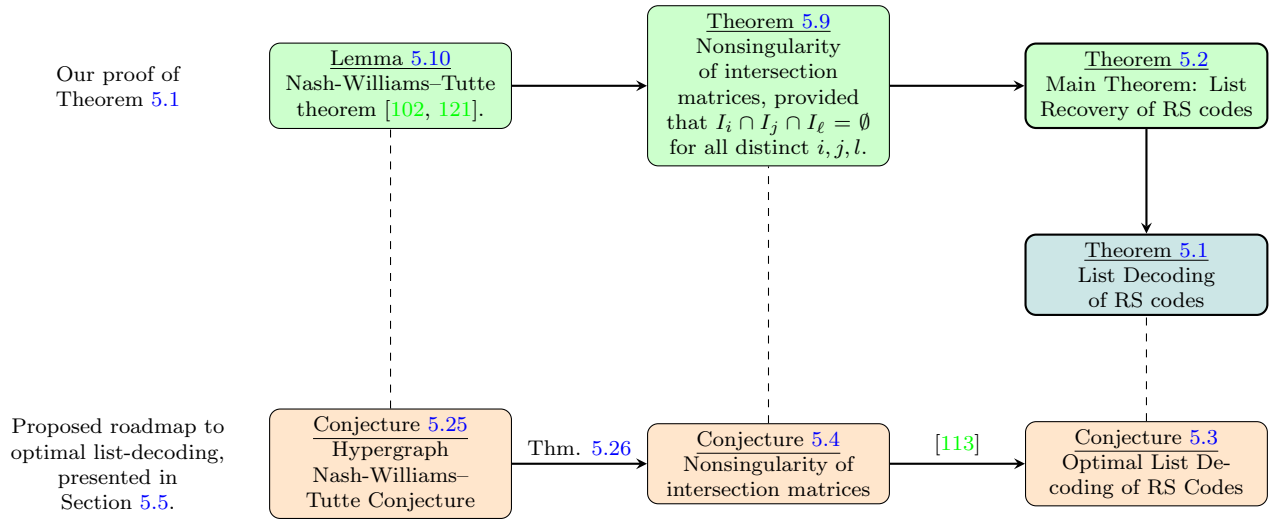


Figure 5.2: A diagram of the results and conjectures presented in this chapter. Solid arrows represent logical implications. Dashed lines indicate how the proposed roadmap to optimal list decoding parallels our proof of Theorem 5.1.

In Sections 5.5, we conjecture a hypergraph generalization of the Nash-Williams–Tutte theorem and prove (Theorem 5.26) that it implies optimal list-decodability of RS codes. We also give some evidence for it by observing an "easy direction" follows from the ordinary Nash-Williams–Tutte theorem, by highlighting a known relaxation, and sketching how this relaxation gives us a second proof of Theorem 5.1.

5.2 Preliminaries

For a finite set X and an integer $1 \leq k \leq |X|$, let $\binom{X}{k} = \{A \subseteq X : |A| = k\}$ be the family of all k-subsets of X. For an integer $t \geq 3$, we define the following lexicographic order on $\binom{[t]}{2}$. For distinct $S_1, S_2 \in \binom{[t]}{2}$, $S_1 < S_2$ if and only if $\max(S_1) < \max(S_2)$ or $\max(S_1) = \max(S_2)$ and $\min(S_1) < \min(S_2)$. For a partition $\mathcal{P}$ of X, let $|\mathcal{P}|$ denote the number of parts of $\mathcal{P}$.

We view a polynomial $f \in \mathbb{F}_q[x]$ of degree at most $k-1$ as a vector of length k defined by its k coefficients, where for $1 \leq i \leq k$, the i-th coordinate of this vector is the coefficient of x^{i-1} in f. By abuse of notation that vector is also denoted by f. We use the following well-known result.

Lemma 5.5 (DeMillo-Lipton-Schwartz-Zippel lemma, see, e.g., [83] Lemma 16.3)**.** *A nonzero polynomial $f \in \mathbb{F}_q[x_1, \ldots, x_n]$ of degree d has at most dq^{n-1} zeros in $\mathbb{F}_q^n$.*

The main goal of this section is to present the definition of t-wise intersection matrices over an arbitrary field $\mathbb{F}$.

5.2.1 Cycle Spaces

We need the notion of the *cycle space* of a graph, which is typically defined over the boolean field $\mathbb{F}_2$ (see, e.g., [28]). Here we define it over an arbitrary field $\mathbb{F}$. An equivalent definition can be found in [8], where it is called the "circuit-subspace".

Let K_t be the undirected complete graph with the vertex set $[t]$. Denote by $\{i,j\}$ the edge connecting vertices i and j. Let K_t^o be the oriented graph obtained by replacing $\{i,j\}$ with the directed edge (i,j) for all $1 \le i < j \le t$. For a graph G with vertex set $[t]$, an *oriented cycle* in G is a set of directed edges of the form

$$C = \{(i_0,i_1),(i_1,i_2),\dots,(i_{m-1},i_m)\}$$

where $m \ge 3$, $i_0,\dots,i_{m-1}$ are distinct, $i_m = i_0$ and $\{i_{j-1},i_j\}$ is an edge of G for all $j = 1,\dots,m$.

Suppose C is a union of edge-disjoint oriented cycles in G. Then C is uniquely represented by a vector $u^C = (u^C_{\{i,j\}} : \{i,j\} \in \binom{[t]}{2}) \in \mathbb{F}^{\binom{t}{2}}$, defined for $1 \le i < j \le t$ by

$$u^C_{\{i,j\}} = \begin{cases} 1 & (i,j) \in C, \\ -1 & (j,i) \in C, \\ 0 & \text{else.} \end{cases}$$

Hence, the sign of a nonzero coordinate $u^C_{\{i,j\}}$ indicates whether the orientation of $\{i,j\}$ in C complies with its orientation in K_t^o. We further assume that the coordinates of u^C are ordered by the aforementioned lexicographic order on $\binom{[t]}{2}$.

Denote by $C(G) \subseteq \mathbb{F}^{\binom{t}{2}}$ the subspace spanned by the set of vectors

$$\{u^C : C \text{ is an oriented cycle in } G\}$$

over $\mathbb{F}$. We call $C(G)$ the *cycle space* of G over $\mathbb{F}$. We are particularly interested in the cycle space $C(K_t)$ of K_t. For distinct $i,j,\ell \in [t]$, denote by $\Delta_{ij\ell}$ the oriented cycle $\{(i,j),(j,\ell),(\ell,i)\}$ and call it an *oriented triangle*. We have the following lemma, generalizing [28, Theorem 1.9.5].

Lemma 5.6. *The vector space $C(K_t) \subseteq \mathbb{F}^{\binom{t}{2}}$ has dimension $\binom{t-1}{2}$, and the set*

$$\mathcal{B}_t = \{u^{\Delta_{ijt}} : 1 \le i < j \le t-1\}$$

is a basis of $C(K_t)$.

Proof. The vectors in $\mathcal{B}_t$ are linearly independent since $u^{\Delta_{ijt}}_{\{i,j\}} = 1$ and $u^{\Delta_{ijt}}_{\{i',j'\}} = 0$ for $1 \le i < j \le t-1$ and $1 \le i' < j' \le t-1$ with $\{i,j\} \ne \{i',j'\}$. Let W be the span of $\mathcal{B}_t$ over $\mathbb{F}$. Consider an arbitrary oriented cycle C in K_t. We claim that $u^C \in W$, and this would imply that $\mathcal{B}_t$ is a basis of $C(K_t)$ and that the dimension of $C(K_t)$ is $|\mathcal{B}_t| = \binom{t-1}{2}$.

Denote by e_C the smallest $\{i,j\} \in \binom{[t]}{2}$ in the lexicographic order such that $(i,j) \in C$ or $(j,i) \in C$. Next, we will prove the claim by a reverse induction on the lexicographic order of e_C. Note that $t \notin e_C$ since $|C| \ge 3$, which implies that the claim is vacuously true when $e_C = \{t-1,t\}$ (which never occurs). Now assume that the claim holds for all oriented cycles C' with $e_{C'} > e_C$. Let $\{i,j\} = e_C$, where $i < j$. We may assume that $(i,j) \in C$ by flipping the orientation of C if necessary, which corresponds to negating u^C.

Let s be the number of directed edges that C and Δ_{ijt} share. If $s = 3$ then it is clear by definition that $C = \Delta_{ijt}$, and we are done. Otherwise, $1 \le s \le 2$ and it is easy to verify that $u^C - u^{\Delta_{ijt}} = u^{C'}$ for a set C' that is either an oriented cycle in G or a disjoint union of two oriented cycles C_1, C_2 in G passing through t. The latter case occurs when C passes through t and $(t,i),(j,t) \notin C$. In either case, the smallest

edge (under the lexicographic order) of C' is greater than the edge $e_C = \{i, j\}$. Hence, by the induction hypothesis and the fact that $u^{C'} = u^{C_1} + u^{C_2}$ when C' is the disjoint union of C_1 and C_2, we have $u^{C'} \in W$. So $u^C = u^{C'} + u^{\Delta_{ijt}} \in W$, completing the proof of the claim. □

The basis $\mathcal{B}_t$ is also viewed as a $\binom{t-1}{2} \times \binom{t}{2}$ matrix over $\mathbb{F}$ whose columns are labeled by the edges $\{i, j\}$ of K_t, according to the lexicographic order defined above. Moreover, the rows of $\mathcal{B}_t$ represent $u^{\Delta_{ijt}}$ for $1 \le i < j \le t-1$, and are labeled by $\{i, j\} \in \binom{[t-1]}{2}$, also according to the lexicographic order. For example, $\mathcal{B}_3 = (1, -1, 1)$ and

$$\mathcal{B}_4 = \begin{pmatrix} 1 & & -1 & 1 & \\ & 1 & -1 & & 1 \\ & & 1 & -1 & 1 \end{pmatrix},$$

where the 6 columns are labeled and ordered lexicographically by $\{1,2\} < \{1,3\} < \{2,3\} < \{1,4\} < \{2,4\} < \{3,4\}$. Observe for example that the ± 1 entries in the first row correspond to the oriented triangle $\Delta_{124} = \{(1,2),(2,4),(4,1)\}$, where we have -1 on the column labeled by the edge $\{1,4\}$, since the directed edge $(4,1)$ in Δ_{124} has the opposite orientation from the orientation of the edge in K_t^o.

We remark that the above definition of $\mathcal{B}_t$, is given with respect to the fixed orientation of the edges of K_t^o, as with the definition of u^C for any oriented cycle C. One may define $\mathcal{B}_t$ with respect to other orientations of edges, which corresponds to changing the signs in some columns. These definitions are all equivalent and the analysis in this chapter holds for any orientation up to change of signs.

Moreover, when the characteristic of $\mathbb{F}$ is two, we recover the definition of $\mathcal{B}_t$ in [113] using the fact that $1 = -1$. While working in the case $\mathrm{char}(\mathbb{F}) = 2$ has the advantage that there is no need to distinguish the signs, the theory holds more generally over any field.

5.2.2 t-Wise Intersection Matrices

We proceed to define t-wise intersection matrices, but we begin with a few preliminary definitions. Given n variables or field elements $x_1, \ldots, x_n$, define the $n \times k$ Vandermonde matrix

$$V_k(x_1, \ldots, x_n) = \begin{pmatrix} 1 & x_1 & \cdots & x_1^{k-1} \\ & & \ddots & \\ 1 & x_n & \cdots & x_n^{k-1} \end{pmatrix}.$$

When the x_i's are understood from the context, for $I \subseteq [n]$, we use the abbreviation $V_k(I) := V_k(x_i : i \in I)$ to denote the restriction of $V_k(x_1, \ldots, x_n)$ to the rows with indices in I.

Let $\mathcal{I}_k$ denote the identity matrix of order k. Next, we give the definition of t-wise intersection matrices.

Definition 5.7 (t-wise intersection matrices)**.** For a positive integer k and $t \ge 3$ subsets $I_1, \ldots, I_t \subseteq [n]$, the t-wise intersection matrix $M_{k,(I_1,\ldots,I_t)}$ is the $(\binom{t-1}{2}k + \sum_{1 \le i < j \le t} |I_i \cap I_j|) \times \binom{t}{2}k$ variable matrix with entries in $\mathbb{F}[x_1, \ldots, x_n]$, defined as

$$\left(\begin{array}{c} \mathcal{B}_t \otimes \mathcal{I}_k \\ \hline \operatorname{diag}\Big(V_k(I_i \cap I_j) : \{i,j\} \in \binom{[t]}{2}\Big) \end{array}\right),$$

where $\otimes$ is tensor product of matrices and

- $\mathcal{B}_t \otimes \mathcal{I}_k$ is a $\binom{t-1}{2}k \times \binom{t}{2}k$ matrix with entries in $\{0, \pm 1\}$,
- $\operatorname{diag}\big(V_k(I_i \cap I_j) : \{i,j\} \in \binom{[t]}{2}\big)$ is a block diagonal matrix with blocks $V_k(I_i \cap I_j)$, ordered by the lexicographic order on $\{i,j\} \in \binom{[t]}{2}$. Note that this matrix has order $(\sum_{1 \le i < j \le t} |I_i \cap I_j|) \times \binom{t}{2}k$. If $I_i \cap I_j = \emptyset$ for some i, j, then $V_k(I_i \cap I_j)$ is of order $0 \times k$ and the $\{i,j\} \in \binom{[t]}{2}$ block of k columns is a $\sum_{1 \le i < j \le t} |I_i \cap I_j| \times k$ zero matrix.

Below, we provide an example of a 4-wise intersection matrix. We note that when $t = 2$, $\mathcal{B}_t$ is an empty matrix and $M_{k,(I_1,I_2)}$ is simply a Vandermonde matrix.

Example 5.8 (4-wise intersection matrices)**.** Given four subsets $I_1, I_2, I_3, I_4 \subseteq [n]$, the 4-wise intersection matrix $M_{k,(I_1,I_2,I_3,I_4)}$ is the $(3k + \sum_{1 \le i < j \le 4} |I_i \cap I_j|) \times 6k$ variable matrix

$$\left(\begin{array}{cccccc} \mathcal{I}_k & & & -\mathcal{I}_k & \mathcal{I}_k & \\ & \mathcal{I}_k & & -\mathcal{I}_k & & \mathcal{I}_k \\ & & \mathcal{I}_k & & -\mathcal{I}_k & \mathcal{I}_k \\ \hline V_k(I_1 \cap I_2) & & & & & \\ & V_k(I_1 \cap I_3) & & & & \\ & & V_k(I_2 \cap I_3) & & & \\ & & & V_k(I_1 \cap I_4) & & \\ & & & & V_k(I_2 \cap I_4) & \\ & & & & & V_k(I_3 \cap I_4) \end{array}\right).$$

For a vector $\alpha \in \mathbb{F}^n$, the evaluation of $M_{k,(I_1,\ldots,I_t)}$ at the vector α is denoted by $M_{k,(I_1,\ldots,I_t)}(\alpha)$, where each variable x_i is assigned the value α_i. Given subsets $I_1, \ldots, I_t \subseteq [n]$, we call the variable matrix $M_{k,(I_1,\ldots,I_t)}$ *nonsingular* if it contains at least one $\binom{t}{2}k \times \binom{t}{2}k$ submatrix whose determinant is a nonzero polynomial in $\mathbb{F}[x_1, \ldots, x_n]$.

The paper [113] connects the nonsingularity of intersection matrices to the list-decodability of RS codes. We will use this connection to prove our main result, Theorem 5.2.

However, we will first prove that *certain* intersection matrices are nonsingular. This will both allow us to cleanly illustrate the connection to disjoint tree packings of graphs. We will do this in Theorem 5.9 in the next section.

5.3 Connection to Tree Packing and an Intermediate Result

In this section we prove the following theorem. We recall from (5.1) the definition of the *weight* of a collection of sets:

$$\mathrm{wt}(I_1, \ldots, I_t) = \sum_{i=1}^{t} |I_i| - \left| \bigcup_{i=1}^{t} I_i \right|,$$

Theorem 5.9. *Let $t \geq 2$ be an integer and $I_1, \ldots, I_t \subset [n]$ be subsets satisfying* (i) $I_i \cap I_j \cap I_l = \emptyset$ for all $1 \leq i < j < l \leq t$; (ii) $\mathrm{wt}(I_J) \leq (|J| - 1)k$ for all nonempty $J \subseteq [t]$; (iii) $\mathrm{wt}(I_{[t]}) = (t-1)k$. Then the t-wise intersection matrix $M_{k,(I_1,\ldots,I_t)}$ is nonsingular over any field.

As discussed above, this theorem stops short of Conjecture 5.4, due to the assumption that $I_i \cap I_j \cap I_\ell = \emptyset$. In the language of list-decoding Reed–Solomon codes, this only gives us a statement about lists of potential codewords that have no three-wise intersections. However, we will build on this statement to prove our main theorem about list-recovery (Theorem 5.2).

The main tool of proving Theorem 5.9 is the following classical result in graph theory.

Lemma 5.10 (Nash-Williams [102], Tutte [121], see also Theorem 2.4.1 of [28])**.** *A multigraph contains k edge-disjoint spanning trees if and only if for every partition $\mathcal{P}$ of its vertex set it has at least $(|\mathcal{P}| - 1)k$ cross-edges. Here an edge is called a cross-edge for $\mathcal{P}$ if its two endpoints are in different members of $\mathcal{P}$.*

In order to apply the Nash-Williams–Tutte theorem, we will construct a graph G from the sets $I_1, I_2, \ldots, I_t$. We first note that the assumptions on $I_1, \ldots, I_t$ from Theorem 5.9 imply some nice properties that will later allow us to apply Lemma 5.10.

Claim 5.11. *Suppose that $I_1, \ldots, I_t$ are subsets satisfying the assumptions of Theorem 5.9. Then the matrix $M_{k,(I_1,\ldots,I_t)}$ is a square matrix of order $\binom{t}{2}k$. Further, for any $J \subseteq [t]$ with $|J| \geq 2$,*

$$\mathrm{wt}(I_J) = \sum_{\{i,j\} \in \binom{J}{2}} |I_i \cap I_j|. \tag{5.2}$$

Proof. By (5.1) (the definition of weight) and the inclusion-exclusion principle

$$\mathrm{wt}(I_{[t]}) = \sum_{i=1}^{t} |I_i| - \left| \bigcup_{i=1}^{t} I_i \right| = \sum_{j=2}^{t} \sum_{J \in \binom{[t]}{j}} (-1)^{|J|} \left| \bigcap_{i \in J} I_i \right|.$$

Therefore, by assumption (i) of Theorem 5.9 we have $\mathrm{wt}(I_{[t]}) = \sum_{1 \leq i < j \leq t} |I_i \cap I_j|$. Then, by assumption (iii) the matrix $M_{k,(I_1,\ldots,I_t)}$ is in fact a square matrix of order $\binom{t}{2}k$. Similarly, (5.2) holds for any $J \subseteq [t]$ with $|J| \geq 2$. □

To prove Theorem 5.9, let us construct a multigraph G defined on a set V of t vertices, say $V = \{v_1, \ldots, v_t\}$. For $1 \leq i < j \leq t$, connect vertices v_i, v_j by $|I_i \cap I_j|$ multiple edges.

Applying Lemma 5.10 to G leads to the following claim.

Claim 5.12. *Let G be as above. Then G contains k edge-disjoint spanning trees.*

Proof. Let $\mathcal{P} = \{V_1, \ldots, V_s\}$ be an arbitrary partition of V. Then it is clear that $\sum_{i=1}^{s} |V_i| = t$. According to Lemma 5.10, to prove the claim it suffices to show that G has at least $(s-1)k$ cross-edges with respect to $\mathcal{P}$. By (5.2) and assumption (iii) of Theorem 5.9 it is easy to see that G contains $\sum_{1 \leq i < j \leq t} |I_i \cap I_j| = (t-1)k$ edges. Moreover, by (5.2) and assumption (ii) of Theorem 5.9 one can infer that for each $i \in [s]$, the induced

subgraph of G on the vertex set V_i has at most $\mathrm{wt}(I_j : j \in V_i) \leq (|V_i|-1)k$ edges. It follows that the number of cross-edges of G (with respect to $\mathcal{P}$) is at least

$$(t-1)k - \sum_{i=1}^{s}(|V_i|-1)k = \left(t-1-\sum_{i=1}^{s}|V_i|+s\right)k = (s-1)k,$$

as needed, thereby completing the proof of the claim. □

Below, we will relate a tree packing of this graph G to the determinant of the intersection matrix $M_{k,(I_1,\ldots,I_t)}$. In order to do this, we first record a property of the matrix $\mathcal{B}_t$. Recall that the columns of $\mathcal{B}_t$ are indexed by $\binom{[t]}{2}$.

Claim 5.13. *Removing a set of columns from $\mathcal{B}_t$ will not reduce its row rank if and only if the columns are labeled by an acyclic subgraph of K_t.*

Proof. First we prove the if direction. Assume to the contrary that we can remove from $\mathcal{B}_t$ some columns labeled by an acyclic subgraph H of K_t and reduce the row rank. Let $\mathcal{B}'_t$ be the submatrix of $\mathcal{B}_t$ after the removal of the columns labeled by H. The rows of $\mathcal{B}'_t$ are linearly dependent by assumption. Hence, there exists a nonzero vector $u \in \mathbb{F}^{\binom{t-1}{2}}$ such that $u \cdot \mathcal{B}'_t = 0$. As $u \neq 0$ and the rows of $\mathcal{B}_t$ are linearly independent, we have $u \cdot \mathcal{B}_t \neq 0$. Let $S \subseteq \binom{[t]}{2}$ be the support of $u \cdot \mathcal{B}_t$, where the support of a vector of length n is the subset of $[n]$ that records the indices of its nonzero coordinates. As $u \cdot \mathcal{B}_t \neq 0$ and $u \cdot \mathcal{B}'_t = 0$, we have $\emptyset \neq S \subseteq H$.

Consider the $\binom{t}{2} \times t$ matrix $D = (D_{\{i,j\},s})$ which is defined by

$$D_{\{i,j\},s} = \begin{cases} 1 & s = j, \\ -1 & s = i, \\ 0 & \text{otherwise}, \end{cases}$$

where $1 \leq i < j \leq t$ and $s \in [t]$. Note that the rows and columns of D are labeled by $\{i,j\} \in \binom{[t]}{2}$ and $s \in [t]$ respectively. It is easy to verify that $\mathcal{B}_t \cdot D = 0$, which implies that $u \cdot \mathcal{B}_t \cdot D = 0$. Denote $u \cdot \mathcal{B}_t$ by $w = (w_{\{i,j\}}) \in \mathbb{F}^{\binom{t}{2}}$, whose support is S. As $\emptyset \neq S \subseteq H$ and H is acyclic, we can find $s_0 \in [t]$ whose degree in S is one, i.e., there exists a unique edge $\{i_0, j_0\} \in S$ such that $s_0 \in \{i_0, j_0\}$. Then, the s_0-th entry of $w \cdot D$ is

$$\sum_{\{i,j\} \in \binom{[t]}{2}} w_{\{i,j\}} D_{\{i,j\},s_0} = w_{\{i_0,j_0\}} D_{\{i_0,j_0\},s_0} = \pm w_{\{i_0,j_0\}} \neq 0,$$

which is a contradiction as $w \cdot D = u \cdot \mathcal{B}_t \cdot D = 0$.

Now we prove the only if direction. It suffices to prove that removing from $\mathcal{B}_t$ a set of columns labeled by a cycle C of K_t will reduce its row rank by at least 1. Let us orient the edges of C to make it an oriented cycle, which by abuse of notation is also denoted by C. Since the rows of $\mathcal{B}_t$ form a basis of $C(K_t)$, there is a nonzero vector $u \in \mathbb{F}^{\binom{t-1}{2}}$ such that $u \cdot \mathcal{B}_t = u^C$. Let $\mathcal{B}'_t$ be the submatrix of $\mathcal{B}_t$ after the removal of the columns labeled by C. Then it is not hard to check that $u \cdot \mathcal{B}'_t = 0$, which implies that the rows of $\mathcal{B}'_t$ are linearly dependent, as needed. □

Next we present the proof of Theorem 5.9. Recall from Claim 5.11 that under the assumptions of Theorem

5.9, the t-wise intersection matrix

$$M_{k,(I_1,\ldots,I_t)} = \left(\frac{\mathcal{B}_t \otimes \mathcal{I}_k}{\operatorname{diag}\Big(V_k(I_i \cap I_j) : \{i,j\} \in \binom{[t]}{2}\Big)} \right),$$

is a square matrix of order $\binom{t}{2}k$, and is defined by exactly $(t-1)k$ variables x_s, $s \in S$, where $S \subseteq [n]$ is some subset of size $(t-1)k$. In order to prove that $M_{k,(I_1,\ldots,I_t)}$ is nonsingular, we proceed to show the nonsingularity of the following matrix, obtained by permuting the columns and rows of $M_{k,(I_1,\ldots,I_t)}$:

$$M'_{k,(I_1,\ldots,I_t)} := \left(\frac{\mathcal{I}_k \otimes \mathcal{B}_t}{\big(\mathbb{C}_i : 0 \le i \le k-1\big)} \right),$$

where $\mathbb{C}_i = \operatorname{diag}\Big(V_k^{(i)}(I_j \cap I_{j'}) : \{j,j'\} \in \binom{[t]}{2}\Big)$ and $V_k^{(i)}(I_j \cap I_{j'})$ is the $(i+1)$-th column of $V_k(I_j \cap I_{j'})$. Above, $\big(\mathbb{C}_i : 0 \le i \le k-1\big)$ is a $(t-1)k \times \binom{t}{2}k$ variable matrix, which consists of the matrices $\mathbb{C}_i$ stacked next to each other. See Figure 5.3 for an illustration, and Example 5.14 below for a concrete example.

Example 5.14. For $k = 2$, instead of considering the following 4-wise intersection matrix

$$\left(\frac{\mathcal{B}_4 \otimes \mathcal{I}_2}{\operatorname{diag}(V_k(I_i \cap I_j) : \{i,j\} \in \binom{[4]}{2}} \right) = \left(\begin{array}{cc|cc|cc|cc|cc|cc} 1 & & & & & & -1 & & 1 & & & \\ & 1 & & & & & & -1 & & 1 & & \\ \hline & & 1 & & & & -1 & & & & 1 & \\ & & & 1 & & & & -1 & & & & 1 \\ \hline & & & & 1 & & & & -1 & & 1 & \\ & & & & & 1 & & & & -1 & & 1 \\ \hline 1 & x_1 & & & & & & & & & & \\ & & 1 & x_2 & & & & & & & & \\ & & & & 1 & x_3 & & & & & & \\ & & & & & & 1 & x_4 & & & & \\ & & & & & & & & 1 & x_5 & & \\ & & & & & & & & & & 1 & x_6 \end{array}\right),$$

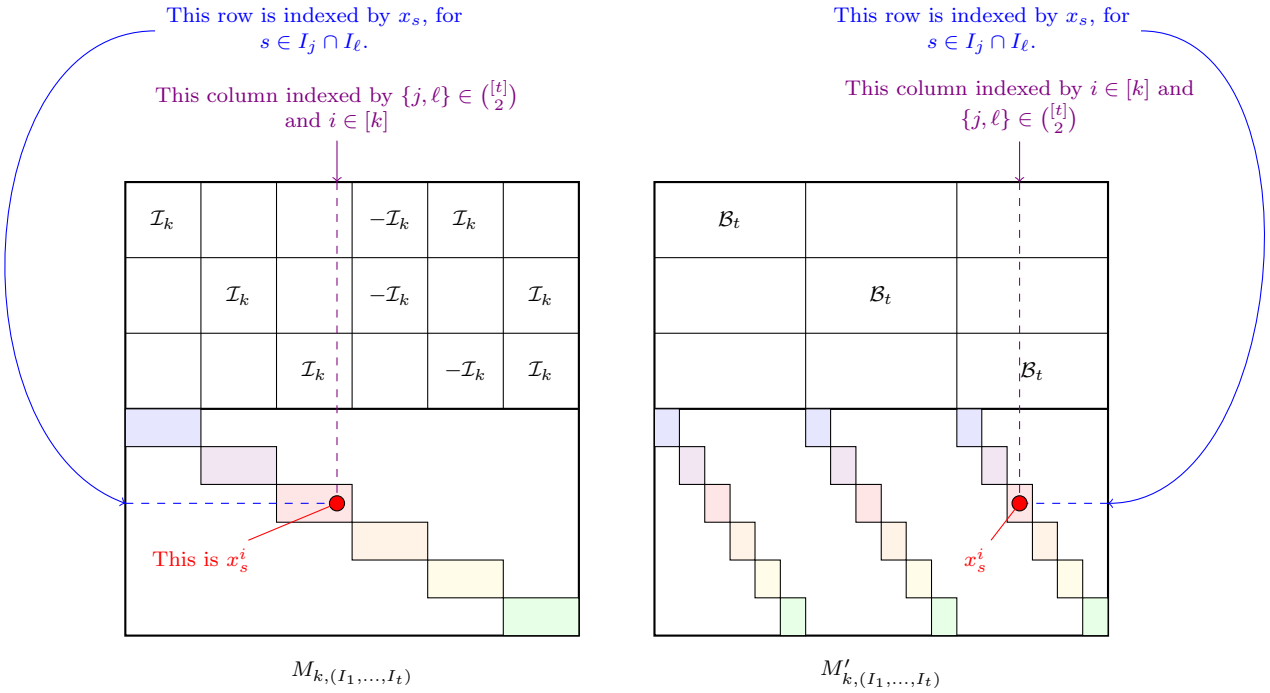


Figure 5.3: Re-ordering the rows/columns of an intersection matrix. (In this cartoon, $t = 4$ and $k = 3$).

we turn to prove the nonsingularity of

$$\left(\begin{array}{c} \mathcal{I}_2 \otimes \mathcal{B}_4 \\ \hline (\mathcal{C}_i : 0 \le i \le 1) \end{array}\right) = \left(\begin{array}{cccccc|cccccc} 1 & & & -1 & 1 & & & & & & & \\ & 1 & & -1 & & 1 & & & & & & \\ & & 1 & & -1 & 1 & & & & & & \\ \hline & & & & & & 1 & & & -1 & 1 & \\ & & & & & & & 1 & & -1 & & 1 \\ & & & & & & & & 1 & & -1 & 1 \\ \hline 1 & & & & & & x_1 & & & & & \\ & 1 & & & & & & x_2 & & & & \\ & & 1 & & & & & & x_3 & & & \\ & & & 1 & & & & & & x_4 & & \\ & & & & 1 & & & & & & x_5 & \\ & & & & & 1 & & & & & & x_6 \end{array}\right).$$

Proof of Theorem 5.9. If $t = 2$, then $M_{k,(I_1,I_2)}$ is a $k \times k$ Vandermonde matrix, which is nonsingular, so assume $t \ge 3$. For the rest of the proof, we will consider the matrix $M' = M'_{k,(I_1,\ldots,I_t)}$ discussed above, and

show that it is nonsingular.

Let the graph G be as in the discussion above; recall that for distinct $i, j \in [t]$, two vertices v_i, v_j in G are connected by $|I_i \cap I_j|$ edges. By Claim 5.11, M' is a square matrix with $\binom{t}{2}k$ rows and columns, and $k(t-1)$ "variable" rows at the bottom. Let $S \subseteq [n]$ be the subset that records the indices of variables x_s that appear in M'.

For $1 \leq i < j \leq t$, fix an arbitrary one-to-one correspondence between the $|I_i \cap I_j|$ edges connecting v_i, v_j and the $|I_i \cap I_j|$ variables $x_s \in S$ so that $s \in I_i \cap I_j$. Since any three distinct subsets I_i, I_j, I_ℓ have empty intersection, this yields a one-to-one correspondence

$$\phi : E(G) \longrightarrow \{x_s : s \in S\},$$

between the $(t-k)k$ edges of G and the $(t-1)k$ variables with indices in S.

By Claim 5.12, the edges of G can be partitioned into k edge-disjoint spanning trees T_i, and $G = \bigcup_{i=0}^{k-1} T_i$.

Observe that for each $0 \leq i \leq k-1$, $\mathcal{C}_i$ has entries that are either zero or of the form x_s^i for some $x_s \in S$. We will show how to use the tree decomposition of G to choose nonzero entries in each $\mathcal{C}_i$ so that (a) every row in the bottom part of M' is chosen exactly once, and (b) when the columns chosen are removed from M', the resulting submatrix of $\mathcal{B}_t$ is nonsingular. This will mean that the product of these non-zero entries appears in the determinant expansion of M'.

For each i, we pick $t-1$ non-zero elements from each $\mathcal{C}_i$: we choose x_s^i for $x_s \in \{\phi(e) : e \in T_i\}$. That is, we consider all of the variables x_s corresponding to edges that appear in T_i. Let $m_i(x)$ denote the product of these entries:

$$m_i(x) = \prod_{x_s \in \{\phi(e)\ :\ e \in T_i\}} x_s^i.$$

Let $m(x) = \prod_{i=0}^{k-1} m_i(x)$. Since ϕ is a bijection, $m(x)$ is a product of $(t-1)k$ distinct entries chosen from the submatrix $\big(\mathbb{C}_i : 0 \leq i \leq k-1\big)$, and crucially, no two of them appear in the same row or column.

To conclude the proof, it is enough to show that $m(x)$ appears as a nonvanishing term in the determinant expansion of $M'_{k,(I_1,\ldots,I_t)}$. Indeed, removing from $M'_{k,(I_1,\ldots,I_t)}$ the $(t-1)k$ rows and columns that correspond to $m(x)$, the resulting submatrix is a block diagonal matrix

$$\operatorname{diag}\Big(\mathcal{B}'_t(i) :\ 0 \leq i \leq k-1\Big),$$

where for each i, $\mathcal{B}'_t(i)$ is a square submatrix of $\mathcal{B}_t$ of order $\binom{t-1}{2}$. By construction, each $\mathcal{B}'_t(i)$ is obtained by removing from $\mathcal{B}_t$ a set of $t-1$ columns labeled by the spanning tree T_i. By Claim 5.13, this implies that $\mathcal{B}'_t(i)$ is nonsingular. Moreover, as each of the sets $I_i \cap I_j$ are disjoint due to the assumptions of the theorem, the monomial $m(x)$ appears only once in the determinant expansion of $M'_{k,(I_1,\ldots,I_t)}$. Consequently, the "coefficient" of $m(x)$ in the determinant expansion of $M'_{k,(I_1,\ldots,I_t)}$ is nonvanishing, completing the proof of the theorem. □

5.4 Near-Optimal List-Recovery of RS Codes: Proof of the Main Theorem

In this section we prove our main theorem on the list-recovery of RS codes, Theorem 5.2. We will in fact prove the following theorem, which implies Theorem 5.2.

Theorem 5.15. *Let $k, n, L, \ell \in \mathbb{N}^+$, $\epsilon \in (0,1]$, and $\delta > 0$ be such that $L \geq (1+\delta)\ell/\epsilon - 1$ and*

$$k/n \leq \frac{\epsilon}{c\sqrt{\ell}(\frac{1+\delta}{\delta})(\log(\frac{1}{\epsilon}) + \log(\frac{1+\delta}{\delta}) + 1)},$$

where $c > 0$ is the constant in Lemma 5.17. Consider the RS code

$$\mathcal{C} = \{(f(\alpha_1), \ldots, f(\alpha_n)) : f(x) \in \mathbb{F}_q[x], \ \deg(f) < k\}$$

where $q \geq 2^{c'(L+n\log L)}$ for a large enough constant $c' > 0$ and $\alpha_1, \ldots, \alpha_n$ are chosen uniformly and independently from $\mathbb{F}_q$ at random. Then with high probability, the code $\mathcal{C}$ has rate $R = k/n$ and is list-recoverable up to relative distance $1-\epsilon$ with input list size ℓ and output list size L. In particular, by choosing δ to be any positive constant, we could achieve $L = O(\ell/\epsilon)$ and $R = \Omega\left(\frac{\epsilon}{\sqrt{\ell}(\log(1/\epsilon)+1)}\right)$.

We begin with an overview of the proof.

5.4.1 Overview of the Proof

We give an overview of our proof of Theorem 5.15. For simplicity, let us first assume the input list size ℓ equals one, i.e., we restrict to the case of list decoding. In this case, Theorem 5.15 states that there exist RS codes of rate $\Omega(\frac{\epsilon}{\log(1/\epsilon)+1})$ that are list-decodable from radius $1-\epsilon$ with list size $O(1/\epsilon)$.

As discussed previously, Conjecture 5.4 about the nonsingularity of intersection matrices would be enough to establish Theorem 5.15, and indeed an even stronger result. While we do not know if Conjecture 5.4 holds in general, Theorem 5.9 states that it holds under an extra condition that $I_i \cap I_{i'} \cap I_{i''} = \emptyset$ for distinct $i, i', i'' \in [t]$. Our proof of Theorem 5.15 is based on this theorem.

As Theorem 5.9 requires the above extra condition, which does not hold in general, we cannot simply follow the proof in [113] and replace Conjecture 5.4 by Theorem 5.9. One naive way of fixing this is removing elements from the sets I_i until the condition $I_i \cap I_{i'} \cap I_{i''} = \emptyset$ for distinct $i, i', i'' \in [t]$ is satisfied. Specifically, for each $j \in [n]$ such that there exist more than two sets $I_{i_1}, \ldots, I_{i_s}$ containing j, we pick two sets (say I_{i_1} and I_{i_2}) and remove j from all the other sets. The resulting sets $I'_1, \ldots, I'_t$ satisfy the condition $I'_i \cap I'_{i'} \cap I'_{i''} = \emptyset$ for distinct $i, i', i'' \in [t]$ and we can now apply Theorem 5.9 to conclude that $M_{k,(I'_1,\ldots,I'_t)}$ is nonsingular.

The problem with this idea, however, is that $\mathrm{wt}(I'_{[t]})$ is generally much smaller than $\mathrm{wt}(I_{[t]})$, possibly by a factor of $\Theta(t) = \Omega(1/\epsilon)$. So in order to achieve $\mathrm{wt}(I'_{[t]}) \geq (t-1)k$ as required by Theorem 5.9,[4] we need to start with sets I_i such that $\mathrm{wt}(I_{[t]}) \gg (t-1)k$. As a consequence, implementing this idea directly only yields RS codes of rate $\Omega(\epsilon^2)$.

To mitigate this problem, we perform a random sampling of the collection $\{I_1, \ldots, I_t\}$ before removing elements from I_i. Namely, we choose a random subset $J \subseteq [t]$ of some appropriate cardinality to be

[4] Theorem 5.9 requires the stronger condition $\mathrm{wt}(I'_{[t]}) = (t-1)k$, but this can be achieved by further removing elements from the sets I'_i.

determined later. Then, we remove elements from the sets I_i just like before, but only for $i \in J$, so that the resulting sets I'_i satisfy the condition $I'_i \cap I'_{i'} \cap I'_{i''} = \emptyset$ for distinct $i, i', i'' \in J$. Finally, we apply Theorem 5.9 to conclude that the $|J|$-wise intersection matrix $M_{k,(I'_i)_{i\in J}}$ is nonsingular, which can still be used to prove the list-decodability of the RS code.

The advantage of replacing $[t]$ by the random sample $J \subseteq [t]$ is that the condition $\mathrm{wt}(I'_{[t]}) \geq (t-1)k$ is replaced by $\mathrm{wt}(I'_J) \geq (|J|-1)k$. It turns out that the condition $I'_i \cap I'_{i'} \cap I'_{i''} = \emptyset$ for distinct $i, i', i'' \in J$ is easier to satisfy since $|J|$ may be much smaller than t. Consequently, we are able to show that there exist RS codes of rate $\Omega(\frac{\epsilon}{\log(1/\epsilon)+1})$ using this improved method.

Finally, we explain how to choose the cardinality of the sample J. Let $j \in [n]$ and denote by s_j the number of sets among $I_1, \ldots, I_t$ that contain j. Then for the index j, it is best to choose $|J| = \Theta(t/s_j)$. However, the number s_j may vary when j ranges over $[n]$, meaning that there may not be a single choice of $|J|$ that works best for all $j \in [n]$ simultaneously.

We solve this problem using the following trick: Create a logarithmic number of "buckets" and put $j \in [n]$ in the i-th bucket if $2^{i-1} \leq s_j < 2^i$. Then choose $|J|$ according to the heaviest bucket. Here, we lose a factor of $O(\log(1/\epsilon)+1)$ in the rate because there are about $\log L = O(\log(1/\epsilon)+1)$ buckets.

Generalization to list recovery. In the case of list decoding, we choose each set I_i to be the subset of coordinates where a codeword c_i and the received word y agree. In the more general setting of list recovery, there are multiple received words $y^{(1)}, \ldots, y^{(\ell)}$ in the input list, so we need to keep track of multiple sets $I_i^{(1)}, \ldots, I_i^{(\ell)}$ for each $i \in [t]$.

One way of extending our proof to list recovery is choosing $r \in [\ell]$ that maximizes $\mathrm{wt}(I^{(r)}_{[t]}) = \mathrm{wt}(I_1^{(r)}, \ldots, I_t^{(r)})$ and then proceeding as in the case of list decoding, with $I_1, \ldots, I_t$ replaced by $I_1^{(r)}, \ldots, I_t^{(r)}$. It is not hard to show that this yields RS codes of rate $\Omega(\frac{\epsilon}{\ell(\log(1/\epsilon)+1)})$ which are list-recoverable from radius $1-\epsilon$ with input list size ℓ and output list size $O(\ell/\epsilon)$.

With a more careful analysis, we show that we can achieve a better rate $\Omega(\frac{\epsilon}{\sqrt{\ell}(\log(1/\epsilon)+1)})$, as stated by Theorem 5.2. Our analysis is inspired by [96] which proved a similar result on the list-recoverability of randomly punctured codes with a different setting of parameters.

5.4.2 A Combinatorial Lemma

In this subsection, we state a combinatorial lemma (Lemma 5.17). It guarantees the existence of a subset $J \subseteq [t]$ and sets $I'_i \subseteq I_i$ for $i \in J$ that satisfy certain conditions, particularly the condition $I'_i \cap I'_{i'} \cap I'_{i''} = \emptyset$ for distinct $i, i', i'' \in J$. We then use this lemma together with Theorem 5.9 to prove Theorem 5.2. The proof of this combinatorial lemma is postponed to Subsection 5.4.4.

First, we need the following generalization of the weight function $\mathrm{wt}(\cdot)$.

Definition 5.16 (Generalized weight function)**.** Let $n, t \in \mathbb{N}^+$ and $I_1, \ldots, I_t \subseteq [n]$. Let $S_j = \{i \in [t] : j \in I_i\}$ for $j \in [n]$. For $J \subseteq [t]$ and $\ell \in \mathbb{N}^+$, define the *l-th generalized weight* $\mathrm{wt}_\ell(I_J)$ of I_J to be

$$\mathrm{wt}_\ell(I_J) := \sum_{j=1}^{n} \max\{|S_j \cap J| - \ell, 0\}.$$

Note that $\mathrm{wt}(I_J) = \sum_{i\in J}|I_i| - |\bigcup_{i\in J} I_i| = \mathrm{wt}_1(I_J)$. Also note that

$$\mathrm{wt}_\ell(I_J) \geq \sum_{j=1}^{n}(|S_j \cap J| - \ell) = \sum_{i\in J}|I_i| - \ell n. \tag{5.3}$$

The proof of Theorem 5.2 uses the following combinatorial lemma, which we prove in the next subsection.

Lemma 5.17. *Let $k, n, t, \ell \in \mathbb{N}^+$, $\epsilon \in (0,1]$, $\delta > 0$, and $I_1^{(r)}, \ldots, I_t^{(r)} \subseteq [n]$ for $r \in [\ell]$. Let $I_i = \bigcup_{r=1}^{\ell} I_i^{(r)}$ for $i \in [t]$. Suppose $t \geq (1+\delta)\ell/\epsilon$, $|I_i| \geq \epsilon n$ for $i \in [t]$, and*

$$\mathrm{wt}_\ell(I_{[t]}) \geq \left(c\sqrt{\ell}\left(\log\left(\frac{1}{\epsilon}\right) + \log\left(\frac{1+\delta}{\delta}\right) + 1\right)\right)\cdot tk.$$

where $c > 0$ is a large enough absolute constant. Then there exist $J \subseteq [t]$ and a collection $(I'_i)_{i\in J}$ of subsets of $[n]$ indexed by J such that $|J| \geq 2$, $I'_i \subseteq I_i$ for $i \in J$, and the following conditions are satisfied:

(1) $I'_i \cap I'_{i'} \cap I'_{i''} = \emptyset$ for distinct $i, i', i'' \in J$.

(2) $\mathrm{wt}(I'_{J'}) \leq (|J'|-1)k$ for all nonempty $J' \subseteq J$.

(3) $\mathrm{wt}(I'_J) = (|J|-1)k$.

(4) For every $j \in [n]$, there exists $r_j \in [\ell]$ such that $\{i \in J : j \in I'_i\} \subseteq \{i \in J : j \in I_i^{(r_j)}\}$.

Remark 5.18. Condition (4) is introduced for list recovery. For the case $\ell = 1$, which corresponds to list decoding, Condition (4) is automatically satisfied by choosing $r_j = 1$ for $j \in [n]$ since in this case $I'_i \subseteq I_i = I_i^{(1)}$ for $i \in J$.

We also need the following lemma that bounds the number of pairs $(J, (I'_i)_{i\in J})$.

Lemma 5.19. *The number of $(J, (I'_i)_{i\in J})$ satisfying Condition (1) of Lemma 5.17 is at most $2^t(1+t+\binom{t}{2})^n$.*

Proof. There are at most 2^t choices of J. Now fix $J \subseteq [t]$. For $j \in [n]$, let $T_j = \{i \in J : j \in I'_i\}$. Note that we have $|T_j| \leq 2$ for all $j \in [n]$ by Condition (1) of Lemma 5.17. So for each $j \in [n]$, the number of choices of T_j is at most $1 + t + \binom{t}{2}$. Also note that the sets I'_i are determined by the sets T_j by $I'_i = \{j \in [n] : i \in T_j\}$. So the number of choices of $(J, (I'_i)_{i\in J})$ is at most $2^t(1+t+\binom{t}{2})^n$. □

5.4.3 Proof of Theorem 5.15

Now we are ready to prove our main theorem. For the reader's convenience, we restate it below.

Theorem (Theorem 5.15, restated)**.** Let $k, n, L, \ell \in \mathbb{N}^+$, $\epsilon \in (0,1]$, and $\delta > 0$ such that $L \geq (1+\delta)\ell/\epsilon - 1$ and

$$k/n \leq \frac{\epsilon}{c\sqrt{\ell}(\frac{1+\delta}{\delta})(\log(\frac{1}{\epsilon}) + \log(\frac{1+\delta}{\delta}) + 1)},$$

where $c > 0$ is the constant in Lemma 5.17. Consider the RS code

$$\mathcal{C} = \{(f(\alpha_1), \ldots, f(\alpha_n)) : f(x) \in \mathbb{F}_q[x], \deg(f) < k\}$$

where $q \geq 2^{c'(L+n\log L)}$ for a large enough constant $c' > 0$ and $\alpha_1, \ldots, \alpha_n$ are chosen uniformly and independently from $\mathbb{F}_q$ at random. Then with high probability, the code $\mathcal{C}$ has rate $R = k/n$ and is list-recoverable

up to relative distance $1-\epsilon$ with input list size ℓ and output list size L. In particular, by choosing δ to be any positive constant, we could achieve $L = O(\ell/\epsilon)$ and $R = \Omega\left(\frac{\epsilon}{\sqrt{\ell}(\log(1/\epsilon)+1)}\right)$.

Proof. Let $t = L + 1$. Consider the following two conditions:

(1) $\alpha_i \neq \alpha_j$ for all distinct $i, j \in [n]$.

(2) For all $J \subseteq [t]$ and $(I'_i)_{i\in J}$ satisfying Conditions (1)–(3) of Lemma 5.17, we have

$$\det(M_{k,(I'_i)_{i\in J}}(\alpha_1, \ldots, \alpha_n)) \neq 0,$$

where $M_{k,(I'_i)_{i\in J}}$ denotes the $\binom{|J|}{2}k \times \binom{|J|}{2}k$ variable matrix[5]

$$M_{k,(I'_i)_{i\in J}} = \left(\frac{\mathcal{B}_{|J|} \otimes \mathcal{I}_k}{\operatorname{diag}\left(V_k(I'_i \cap I'_j) : \{i,j\} \in \binom{J}{2}\right)}\right).$$

The first condition is satisfied with probability at least $1 - \binom{n}{2}/q$. For the second condition, consider fixed $J \subseteq [t]$ and $(I'_i)_{i\in J}$ satisfying Conditions (1)–(3) of Lemma 5.17. We know $\det(M_{k,(I'_i)_{i\in J}}) \neq 0$ by Theorem 5.9. Also note that $\det(M_{k,(I'_i)_{i\in J}})$ is a multivariate polynomial of total degree at most $(|J|-1)k(k-1) \leq Lk^2$. So by Lemma 5.5, $\det(M_{k,(I'_i)_{i\in J}})(\alpha_1, \ldots, \alpha_n) \neq 0$ holds with probability at least $1 - Lk^2/q$ for fixed J and $(I'_i)_{i\in J}$. The number of choices of $(J, (I'_i)_{i\in J})$ is at most $2^t(1 + t + \binom{t}{2})^n$ by Lemma 5.19. By the union bound, the two conditions are simultaneously satisfied with probability at least

$$1 - \binom{n}{2}/q - 2^t\left(1 + t + \binom{t}{2}\right)^n Lk^2/q = 1 - o(1)$$

over the random choices of $\alpha_1, \ldots, \alpha_n$, where we use the assumption that $q \geq 2^{c'(L+n\log L)}$ and $c' > 0$ is a large enough constant.

Fix $\alpha_1, \ldots, \alpha_n \in \mathbb{F}_q$ that satisfy the above two conditions. By the first condition, the code $\mathcal{C}$ has rate exactly k/n. It remains to show that $\mathcal{C}$ is list-recoverable up to relative distance $1-\epsilon$ with input list size ℓ and output list size L. Assume to the contrary that this does not hold. Then there exist t distinct polynomials $f_1, \ldots, f_t \in \mathbb{F}_q[x]$ of degree less than k and ℓ received words $y^{(r)} = (y_1^{(r)}, \ldots, y_n^{(r)}) \in \mathbb{F}_q^n$, where $r = 1, 2, \ldots, \ell$, such that for all $i \in [t]$, the cardinality of the set

$$I_i := \{j \in [n] : \text{ there exists } r \in [\ell] \text{ such that } f_i(\alpha_j) = y_j^{(r)}\}$$

is at least ϵn.

Let $I_i^{(r)} = \{j \in [n] : f_i(\alpha_j) = y_j^{(r)}\}$ for $i \in [t]$ and $r \in [\ell]$, i.e., $I_i^{(r)}$ denotes the set of coordinates where $C(f_i) := (f_i(\alpha_1), \ldots, f_i(\alpha_n))$ and $y^{(r)}$ agree. So $I_i = \bigcup_{r=1}^{\ell} I_i^{(r)}$ for $i \in [t]$. As $t = L + 1 \geq (1+\delta)\ell/\epsilon$ and

[5]The number of rows of $M_{k,(I'_i)_{i\in J}}$ is $\binom{|J|-1}{2}k + \sum_{\{i,j\}\in\binom{J}{2}} |I'_i \cap I'_j|$, which equals $\binom{|J|-1}{2}k + \sum_{i\in J}|I'_i| - |\bigcup_{i\in J} I'_i| = \binom{|J|-1}{2}k + \operatorname{wt}(I'_J)$ by Condition (1) of Lemma 5.17. This number further equals $\binom{|J|-1}{2}k + (|J|-1)k = \binom{|J|}{2}k$ by Condition (3) of Lemma 5.17.

$k/n \le \frac{\epsilon}{c\sqrt{\ell}(\frac{1+\delta}{\delta})(\log(\frac{1}{\epsilon})+\log(\frac{1+\delta}{\delta})+1)}$, we also have

$$\mathrm{wt}_\ell(I_{[t]}) \overset{(5.3)}{\ge} t\epsilon n - \ell n \ge \frac{\delta}{1+\delta}\cdot t\epsilon n \ge \left(c\sqrt{\ell}\left(\log\left(\frac{1}{\epsilon}\right)+\log\left(\frac{1+\delta}{\delta}\right)+1\right)\right)\cdot tk.$$

By Lemma 5.17, there exist $J \subseteq [t]$ and $(I'_i)_{i\in J}$ such that $|J| \ge 2$, $I'_i \subseteq I_i$ for $i \in J$, and Conditions (1)–(4) of Lemma 5.17 are satisfied.

Let u be the vector $(f_{ij} : \{i,j\} \in \binom{J}{2}, i<j) \in \mathbb{F}_q^{\binom{|J|}{2}k}$, where $f_{ij} := f_i - f_j$. As $|J| \ge 2$ and $f_1,\dots,f_t$ are distinct, we have $u \neq 0$.

We claim that

$$M_{k,(I'_i)_{i\in J}}(\alpha_1,\dots,\alpha_n)\cdot u^T = 0. \tag{5.4}$$

To see this, first note that $(\mathcal{B}_{|J|}\otimes\mathcal{I}_k)\cdot u^T = 0$. Now consider a row v of the submatrix

$$\mathrm{diag}\Big(V_k(I'_i\cap I'_j) : \{i,j\} \in \binom{J}{2}\Big)(\alpha_1,\dots,\alpha_n),$$

of $M_{k,(I'_i)_{i\in J}}(\alpha_1,\dots,\alpha_n)$, which corresponds to some $\{i,j\} \in \binom{J}{2}$ with $i<j$ and $s \in I'_i \cap I'_j$. By definition, we have $v\cdot u^T = f_{ij}(\alpha_s)$, i.e., the row v represents the linear constraint $f_{ij}(\alpha_s) = 0$. By Condition (4) of Lemma 5.17, we have $s \in I_i^{(r_s)} \cap I_j^{(r_s)}$ for some $r_s \in [\ell]$, which implies $f_i(\alpha_s) = y_s^{(r_s)}$ and $f_j(\alpha_s) = y_s^{(r_s)}$. So $v\cdot u^T = f_{ij}(\alpha_s) = f_i(\alpha_s) - f_j(\alpha_s) = 0$. This proves (5.4).

By (5.4), we have $\det(M_{k,(I'_i)_{i\in J}})(\alpha_1,\dots,\alpha_n) = 0$. But this contradicts the choice of $\alpha_1,\dots,\alpha_n$. □

5.4.4 Proof of Lemma 5.17

We present the proof of Lemma 5.17 in this subsection.

Let $k,n,t,\ell \in \mathbb{N}^+$, $\epsilon \in (0,1]$, $\delta > 0$, and the sets $I_i^{(r)}$, $I_i \subseteq [n]$ for $i \in [t]$ and $r \in [\ell]$ be as in Lemma 5.17. That is, we have $t \ge (1+\delta)\ell/\epsilon$, $I_i = \bigcup_{r=1}^{\ell} I_i^{(r)}$ and $|I_i| \ge \epsilon n$ for $i \in [t]$, and

$$\mathrm{wt}_\ell(I_{[t]}) \ge \left(c\sqrt{\ell}\left(\log\left(\frac{1}{\epsilon}\right)+\log\left(\frac{1+\delta}{\delta}\right)+1\right)\right)\cdot tk. \tag{5.5}$$

where $c>0$ is a large enough absolute constant. We may assume without loss of generality that $I_i^{(1)},\dots,I_i^{(\ell)}$ are pairwise disjoint for all $i \in [t]$: if an element appears in both $I_i^{(r)}$ and $I_i^{(r')}$ for $r \neq r'$, we can remove it from one of them, and the set I_i does not change. Thus, if we can prove Lemma 5.17 when these pairwise disjoint conditions hold, we can prove it in general, since we can choose the same subsets $I'_i \subseteq I_i$ after removing redundant elements.

For $j \in [n]$, $S_j := \{i \in [t] : j \in I_i\}$. By definition, we have

$$\mathrm{wt}_\ell(I_{[t]}) = \sum_{j=1}^{n} \max\{|S_j| - \ell, 0\}. \tag{5.6}$$

Assume for a moment that there exists an integer $K \in \mathbb{N}^+$ such that $\max\{|S_j|-\ell,0\}$ equals either K or zero for all $j \in [n]$. Then by (5.6), the number of $j \in [n]$ for which $\max\{|S_j|-\ell,0\} = K$ holds (or equivalently, $|S_j| = K+\ell$ holds) is precisely $\mathrm{wt}(I_{[t]})/K$. The next lemma extends this fact to the general case with only

logarithmic loss.

Lemma 5.20. *There exists an integer $K > 0$ such that the number of $j \in [n]$ satisfying $|S_j| \geq K + \ell$ is at least $\frac{\mathrm{wt}_\ell(I_{[t]})}{c_0 K(\log(\frac{1}{\epsilon})+\log(\frac{1+\delta}{\delta})+1)}$, where $c_0 > 0$ is some absolute constant.*

Proof. By (5.3) and the fact that $t \geq (1+\delta)\ell/\epsilon$, we have

$$\mathrm{wt}_\ell(I_{[t]}) \geq t\epsilon n - \ell n \geq \frac{\delta}{1+\delta} \cdot t\epsilon n. \tag{5.7}$$

For $i = 0, 1, 2, \ldots$, let $B_i = \{j \in [n] : 2^i \leq |S_j| - \ell < 2^{i+1}\}$. Then

$$\mathrm{wt}_\ell(I_{[t]}) = \sum_{j=1}^{n} \max\{|S_j| - \ell, 0\} = \sum_{i=0}^{\lceil \log t \rceil - 1} \sum_{j \in B_i} (|S_j| - \ell).$$

Let $d = \lfloor \log(\frac{\delta}{1+\delta} \cdot t\epsilon/2) \rfloor$. Note that d could be negative (possibly $\frac{\delta}{1+\delta} \cdot t\epsilon/2 \in (0,1)$). Then

$$\sum_{0 \leq i < d} \sum_{j \in B_i} (|S_j| - \ell) \leq n2^d \leq \frac{\delta}{1+\delta} \cdot t\epsilon n/2 \overset{(5.7)}{\leq} \mathrm{wt}_\ell(I_{[t]})/2.$$

Therefore

$$\sum_{i=\max\{d,0\}}^{\lceil \log t \rceil - 1} \sum_{j \in B_i} (|S_j| - \ell) \geq \mathrm{wt}_\ell(I_{[t]})/2. \tag{5.8}$$

Let $\Delta = \lceil \log t \rceil - \max\{d, 0\} = O(\log(\frac{1}{\epsilon}) + \log(\frac{1+\delta}{\delta}) + 1)$. By (5.8), there exists an integer i_0 such that $\max\{d, 0\} \leq i_0 \leq \lceil \log t \rceil - 1$ and

$$\sum_{j \in B_{i_0}} (|S_j| - \ell) \geq \frac{\mathrm{wt}_\ell(I_{[t]})}{2\Delta}. \tag{5.9}$$

Choose $K = 2^{i_0}$. Then $K \leq |S_j| - \ell < 2K$ for all $j \in B_{i_0}$. The upper bound $|S_j| - \ell < 2K$ for $j \in B_{i_0}$, together with (5.9), implies $|B_{i_0}| \geq \frac{\mathrm{wt}_\ell(I_{[t]})}{4K\Delta}$. So the number of $j \in [n]$ satisfying $|S_j| \geq K + \ell$ is at least $\frac{\mathrm{wt}_\ell(I_{[t]})}{4K\Delta} = \Omega\left(\frac{\mathrm{wt}_\ell(I_{[t]})}{K(\log(\frac{1}{\epsilon})+\log(\frac{1+\delta}{\delta})+1)}\right)$. □

Fix K satisfying Lemma 5.20. Define

$$A := \{j \in [n] : |S_j| \geq K + \ell\} \subseteq [n].$$

By the choice of K and Lemma 5.20, we have

$$|A| \geq \frac{\mathrm{wt}_\ell(I_{[t]})}{c_0 K(\log(\frac{1}{\epsilon}) + \log(\frac{1+\delta}{\delta}) + 1)}. \tag{5.10}$$

For $j \in [n]$ and $r \in [\ell]$, let $S_j^{(r)} := \{i \in [t] : j \in I_i^{(r)}\}$. So $S_j = \bigcup_{r=1}^{\ell} S_j^{(r)}$ for $j \in [n]$. Note that $S_j^{(1)}, \ldots, S_j^{(\ell)}$ are pairwise disjoint for all j: if $i \in S_j^{(r)} \cap S_j^{(r')}$, we must have $j \in I_i^{(r)} \cap I_i^{(r')}$, but we have assumed that $I_i^{(1)}, \ldots, I_i^{(\ell)}$ are pairwise disjoint for all i.

We also need the following technical lemma.

Lemma 5.21. *For real numbers $p \in (0, \frac{1}{2}]$ and $x \geq 0$, we have $(1-p)^x(1+px) \leq 1 - \frac{1}{8}p^2x^2$ if $x \leq \frac{1}{p}$, and $(1-p)^x(1+px) \leq \frac{2}{e}$ otherwise.*

Proof. Fix p and let $f(y) = (1-p)^y(1+py)$. For $y \geq 0$, the derivative $f'(y)$ satisfies

$$\begin{aligned} f'(y) &= (1-p)^y(\ln(1-p)\cdot(1+py)+p) \\ &\leq (1-p)^y(-p(1+py)+p) \\ &= -p^2y(1-p)^y. \end{aligned} \tag{5.11}$$

So $f'(y) \leq -p^2y(1-p)^x$ for $y \in [0,x]$. As $f(0) = 1$, we have $f(x) \leq 1 + \int_0^x -p^2y(1-p)^x dy = 1 - \frac{1}{2}p^2x^2(1-p)^x$. If $x \leq \frac{1}{p}$, we have $(1-p)^x \geq (1-p)^{1/p} \geq 1/4$, as $(1-p)^{1/p}$ is decreasing with p, and thus is minimized at $p = \frac{1}{2}$. Hence, we have $f(x) \leq 1 - \frac{1}{8}p^2x^2$. By (5.11), $f(y)$ is decreasing and thus maximized at $y = 1/p$ on the interval $[1/p, \infty)$, so for $x \geq 1/p$, we have $f(x) \leq f(1/p) = (1-p)^{1/p}(1+1) \leq \frac{2}{e}$. □

The above lemma is used to prove the following statement.

Lemma 5.22. *Choose a random subset $J \subseteq [t]$ by independently including each $i \in [t]$ in J with probability $p = \min\{\sqrt{\ell}/(2K), \frac{1}{2}\}$. Let*

$$A_J := \{j \in A : \text{ there exists } r \in [\ell] \text{ such that } |S_j^{(r)} \cap J| \geq 2\}.$$

Then $\mathrm{ex}[|A_J|] = \Omega(|A|)$.

Proof. Fix $j \in A$. It suffices to prove that $\mathbf{Pr}[j \in A_J] \geq c$ for some constant c. Let

$$t_r \coloneqq \max\{|S_j^{(r)}| - 1, 0\}$$

for $r = 1, \ldots, \ell$. Let $K' \coloneqq \sum_{r=1}^{\ell} t_j$. Since $j \in A$ and $S_j = \bigcup_{r=1}^{\ell} S_j^{(r)}$, we have

$$K' = \sum_{r=1}^{\ell} t_j \geq |S_j| - \ell \geq K.$$

For all $r = 1, \ldots, \ell$, we have

$$\mathbf{Pr}[|S_j^{(r)} \cap J| \leq 1] = (1-p)^{t_r}(1+t_rp)$$

This is because the probability is exactly $(1-p)^{t_r+1} + (t_r+1)p(1-p)^{t_r}$ when $t_r \geq 1$, and is exactly 1 when $t_r = 0$.

As $S_j^{(1)}, \ldots, S_j^{(\ell)}$ are disjoint, the events that $|S_j^{(r)} \cap J| \geq 2$ are independent. Thus, the probability that $j \in A_J$ is

$$\begin{aligned} \mathbf{Pr}[j \in A_J] = 1 - \prod_{r=1}^{\ell} \mathbf{Pr}[|S_j^{(r)} \cap J| \leq 1] &= 1 - \prod_{r=1}^{\ell} (1-p)^{t_r}(1+t_rp) \\ &= 1 - (1-p)^{K'} \prod_{r=1}^{\ell} (1+t_rp). \end{aligned}$$

We now bound this below by a constant. First consider the case $K' \leq \ell$. Then $\ell \geq K$ and $p \geq \frac{1}{2\sqrt{K}} \geq \frac{1}{2\sqrt{K'}}$. When $x_1, \ldots, x_\ell$ are constrained to be nonnegative integers with a fixed sum, if there exists $x_i \leq x_j - 2$,

we can strictly increase the product $f(x_1,\dots,x_\ell) \coloneqq \prod_{r=1}^{\ell}(1+x_r p)$ by replacing x_i with x_i+1 and x_j with x_j-1. Thus, the maximum value of $f(x_1,\dots,x_\ell)$ occurs when K' of the x_i are 1 and the rest are zero. Hence, we have

$$\mathbf{Pr}[j \in A_J] \geq 1-(1-p)^{K'}(1+p)^{K'} = 1-(1-p^2)^{K'} \geq 1-(1-\frac{1}{4K'})^{K'} \geq 1-e^{-1/4},$$

as desired.

Now suppose $K' > \ell$. Note that $p = \min\{\sqrt{\ell}/(2K), \frac{1}{2}\} \geq \sqrt{\ell}/(2K')$. As $\log(1+xp)$ is concave for nonnegative real numbers x, we have that $f(x_1,\dots,x_\ell) = \prod_{r=1}^{\ell}(1+x_r p)$ subject to $x_1+\cdots+x_\ell = K'$ is maximized when all the x_i's are equal. Hence,

$$\begin{aligned}\mathbf{Pr}[j \in A_J] = 1-(1-p)^{K'}\prod_{r=1}^{\ell}(1+t_r p) &\geq 1-(1-p)^{K'}\left(1+\frac{K'}{\ell}p\right)^{\ell} \\ &= 1-\left(\left((1-p)^{K'/\ell}\right)\left(1+\frac{K'}{\ell}p\right)\right)^{\ell}\end{aligned}$$

as desired. If $x \coloneqq K'/\ell \leq 1/p$, then, by Lemma 5.21, we have

$$\mathbf{Pr}[j \in A_J] \geq 1-\left(1-p^2\frac{(K')^2}{8\ell^2}\right)^{\ell} \geq 1-\left(1-\frac{1}{32\ell}\right)^{\ell} \geq 1-e^{-1/32}$$

where the second inequality uses the fact $p \geq \sqrt{\ell}/(2K')$. If $x \geq 1/p$, then by Lemma 5.21, we have $\mathbf{Pr}[j \in A_J] \geq 1-(2/e)^{\ell} \geq 1-2/e$. In all cases, $\mathbf{Pr}[j \in A_J]$ is bounded below by a constant, as desired. □

Corollary 5.23. *There exists $J \subseteq [t]$ of cardinality at most $c_1\sqrt{\ell}t/K$ such that the cardinality of the set A_J as defined in Lemma 5.22 is at least $c_2|A|$, where $c_1, c_2 > 0$ are absolute constants.*

Proof. Choose a random set $J \subseteq [t]$ as in Lemma 5.22. Then $\mathrm{ex}[|A_J|] = \Omega(|A|)$ by Lemma 5.22. As $|A_J| \leq |A|$, we have $\mathbf{Pr}[|A_J| \geq c_2|A|] \geq c_3$ for some absolute constants $c_2, c_3 > 0$.

Observe that by Lemma 5.22 and the linearity of expectation we have $\mathrm{ex}[|J|] = pt = O(\sqrt{\ell}t/K)$. Moreover, by Markov's inequality we have $\mathbf{Pr}[|J| > c_1\sqrt{\ell}t/K] \leq c_3/2$ for some sufficiently large constant $c_1 > 0$. By the union bound, we know the conditions $|J| \leq c_1\sqrt{\ell}t/K$ and $|A_J| \geq c_2|A|$ are simultaneously satisfied with probability at least $c_3/2 > 0$, so there exists $J \subseteq [t]$ that satisfies these two conditions. □

Fix $J \subseteq [t]$ as in Corollary 5.23, so that $|J| \leq c_1\sqrt{\ell}t/K$ and $|A_J| \geq c_2|A|$. As the constant c in (5.5) is large enough, we may assume $c \geq c_0 c_1/c_2$, where c_0 is as in (5.10). Then we have

$$|A_J| \geq c_2|A| \overset{(5.10)}{\geq} c_2 \cdot \frac{\mathrm{wt}_\ell(I_{[t]})}{c_0 K(\log(\frac{1}{\epsilon})+\log(\frac{1+\delta}{\delta})+1)} \overset{(5.5)}{\geq} (c_1\sqrt{\ell}t/K)k > (|J|-1)k. \tag{5.12}$$

For each $j \in A_J$, choose a subset $T_j \subseteq S_j \cap J$ and an index $r_j \in [\ell]$ such that $|T_j| = 2$ and $T_j \subseteq S_j^{(r_j)}$. This is possible by the definition of A_J in Lemma 5.22. For $j \in [n] \setminus A_J$, let $T_j = \emptyset$. So for $j \in [n]$, we have

$$|T_j| = \begin{cases} 2 & j \in A_J, \\ 0 & j \notin A_J. \end{cases}$$

Let $I'_i = \{j \in [n] : i \in T_j\} \subseteq I_i$ for $i \in J$. We have

$$\mathrm{wt}(I'_J) = \sum_{j=1}^{n} \max\{|T_j| - 1, 0\} = |A_J| \overset{(5.12)}{\geq} (|J| - 1)k.$$

Moreover, the fact $|T_j| \leq 2$ for $j \in [n]$ implies that $I'_i \cap I'_{i'} \cap I'_{i''} = \emptyset$ for distinct $i, i', i'' \in J$, by noting that $T_j = \{i \in J : j \in I'_i\}$.

For $j \in A_J$, we have

$$\{i \in J : j \in I'_i\} = T_j \subseteq S_j^{(r_j)} \cap J = \{i \in J : j \in I_i^{(r_j)}\}.$$

And for $j \in [n] \setminus A_J$, we have

$$\{i \in J : j \in I'_i\} = T_j = \emptyset \subseteq \{i \in J : j \in I_i^{(r)}\} \text{ for any } r \in [\ell].$$

Finally, we have $|J| \geq 2$ as $|A_J| \geq c_2|A| > 0$. To summarize, we have proved the following weaker version of Lemma 5.17.

Lemma 5.24. *Under the assumption of Lemma 5.17, there exist $J \subseteq [t]$ and a collection $(I'_i)_{i \in J}$ of subsets of $[n]$ such that $|J| \geq 2$, $I'_i \subseteq I_i$ for $i \in J$, and the following conditions are satisfied:*

(1) $I'_i \cap I'_{i'} \cap I'_{i''} = \emptyset$ for distinct $i, i', i'' \in J$.

(2) $\mathrm{wt}(I'_J) \geq (|J| - 1)k$.

(3) For every $j \in [n]$, there exists $r_j \in [\ell]$ such that $\{i \in J : j \in I'_i\} \subseteq \{i \in J : j \in I_i^{(r_j)}\}$.

Now we are ready to prove Lemma 5.17.

Proof of Lemma 5.17. Choose the sets J and $(I'_i)_{i \in J}$ satisfying Lemma 5.24 such that $|J| \geq 2$ is minimized. Note that removing one element from I'_i for some $i \in J$ preserves (1) and (3) of Lemma 5.24 and reduces $\mathrm{wt}(I'_J)$ by at most one. Removing elements from the sets in $(I'_i)_{i \in J}$ one by one until $\mathrm{wt}(I'_J) = (|J| - 1)k$ holds. Then J and $(I'_i)_{i \in J}$ satisfy (1), (3), and (4) of Lemma 5.17.

The minimality of $|J|$ guarantees that $\mathrm{wt}(I'_{J'}) \leq (|J'| - 1)k$ for all nonempty $J' \subseteq J$. (When $|J'| = 1$, this holds since $\mathrm{wt}(I'_{J'}) = 0$.) So J and $(I'_i)_{i \in J}$ satisfy (2) of Lemma 5.17 as well. □

5.5 Towards Conjecture 5.3: A Hypergraph Nash-Williams–Tutte Conjecture

Recall that Conjecture 5.3 states that RS codes of rate R are list-decodable from radius $1 - R - \varepsilon$ with list size at most $\lceil \frac{1-R-\varepsilon}{\varepsilon} \rceil$. As discussed in the introduction, it was shown in [113, Theorem 5.8] that resolving Conjecture 5.4 (about the non-singularity of intersection matrices) would resolve Conjecture 5.3 (about list-decoding).

Our approach above was to show in Theorem 5.9 that intersection matrices are nonsingular under the additional assumption that $I_i \cap I_j \cap I_\ell = \emptyset$ for all $1 \leq i < j < \ell \leq t$, and then use that to conclude our main result about list-recovery. However, to prove Conjecture 5.3 in full, we need to remove the additional

three-wise intersection assumption. In this section, we describe an approach that could potentially resolve Conjecture 5.3 in full. To do so, we conjecture a hypergraph generalization of the Nash-Williams–Tutte theorem [102, 121] that may be of independent interest, and prove that this conjecture implies Conjecture 5.3. Indeed, one might hope that such a generalization would be useful for Conjecture 5.3, because the Nash-Williams–Tutte theorem has been instrumental in proving several of the results in this chapter, including Theorem 5.1, Theorem 5.2, Theorem 5.9. Along the way, we give a second proof of our main list decoding theorem, Theorem 5.1, by combining this approach with known results on hypergraph packings [25, 18] In Section 5.5.1, we describe our conjecture, Conjecture 5.25, and in Section 5.5.2, we prove that it implies the optimal list decoding conjecture, Conjecture 5.3.

5.5.1 A Hypergraph Nash-Williams–Tutte Conjecture

In this section we state a conjecture (Conjecture 5.25 below), and prove that it implies our main goal Conjecture 5.3. We are not able to prove Conjecture 5.25, but give some evidence for it, pointing out that special cases and relaxations are known to be true.

Throughout, we use t as the number of vertices in a (hyper)graph. This variable corresponds to the same t used in t-wise intersection matrices. A (multi)graph G is called *k-partition-connected* if every partition $\mathcal{P}$ of the vertex set has at least $k(|\mathcal{P}|-1)$ edges crossing the partition. By the Nash-Williams–Tutte theorem, this is equivalent to the graph having k edge-disjoint spanning trees. The parameter k here is the same k used as the dimension of the Reed–Solomon code and the same k used for the Vandermonde matrix degrees in the intersection matrices.

We say a hypergraph H is *k-weakly-partition-connected*[6] if, for every partition $\mathcal{P}$ of the vertices of H, we have

$$\sum_{e \in E(H)} (\mathcal{P}(e) - 1) \geq k(|\mathcal{P}| - 1), \tag{5.13}$$

where $\mathcal{P}(e)$ is the number of parts of $\mathcal{P}$ that e intersects. For example, any k-partition-connected graph is k-weakly-partition-connected as a hypergraph. As another example, k copies of a hyperedge covering all t vertices of H is also k-weakly partition-connected.

An *edge-labeled* graph is a graph G where each edge is assigned a label from some set E. Let H be a hypergraph. A *tree-assignment* of H is an edge-labeled graph G obtained by replacing each edge e of H with a tree F_e of $|e|-1$ edges on the vertices of e. Furthermore, each edge of the graph F_e is labeled with e. The graph G is thus the union of the graphs F_e for $e \in H$.

A *k-tree-decomposition* of a graph on $k(t-1)$ edges is a partition of its edges into k edge-disjoint spanning trees $T_0, \ldots, T_{k-1}$. We say *tree-decomposition* when k is understood. In an edge-labeled graph T with edge-labels from some set E, let $v^T \in \mathbb{N}^E$ be the vector counting the edge-labels in T. Specifically, v^T_e is the number of edges of label e in T. For a tree-decomposition $(T_0, \cdots, T_{k-1})$ of an edge-labeled graph, define

[6] There is also a notion of "k-partition-connected" for hypergraphs which uses $\min\{\mathcal{P}(e)-1, 1\}$ in the sum. In other words, a hypergraph is k-partition-connected if any partition $\mathcal{P}$ has at least $k(|\mathcal{P}|-1)$ crossing edges. This notion admits a Nash-Williams–Tutte type theorem: any k-partition-connected hypergraph can be decomposed into k 1-partition-connected hypergraphs [37]

its *signature* $v^{(T_0,\dots,T_{k-1})}$ by

$$v^{(T_0,\dots,T_{k-1})} := \sum_{i=0}^{k-1} i \cdot v^{T_i}. \tag{5.14}$$

An edge-labeled graph G on t vertices is called *k-distinguishable* if G has $k(t-1)$ edges and there exists a tree-decomposition $T_0,\dots,T_{k-1}$ of G with a unique signature. That is, for any tree-decomposition $T'_0,\dots,T'_{k-1}$ with the same signature $v^{(T'_0,\dots,T'_{k-1})} = v^{(T_0,\dots,T_{k-1})}$, we have $T'_i = T_i$ for $i = 0,\dots,k-1$.

With these definitions, we can now conjecture a hypergraph version of the Nash-Williams–Tutte theorem.

Conjecture 5.25. *Let t and k be positive integers. Every k-weakly-partition-connected hypergraph H on t vertices has a k-distinguishable tree-assignment.*

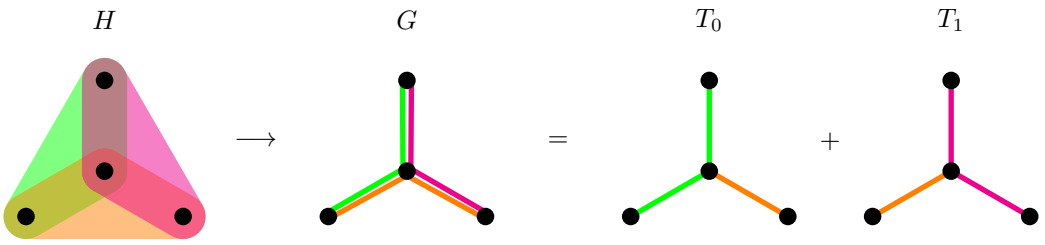
The key result of this section is that our optimal list decoding of Reed–Solomon codes conjecture follows from our hypergraph Nash-Williams–Tutte conjecture. This connection is proved in Section 5.5.2 below.

Theorem 5.26. *Conjecture 5.25 implies Conjecture 5.4 and thus Conjecture 5.3.*

One should convince themselves that Conjecture 5.25 (if true) is a generalization of the Nash-Williams–Tutte theorem. Indeed, when H is a (non-hyper) graph, the conjecture boils down to the Nash-Williams–Tutte theorem. If H is a graph on $k(t-1)$ edges, then there is only one tree-assignment G of H, namely H itself with each edge labeled by itself. All edges have distinct edge-labels, so for any tree-decomposition $T_0,\dots,T_{k-1}$, the signature $v^{(T_0,\dots,T_{k-1})}$ is unique. Thus, showing G is distinguishable, is equivalent to showing G has k edge-disjoint spanning trees, which follows from the Nash-Williams–Tutte theorem. In the correspondence between hypergraph partitions and intersection matrices, the special case when H is a graph corresponds to Theorem 5.9.

To give more intuition when H is not a graph, we give the following example.

Example 5.27. Let $t = 4$, and $k = 2$. Below, H is a 2-weakly-partition-connected hypergraph. We take a tree assignment of H to obtain an edge-labeled graph G on 6 edges, where each edge is labeled by its color. The tree-decomposition $T_0 \cup T_1$ demonstrates that G is 2-distinguishable: We have $v^{T_0} = (2,1,0)$ and $v^{T_1} = (0,1,2)$ so the signature is $v^{(T_0,T_1)} = 0 \cdot v^{T_0} + 1 \cdot v^{T_1} = (0,1,2)$. One can check that any other tree decomposition (T'_0,T'_1) of G has a different signature $v^{(T'_0,T'_1)} \neq (0,1,2)$. Thus, G is 2-distinguishable. Hence, H is a 2-weakly partition-connected hypergraph with a 2-distinguishable tree-assignment, as predicted by Conjecture 5.25.



As evidence towards Conjecture 5.25, we point out that the "easy part" of the conjecture follows from the Nash-Williams–Tutte theorem. One can check that, even in general hypergraphs, every tree-assignment of a k-weakly-partition-connected hypergraph gives a k-partition connected graph. Thus, any tree-assignment

graph can be partitioned into k spanning trees by the Nash-Williams–Tutte theorem, establishing the "existence" part of the graph G being k-distinguishable. Thus, the hard part of Conjecture 5.25 is the "uniqueness" part, finding a spanning tree partition with a unique signature.

As further evidence towards Conjecture 5.25, we point out that a relaxation of Conjecture 5.25 is true by assuming a larger weak-partition-connectivity [25, 18].

Theorem 5.28 (Follows from [25, 18])**.** *There exists an absolute constant C such that the following holds. Let t and k be positive integers. Every $C(\log t)k$-weakly-partition-connected hypergraph H on t vertices has a tree-assignment with a k-distinguishable subgraph.*

The results in [25, 18] look slightly different from Theorem 5.28 and we briefly describe the connection here. Both of the works prove that every $C(\log t)k$-weakly partition-connected hypergraph has k hyperedge-disjoint connected subhypergraphs:[7] the first [25] proves an equivalent statement about bipartite Steiner tree packings, and the second [18] proves a more general result about disjoint bases of polymatroids and also improves the constant C in front of the log factor over [25]. Furthermore, it is not difficult to show that, for any hypergraph H with k hyperedge-disjoint connected subhypergraphs $H_0, \ldots, H_{k-1}$, any tree assignment of H has a k-distinguishable subgraph: briefly, the k-distinguishable subgraph G is the union of spanning trees $T_0, \ldots, T_{k-1}$ of the tree assignments of $H_0, \ldots, H_{k-1}$ that are implied by the tree assignments of H, and the spanning trees $T_0, \ldots, T_{k-1}$ give the k edge-disjoint spanning trees of G that certify k-distinguishability. Hence, any $C(\log t)k$-weakly-partition-connected hypergraph has k hyperedge-disjoint connected subhypergraphs and thus is k-distinguishable, which is Theorem 5.28.

While it is known that the log factor in the results of [25, 18] cannot be removed [6][8], one can still hope that Conjecture 5.25 is true. The results in [25, 18] are used to obtain k-distinguishability by taking a tree packing where edges of the same label are always in the same tree. Keeping the same edge labels in the same trees is not necessary in general to obtain distinguishability. Indeed Example 5.27 illustrates that a tree-assignment can be k-distinguishable even when the original hypergraph does not have a partition into k connected subhypergraphs. Thus, we hope that the log factor can still be saved by splitting edge labels of the tree-assignment graph across the k spanning trees. As we pointed out earlier, the existence of a tree-packing in the tree-assignment graph follows from the Nash-Williams–Tutte theorem, so the hard part of the conjecture is not finding the tree-packing, but finding a signature-unique one.

We point out that Theorem 5.28 can be used to obtain a second proof of Theorem 5.1 by following the proof of Theorem 5.26, which connects hypergraph partitions to intersection matrices and then applies a polynomial method to obtain list decodable Reed–Solomon codes. The extra $O(\log t)$ factor in the weak-partition-connectivity of Theorem 5.28 appears as an $O(\log \frac{1}{\varepsilon})$ factor loss in the rate of the Reed–Solomon code. Since one proof of Theorem 5.1 is already given, and the key ideas for this second proof are covered throughout the rest of the chapter, we omit the details of this second proof of Theorem 5.1.

In addition to implying optimal list decoding of Reed Solomon codes (Theorem 5.28), Conjecture 5.25 may be of independent interest as a candidate hypergraph Nash-Williams–Tutte generalization. On one hand, a hypergraph Nash-Williams–Tutte generalization is known for *partition-connectivity* [37]. A hypergraph is k-partition-connected if any partition $\mathcal{P}$ has at least $k(|\mathcal{P}| - 1)$ crossing edges. Frank, Király, and Kriesell

[7] A hypergraph is connected if, for every two vertices v and v', there is a path $v = v_0, v_1, \ldots, v_\ell = v'$ such that, for all i, v_{i-1}, v_i share a hyperedge.

[8] [6] shows there exist $\Omega(\log n)$-weakly-partition-connected hypergraphs without two edge-disjoint connected subhypergraphs.

[37] showed that every k-partition-connected hypergraph has k edge-disjoint 1-partition-connected subhypergraphs. On the other hand, Nash-Williams–Tutte generalizations for *weak-partition-connectivity* seem to be less studied, and Conjecture 5.25 provides a plausible generalization for weak-partition-connectivity.

5.5.2 Proof of Theorem 5.26

In this section we prove Theorem 5.26. To do so, we show the first implication, establishing the connection between hypergraph partitions and intersection matrices outlined in Section 5.5.1. The second implication was proved in [113]. At a high level, the first implication of Theorem 5.26 holds because the uniqueness of the signature of k edge-disjoint spanning trees implies the uniqueness of a monomial in the determinant expansion of an intersection matrix. Because such a monomial is unique, it does not cancel with any other terms in the determinant expansion, implying that the determinant is nonzero. We now give the details.

We first derive a sufficient condition for a hypergraph being k-weakly-partition-connected.

Lemma 5.29. *Let H be hypergraph on the vertex set $[t]$ where for all $J \subsetneq [t]$,*

$$\sum_{e \in E(H)} \max(0, |e \cap J| - 1) \;\leq\; k(|J| - 1)$$
$$\text{and} \quad \sum_{e \in E(H)} (|e| - 1) \;\geq\; k(t-1). \tag{5.15}$$

Then H is k-weakly-partition-connected.

Proof. According to (5.13) it suffices to show that for any partition $\mathcal{P}$ of the vertices of H, $\sum_{e \in E(H)} (\mathcal{P}(e) - 1) \geq k(|\mathcal{P}| - 1)$. To see this, assume that $\mathcal{P} = \{V_1, \ldots, V_s\}$. Then $\sum_{i=1}^{s} |V_i| = t$, and for each $e \in E(H)$, $|e| = \sum_{i=1}^{s} |e \cap V_i|$. By the last equality, it is not hard to check that

$$|e| = \mathcal{P}(e) + \sum_{i=1}^{s} \max\{0, |e \cap V_i| - 1\}.$$

It follows that

$$\begin{aligned}
\sum_{e \in E(H)} (\mathcal{P}(e) - 1) &= \sum_{e \in E(H)} \Big(|e| - \sum_{i=1}^{s} \max\{0, |e \cap V_i| - 1\} - 1 \Big) \\
&= \sum_{e \in E(H)} (|e| - 1) - \sum_{i=1}^{s} \sum_{e \in E(H)} \max\{0, |e \cap V_i| - 1\} \\
&\geq k(t-1) - \sum_{i=1}^{s} k(|V_i| - 1) = k(s-1),
\end{aligned}$$

where the last inequality follows from (5.15). □

We now present the proof of Theorem 5.26.

Proof of Theorem 5.26. Assume Conjecture 5.25 is true. Let $I_1, \ldots, I_t \subseteq [n]$ be subsets satisfying the conditions of Conjecture 5.4. For all $i \in [n]$, let $e_i = \{j \in [t] : i \in I_j\}$. Let H be a (multi)hypergraph with vertex set $[t]$ and edge set $E(H) = \{e_i : i \in [n]\}$.

Claim 5.30. *For all subsets $J \subseteq [t]$, we have $\sum_{e \in E(H)} \max(0, |e \cap J| - 1) = \mathrm{wt}(I_j : j \in J)$.*

Proof. We have

$$\sum_{i \in [n]} \max\{0, |e_i \cap J| - 1\} = \sum_{i \in [n]} |e_i \cap J| - \left| \bigcup_{j \in J} I_j \right| = \sum_{j \in J} |I_j| - \left| \bigcup_{j \in J} I_j \right| = \mathrm{wt}(I_J),$$

where the first equality follows from the fact $\cup_{j \in J} I_j = \{i \in [n] : |e_i \cap J| \geq 1\}$, the second equality follows from easy double-counting, and the last equality follows from (5.1). □

The following is an easy consequence of Lemmas 5.29 and Claim 5.30.

Claim 5.31. *H is k-weakly-partitioned-connected.*

Proof. By Claim 5.30 and the setup of Conjecture 5.4, it is clear that for each $J \subsetneq [t]$

$$\sum_{e \in E(H)} \max\{0, |e \cap J| - 1\} \;=\; \mathrm{wt}(I_J) \;\leq\; k(|J| - 1)$$
$$\text{and} \sum_{e \in E(H)} \max\{0, |e| - 1\} \;=\; \mathrm{wt}(I_{[t]}) \;\geq\; k(t - 1).$$

It follows by Lemma 5.29 that H is k-weakly-partition-connected. □

By our assumption that Conjecture 5.25 is true, there exists a tree-assignment G of H that is k-distinguishable. Note that by definition G has $k(t-1)$ edges, which are labeled by the hyperedges of H. Let $S \subseteq [n]$ be the subset so that $\{e_s : s \in S\}$ forms the set of those labels. Then, an edge $\{j, j'\}$ of G has label e_s for some $s \in S$ if and only if $s \in I_j \cap I_{j'}$.

Recall that

$$M_{k,(I_1,\ldots,I_t)} = \left(\frac{\mathcal{B}_t \otimes \mathcal{I}_k}{\mathrm{diag}\Big(V_k(I_j \cap I_{j'}) : \{j, j'\} \in \binom{[t]}{2}\Big)} \right),$$

and that the $\binom{t}{2}k$ columns are labeled by the pairs $\{j, j'\} \in \binom{[t]}{2}$, according to the $\binom{t}{2}$ Vandermonde matrices in the bottom diagonal. Our goal is to show that $M_{k,(I_1,\ldots,I_t)}$ is nonsingular. As in the proof of Theorem 5.9, it suffices to show the nonsingularity of

$$M'_{k,(I_1,\ldots,I_t)} \coloneqq \left(\frac{\mathcal{I}_k \otimes \mathcal{B}_t}{(\mathbb{C}_i : 0 \leq i \leq k-1)} \right),$$

where $\mathbb{C}_i = \mathrm{diag}\Big(V_k^{(i)}(I_j \cap I_{j'}) : \{j, j'\} \in \binom{[t]}{2}\Big)$ and $V_k^{(i)}(I_j \cap I_{j'})$ is the $(i+1)$-th column of $V_k(I_j \cap I_{j'})$. Note that $M'_{k,(I_1,\ldots,I_t)}$ is obtained by permuting the columns of $M_{k,(I_1,\ldots,I_t)}$, with the column labels remaining unchanged.

The following fact is easy to verify by definition.

Fact 5.32. *Each row of* $\big(\mathbb{C}_i : 0 \le i \le k-1\big)$ *has exactly* k *nonzero entries, which has the form* $x_s^0, x_s^1, \ldots, x_s^{k-1}$ *for some* $s \in S$*. Moreover, there is* $\{j, j'\} \in \binom{[t]}{2}$ *so that* $s \in I_j \cap I_{j'}$*, and those* k *nonzero entries are all contained in* $\{j, j'\}$*-labeled columns.*

Let us consider $\binom{t}{2}k \times \binom{t}{2}k$ submatrix M' of $M'_{k,(I_1,\ldots,I_t)}$ obtained as follows:

1. Keep the top $\binom{t-1}{2}k \times \binom{t}{2}k$ submatrix $\mathcal{I}_k \otimes B_t$.
2. For every edge $\{j, j'\}$ in G of label e_s, keep the row in $\big(\mathbb{C}_i : 0 \le i \le k-1\big)$ with nonzero entries $x_s^0, \ldots, x_s^{k-1}$ in $\{j, j'\}$-labeled columns (this is well-defined according to Fact 5.32).
3. Remove all other rows.

As G has $k(t-1)$ edges, precisely $k(t-1)$ rows are kept in step 2. Therefore, M' has $\binom{t-1}{2}k + k(t-1) = \binom{t}{2}k$ rows and is thus square.

Below we show that M' has a nonzero determinant, thereby implying that $M'_{k,(I_1,\ldots,I_t)}$ and hence $M_{k,(I_1,\ldots,I_t)}$ are nonsingular, which is the claim of Conjecture 5.4, and thus establishing Theorem 5.26. For that purpose, it is enough to show that there is a monomial that appears as a nonvanishing term in the determinant expansion of M'. To find such a monomial, we use the fact that G is k-distinguishable.

Recall that for a subgraph F of G, we use $v^F \in \mathbb{N}^S$ to denote the vector that counts the edge labels in F, where for $s \in S$, v_s^F is the number of edges with label e_s. Note that v^F is a vector of length $|S|$ whose coordinates are indexed by elements in S. For spanning trees $T, T_0, \ldots, T_{k-1}$ of G, define

$$x^T := \prod_{s \in S} x_s^{v_s^T} \qquad \text{and} \qquad x^{(T_0,\ldots,T_{k-1})} := \prod_{i=0}^{k-1} (x^{T_i})^i. \tag{5.16}$$

Observe that, by the definition in (5.14), we have $v_s^{(T_0,\ldots,T_{k-1})} = \sum_{i=0}^{k-1} i \cdot v_s^{T_i}$ for all $s \in S$. Hence, it follows from (5.16) that

$$x^{(T_0,\ldots,T_{k-1})} = \prod_{s \in S} x_s^{v_s^{(T_0,\ldots,T_{k-1})}}. \tag{5.17}$$

Since G is k-distinguishable, there exists a tree decomposition $T_0, \ldots, T_{k-1}$ such that for any other tree-assignment $T'_0, \ldots, T'_{k-1}$, we have that the signatures $v^{(T_0,\ldots,T_{k-1})} \neq v^{(T'_0,\ldots,T'_{k-1})}$. Thus, it follows by (5.17) that the monomials $x^{(T_0,\ldots,T_{k-1})} \neq x^{(T'_0,\ldots,T'_{k-1})}$.

Claim 5.33. *Let* $T_0, \ldots, T_{k-1}$ *be spanning trees as defined above. Then,* $x^{(T_0,\ldots,T_{k-1})}$ *appears as a nonvanishing term in the determinant expansion of* M'*.*

Proving Claim 5.33 establishes the nonsingularity of M' and thus, as discussed above, Theorem 5.26. For that purpose, we identify the nonzero entries in the bottom $(t-1)k$ rows of M' by tuples $(s, \{j, j'\}, i)$, where $s \in S \cap (I_j \cap I_{j'})$, $\{j, j'\} \in \binom{[t]}{2}$, and $0 \le i \le k-1$. Indeed, such a tuple corresponds to the entry x_s^i in the $\{j, j'\}$-labeled column of $\mathcal{C}_i$. It is worth mentioning that we used two types of labeling here: first, each column of $\mathcal{I}_k \otimes \mathcal{B}_t$ and $(\mathcal{C}_i : \ 0 \le i \le k-1)$ is labeled by some edge $\{j, j'\} \in \binom{[t]}{2}$; second, each edge of G is labeled by some variable x_s, $s \in S$.

Let U denote the set of all $(t-1)k^2$ nonzero entries in the bottom $(t-1)k$ rows of M'. For $Q \subset U$, let M_Q denote the submatrix of M' obtained by removing all of the rows and columns that contain some entry in Q. We say Q is a *partial transversal* if it contains exactly one element in each of the bottom $(t-1)k$ rows of M', and no two share a column. By the definition of determinant,

$$\det(M') = \sum_{Q \text{ partial transversal}} \pm \det(M_Q) \cdot \prod_{(s,\{j,j'\},i) \in Q} x_s^i. \tag{5.18}$$

For a partial transversal Q and $0 \le i \le k-1$, let Q_i be the subgraph of G that corresponds to the tuples $(s,\{j,j'\},i) \in Q$, namely,

$$Q_i = \left\{ \{j,j'\} \in \binom{[t]}{2} : \ (s,\{j,j'\},i) \in Q \text{ for some } s \in I_j \cap I_{j'} \right\}.$$

Note that we view each Q_i as a labeled subgraph that preserves the labeling of G. Moreover, as Q forms a partial transversal, each Q_i is a simple graph with no multiple edges, while G could be a multigraph.

We have the following claim.

Claim 5.34. *Let $Q \subseteq U$ be a partial transversal. Then, $\det(M_Q) \neq 0$ if and only if $Q_0, \ldots, Q_{k-1}$ form pairwise edge-disjoint spanning trees of G.*

Proof of Claim 5.34. For the "only if" part, note, that by the definition of a partial transversal, M_Q is a $\binom{t-1}{2}k \times \binom{t-1}{2}k$ square matrix with k diagonal blocks, where for each $0 \le i \le k-1$, the $(i+1)$-th diagonal block is obtained by removing from $\mathcal{B}_t$ all of the columns that are labeled by the edges of Q_i. As $\mathcal{B}_t$ has full row rank, $\det(M_Q) \neq 0$ if and only if each of the k diagonal blocks also has full row rank. By Claim 5.13, the $(i+1)$-th block has full row rank if and only if the labels of the removed columns form an acyclic graph on the vertices $[t]$, namely, Q_i is acyclic.

Since $k(t-1)$ columns are removed in total, and an acyclic subgraph on t vertices can have at most $t-1$ edges, $\det(M_Q) \neq 0$ can only happen if each Q_i is a spanning tree. Moreover, as elements in Q form a partial transversal, we never have $(s,\{j,j,'\},i)$ and $(s,\{j,j'\},i')$ both in Q, for $i \neq i'$. It follows that the Q_i's are also pairwise edge-disjoint, completing the proof of the "only if" part.

According to the discussions above, it is not hard to see that the "if" part follows fairly straightforwardly from Claim 5.13. Therefore, we omit its proof.

We are now in a position to present the proof of Claim 5.33.

Proof of Claim 5.33. With the notation above, it is not hard to check by definition that for a partial transversal $Q \subseteq U$ with $\det(M_Q) \neq 0$,

$$\prod_{(s,\{j,j'\},i) \in Q} x_s^i = \prod_{s \in S} \prod_{i=0}^{k-1} (x_s^{v_s^{Q_i}})^i = \prod_{s \in S} x_s^{v_s^{(Q_0,\ldots,Q_{k-1})}} = x^{(Q_0,\ldots,Q_{k-1})}. \tag{5.19}$$

It thus follows from (5.18), (5.19), and Claim 5.34 that

$$\det(M') = \sum_{Q_0,\ldots,Q_{k-1} \text{ edge-disj. spanning trees of } G} \pm \det(M_Q) \cdot x^{(Q_0,\ldots,Q_{k-1})}.$$

It is clear that by the definition of $T_0, \ldots, T_{k-1}$, in the above summation the monomial $x^{(T_0,\ldots,T_{k-1})}$ appears exactly once, and hence appears as a nonvanishing term in the determinant expansion, completing the proof of Claim 5.33 and thus the proof of Theorem 5.26. □

Acknowledgments

This chapter appeared as [43] and is joint work with Zeyu Guo, Chong Shangguan, Itzhak Tamo, and Mary Wootters. We thank Bruce Spang, Noga Ron-Zewi, and Karthik Chandrasekaran for helpful discussions, and we thank Karthik Chandrasekaran for the reference [18].

Chapter 6

Summary and Conclusions

6.1 Summary of contributions

In this thesis we gave improved combinatorial bounds for error-correcting codes in two basic settings: deletion errors and list-decoding.

For deletion codes, we gave the first non-trivial upper bound the zero-rate threshold for bit-deletions, showing that it is strictly less than $\frac{1}{2}$. We achieved this via a structural lemma that classifies the oscillation patterns of 0's and 1's in a balanced string, and then exploiting it to carefully orchestrate a noticeable advantage over just matching 1's by switching to matching 0's at judicious points.

For list-decoding over binary codes we gave improved analyses of the list-decodability of random linear binary codes. We gave a list-size upper bound that, in contrast to prior work, works for all values of the radius p, and also obtains improved bounds on the list size as the rate approaches list-decoding capacity. In particular, not only do our bounds improve on previous work for random linear codes, but they show that random linear codes are more list-decodable than completely random codes, in the sense that the list size is strictly smaller. Furthermore, we pinned down the output list size for list-decoding showing a tight lower bound that proves 3-point concentration of the list-size of random linear binary codes.

For list-decoding over large alphabets, we gave improved bounds for list-decoding the ubiquitous Reed–Solomon codes. We showed the existence of near-optimally list-decodable RS codes in the large-radius parameter regime. To do this, we established a connection between the intersection matrix approach of [113] and tree packings. Along the way, we also developed applications to the construction of strongly perfect hash matrices, and we have introduced a new hypergraph version of the Nash-Williams–Tutte theorem.

6.2 Future work

We now list some directions for future work below.

6.2.1 Deletion codes

Now that we finally have a resolution to the first order question of whether $p_{\text{del}}^{\text{thr}} = \frac{1}{2}$, it opens up the opportunity to address several related questions.

- An obvious and major challenge is to determine the exact value of the zero-rate threshold for bit-deletions which remains unknown. Our current state of knowledge is $\sqrt{2}-1 \leq p_{\mathrm{del}}^{\mathrm{thr}} \leq 1/2-\delta_0$, for $\delta_0 > 0$ given by Theorem 3.1. The true value of $\frac{1}{2}-p_{\mathrm{del}}^{\mathrm{thr}}$ is presumably much bigger than the minuscule δ_0 our proof yields. We made no attempt to optimize δ_0, but it is likely to be very small regardless. We hazard a guess that the true value might be closer to the lower bound. More audaciously, one might even postulate that $p_{\mathrm{del}}^{\mathrm{thr}} = \sqrt{2}-1$ since this threshold appears to be the limit of the techniques in the spirit of [14].

- As an interesting and necessary step toward improving the upper bound on $p_{\mathrm{del}}^{\mathrm{thr}}$, can one obtain better upper bounds on the *span* of a large enough code (which are implied by, but possibly easier to establish than, a lower bound on LCS)? Here, we define the *span* of two strings s and t as the minimum ratio $(|s'|+|t'|)/\operatorname{LCS}(s',t')$ over all pairs of (contiguous) substrings $s' \subseteq s$ and $t' \subseteq t$ which are of lengths $|s'| \geq \Omega(|s|)$, $|t'| \geq \Omega(|t|)$, and the span of a code is the minimum span between any two distinct strings in the code. Note that if $\mathcal{C}$ has span α, then the LCS of any two distinct strings in $\mathcal{C}$ is at most $2N/\alpha$. The codes of [14] are based on the construction of codes (a variant of the Bukh-Ma code) of growing size with span at least $2+\sqrt{2}$. Prior to our work, no non-trivial upper bound (bounded away from the trivial limit of 4) was known on the span of positive-rate codes.

  We point out that using our techniques, proving that there exist two codewords with span at most $4-\delta_0$ is easier than proving there exist two codewords with LCS at least $(1/2+\delta)N$. To show span at most $4-\delta_0$, matched flags do not have to be at similar locations in the two strings, so we have more flexibility with our random shifting argument. In particular, we can apply the structure lemma to entire codewords rather than dyadic substrings as we do in Lemma 3.28, so we do not need to combine LCS in prefixes/suffixes with Lemma 3.9. Furthermore, in the Blue-Yellow case, it is okay to use an imbalanced matching where say, we match Blue flags only in s, consuming more ones in t, and hence we do not need the string regularity and string reversal $\mathrm{rev}(\cdot)$ arguments which were used to ensure a balanced matching.

- Our quasi-polynomial in N upper bound on the size of codes $\mathcal{C} \subset \{0,1\}^N$ with $\operatorname{LCS}(\mathcal{C}) \leq (\frac{1}{2}+\delta_0)N$ can likely be improved with some more care in the argument, though we settled for it for sake of simplicity. We conjecture that in fact $|\mathcal{C}| \leq O(\log N)$. This would be tight, as the Bukh-Ma code has size $\Omega(\log N)$ and $\operatorname{LCS}(\mathcal{C}) \leq (\frac{1}{2}+\delta_0)N$. As evidence towards this conjecture, our techniques can be used to show that $|\mathcal{C}| \leq O(\log N)$ when any two codewords have span at most $4-\delta_0$, by following the sketch in the previous item.

- For the q-ary alphabet, the q-ary codes in [14] correct a fraction of deletions approaching $1-\frac{2}{q+\sqrt{q}}$, and thus we have

$$1-\tfrac{2}{q+\sqrt{q}} \leq p_{\mathrm{del}}^{\mathrm{thr}}(q) \leq 1-\tfrac{1+2\delta_0}{q} \ . \tag{6.1}$$

  An interesting question is to determine the infimum of constants c such that $p_{\mathrm{del}}^{\mathrm{thr}}(q) \geq 1-c/q$ for large enough q. The bounds in (6.1) imply that $c \in [1+2\delta_0, 2]$.

- In the list-decoding model, there are codes of rate $\mathrm{poly}(\varepsilon)$ which can be list-decoded from a fraction $1/2-\varepsilon$ of deletions with list-size $\mathrm{poly}(1/\varepsilon)$. Can one prove a lower bound $L(\varepsilon)$ on the list-size for list decoding from $(1/2-\varepsilon)$ fraction of deletions such that $L(\varepsilon) \to \infty$ as $\varepsilon \to 0$? What is the optimal

growth rate of the list-size as a function of ε? (For correcting bit-flips, the optimal list-size is known to be $\Theta(1/\varepsilon^2)$ [9, 69].)

- A *pair of twins* in a string s is a pair of two disjoint subsequences (recall that in our language *subsequences* of strings are not necessarily contiguous, unlike substrings) of s which are identical. A natural question to consider is: what is the length $t(N)$ of the longest pair of twins guaranteed to exist in any $s \in \{0,1\}^N$? This question is relevant for two reasons. First, the question of finding twins within a single string is closely related to the problem of finding longest common subsequences between distinct strings. Second, the twins problem was solved asymptotically by [5] using the regularity technique, which is also one of the ingredients in our work. It is now known that

$$\frac{N}{2} - O\Big(\frac{N(\log\log N)^{1/4}}{(\log N)^{1/4}}\Big) \leq t(N) \leq \frac{N}{2} - \Omega(\log N),$$

and it would be interesting to determine the exact growth rate of the lower-order term.

- Fundamentally, our main result is a result about finding common subsequences, which are ubiquitous objects. Can our techniques for finding common subsequences be applied to other kinds of subsequence questions?

6.2.2 List-decoding small alphabet codes

In Chapter 4, we settle the list-decodability of random linear binary codes. Our work raises several open questions.

- Our list-size upper bound shows that random linear binary codes of rate $1 - h(p) - \varepsilon$ are not (p, L)-list-decodable for $L \sim \frac{h(p)}{\varepsilon}$. Our techniques however do not generalize to q-ary alphabets for $q > 2$, and it would be interesting to prove a stronger upper bound.

  Our list-size lower bound generalizes to showing that random linear q-ary codes of rate $1 - h_q(p) - \varepsilon$ are not (p, L)-list-decodable for $L \sim \frac{h_q(p)}{\varepsilon}$. We conjecture that this lower bound is tight, i.e. that random linear codes of rate $1 - h_q(p) - \varepsilon$ are (p, L) list-decodable for $L = \frac{h_q(p)}{\varepsilon}(1 + o(1))$, where the $o(1) \to 0$ as $\varepsilon \to 0$. Our Theorem 4.4 shows it is true for $q = 2$, and we conjecture this is true for larger q.

- We have used different techniques for our upper and lower bounds. However, we think it is an interesting direction to use the characterization of [101]—which we used to prove our lower bounds—to prove upper bounds as well. This would entail showing that every sufficiently bad list is implicitly rare.

- Our results show that list-decoding and average-radius list-decoding have essentially the same output list sizes over binary alphabets, for random linear codes. It would be interesting to extend this to larger alphabets, or even to more general families of codes. This is especially interesting given that there is an exponential gap in the best known lower bounds (on the list-size for arbitrary codes) between list-decoding and average-radius list-decoding for general codes. [11, 61].

- Finally, and most ambitiously, there are currently no known explicit constructions of capacity-achieving binary list-decodable codes for general p. It is our hope that this thesis—which gives more information about the structure of linear codes which achieve list-decoding capacity—could lead to progress on this

front. Given that we don't know how to efficiently check if a given code is (p, L)-list-decodable, even an efficient Las Vegas construction (as opposed to a Monte Carlo construction) would be interesting. We remark that recently progress was made towards this goal: [101] showed that Gallager's ensemble of LDPC codes [38], which have even more structure than random-linear codes, achieve list-decoding capacity with high probability.

6.2.3 List-decoding Reed–Solomon codes

Lastly, we conclude with some open questions raised by our work in Chapter 5 on list-decoding Reed–Solomon codes.

- We remark that, using a simple idea from [113] one can convert each of the existence results of RS codes reported in this paper into an explicit code construction, although over a much larger field size. Is it possible to find explicit evaluation points over a smaller field size?

- Can we efficiently list-decode Reed–Solomon codes? In particular, given such an explicit Reed–Solomon code from the previous question, is it possible to decode it efficiently up to its guaranteed list-decoding radius? A similar question can be asked for list-recoverability. We note that [22], which shows that decoding RS codes much beyond the Johnson bound is likely hard in certain parameter regimes, does not apply to our parameter regime when the field size is large.

- Can we generalize the Nash-Williams–Tutte theorem to hypergraphs? In an attempt to resolve Conjecture 5.3, we present Conjecture 5.25, a new graph-theoretic conjecture, which can be viewed as a generalization of the Nash-Williams–Tutte theorem to hypergraphs. In addition to being interesting on its own, resolving this conjecture would imply the existence of optimally list-decodable RS codes.

Bibliography

[1] Dimitris Achlioptas and Assaf Naor. The two possible values of the chromatic number of a random graph. *Annals of Mathematics*, 162(3):1335–1351, 2005.

[2] Noga Alon and Michael Krivelevich. The concentration of the chromatic number of random graphs. *Combinatorica*, 17(3):303–313, 1997.

[3] Noga Alon and Joel Spencer. *The Probabilistic Method*. John Wiley, 1992.

[4] Sergey Avgustinovich, Julien Cassaigne, Juhani Karhumäki, Svetlana Puzynina, and Aleksi Saarela. On abelian saturated infinite words. *Theoretical Computer Science*, 792:154–160, 2019.

[5] Maria Axenovich, Yury Person, and Svetlana Puzynina. A regularity lemma and twins in words. *Journal of Combinatorial Theory, Series A*, 120(4):733–743, 2013.

[6] Jørgen Bang-Jensen and Stéphan Thomassé. Highly connected hypergraphs containing no two edge-disjoint spanning connected subhypergraphs. *Discrete Applied Mathematics*, 131(2):555–559, 2003.

[7] E. Ben-Sasson, S. Kopparty, and J. Radhakrishnan. Subspace polynomials and limits to list decoding of Reed-Solomon codes. *IEEE Trans. Inform. Theory*, 56(1):113–120, Jan 2010.

[8] Norman Biggs, Norman Linstead Biggs, and Biggs Norman. *Algebraic graph theory*, volume 67. Cambridge University Press, 1993.

[9] Vladimir M. Blinovsky. Bounds for codes in the case of list decoding of finite volume. *Problems of Information Transmission*, 22(1):7–19, 1986.

[10] Vladimir M Blinovsky. Code bounds for multiple packings over a nonbinary finite alphabet. *Problems of Information Transmission*, 41(1):23–32, 2005.

[11] Volodia M. Blinovsky. Bounds for codes in the case of list decoding of finite volume. *Problems of Information Transmission*, 22(1):719, 1986.

[12] Béla Bollobás and Paul Erdös. Cliques in random graphs. In *Mathematical Proceedings of the Cambridge Philosophical Society*, volume 80, pages 419–427. Cambridge University Press, 1976.

[13] Joshua Brakensiek, Venkatesan Guruswami, and Samuel Zbarsky. Efficient low-redundancy codes for correcting multiple deletions. *IEEE Trans. Inf. Theory*, 64(5):3403–3410, 2018.

[14] Boris Bukh, Venkatesan Guruswami, and Johan Håstad. An improved bound on the fraction of correctable deletions. *IEEE Trans. Inf. Theory*, 63(1):93–103, 2017.

[15] Boris Bukh and Jie Ma. Longest common subsequences in sets of words. *SIAM J. Discrete Math.*, 28(4):2042–2049, 2014.

[16] Boris Bukh and Lidong Zhou. Twins in words and long common subsequences in permutations. *Israel Journal of Mathematics*, 213(1):183–209, 2016.

[17] Jin-Yi Cai, Aduri Pavan, and D Sivakumar. On the hardness of permanent. In *Annual Symposium on Theoretical Aspects of Computer Science*, pages 90–99. Springer, 1999.

[18] Gruia Călinescu, Chandra Chekuri, and Jan Vondrák. Disjoint bases in a polymatroid. *Random Structures & Algorithms*, 35(4):418–430, 2009.

[19] Yeow Meng Chee, Ryan Gabrys, Alexander Vardy, Van Khu Vu, and Eitan Yaakobi. Reconstruction from deletions in racetrack memories. In *IEEE Information Theory Workshop, ITW 2018*, pages 1–5. IEEE, 2018.

[20] Yeow Meng Chee, Han Mao Kiah, Alexander Vardy, Van Khu Vu, and Eitan Yaakobi. Coding for racetrack memories. *IEEE Trans. Inf. Theory*, 64(11):7094–7112, 2018.

[21] Kuan Cheng, Zhengzhong Jin, Xin Li, and Ke Wu. Deterministic document exchange protocols, and almost optimal binary codes for edit errors. In *Proceedings of the 59th IEEE Annual Symposium on Foundations of Computer Science*, pages 200–211, 2018.

[22] Q. Cheng and D. Wan. On the list and bounded distance decodability of Reed-Solomon codes. *SIAM J. Comput.*, 37(1):195–209, April 2007.

[23] Mahdi Cheraghchi, Venkatesan Guruswami, and Ameya Velingker. Restricted isometry of fourier matrices and list decodability of random linear codes. In *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2013*, pages 432–442, 2013.

[24] Mahdi Cheraghchi and João L. Ribeiro. An overview of capacity results for synchronization channels. *IEEE Trans. Inf. Theory*, 67(6):3207–3232, 2021.

[25] Joseph Cheriyan and Mohammad R Salavatipour. Packing element-disjoint steiner trees. *ACM Transactions on Algorithms*, 3(4):47–es, 2007.

[26] Václáv Chvatal and David Sankoff. Longest common subsequences of two random sequences. *Journal of Applied Probability*, pages 306–315, 1975.

[27] Ph Delsarte. Bilinear forms over a finite field, with applications to coding theory. *Journal of Combinatorial Theory, Series A*, 25(3):226–241, 1978.

[28] Reinhard Diestel. *Graph theory*, volume 173 of *Graduate Texts in Mathematics*. Springer, Berlin, fifth edition, 2017.

[29] Suhas N. Diggavi and Matthias Grossglauser. On information transmission over a finite buffer channel. *IEEE Trans. Inf. Theory*, 52(3):1226–1237, 2006.

[30] Yang Ding. On list-decodability of random rank metric codes and subspace codes. *IEEE Trans. Information Theory*, 61(1):51–59, 2015.

[31] Dean Doron, Dana Moshkovitz, Justin Oh, and David Zuckerman. Nearly optimal pseudorandomness from hardness. Technical report, 2020.

[32] Dean Doron and Mary Wootters. High-probability list-recovery, and applications to heavy hitters. In *Electron. Colloquium Comput. Complex.*, volume 27, page 162, 2020.

[33] Zeev Dvir and Shachar Lovett. Subspace evasive sets. In *Proceedings of the forty-fourth annual ACM symposium on Theory of computing*, pages 351–358. ACM, 2012.

[34] Peter Elias. List decoding for noisy channels. *Wescon Convention Record, Part 2*, pages 94–104, 1957.

[35] Peter Elias. Error-correcting codes for list decoding. *IEEE Transactions on Information Theory*, 37(1):5–12, 1991.

[36] Asaf Ferber, Matthew Kwan, and Lisa Sauermann. List-decodability with large radius for reed-solomon codes. *IEEE Transactions on Information Theory*, 2022.

[37] András Frank, Tamás Király, and Matthias Kriesell. On decomposing a hypergraph into k connected sub-hypergraphs. *Discrete Applied Mathematics*, 131(2):373–383, 2003.

[38] Robert G. Gallager. Low-density parity-check codes. *IRE Trans. Information Theory*, 8(1):21–28, 1962.

[39] Anna C Gilbert, Hung Q Ngo, Ely Porat, Atri Rudra, and Martin J Strauss. l2/l2-foreach sparse recovery with low risk. In *International Colloquium on Automata, Languages, and Programming*, pages 461–472. Springer, 2013.

[40] Edgar N Gilbert. A comparison of signalling alphabets. *The Bell system technical journal*, 31(3):504–522, 1952.

[41] Eitan Goldberg, Chong Shangguan, and Itzhak Tamo. List-decoding and list-recovery of reed-solomon codes beyond the Johnson radius for any rate. *arXiv preprint arXiv:2105.14754*, 2021.

[42] SW Golomb, J Davey, I Reed, H Van Trees, and J Stiffler. Synchronization. *IEEE Transactions on Communications Systems*, 11(4):481–491, 1963.

[43] Zeyu Guo, Ray Li, Chong Shangguan, Itzhak Tamo, and Mary Wootters. Improved list-decodability and list-recoverability of reed-solomon codes via tree packings. In *62nd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2021*, pages 708–719. IEEE, 2021.

[44] V. Guruswami and A. Rudra. Limits to list decoding Reed–Solomon codes. *IEEE Trans. Inform. Theory*, 52(8):3642–3649, August 2006.

[45] V. Guruswami and C. Wang. Linear-algebraic list decoding for variants of Reed-Solomon codes. *IEEE Trans. Inform. Theory*, 59(6):3257–3268, June 2013.

[46] Venkatesan Guruswami. List decoding of binary codes-a brief survey of some recent results. In *Proceedings of Coding and Cryptology, Second International Workshop, IWCC 2009*, pages 97–106, 2009.

[47] Venkatesan Guruswami, Bernhard Haeupler, and Amirbehshad Shahrasbi. Optimally resilient codes for list-decoding from insertions and deletions. In *Proccedings of the 52nd Annual ACM Symposium on Theory of Computing*, pages 524–537, 2020.

[48] Venkatesan Guruswami, Johan Håstad, and Swastik Kopparty. On the list-decodability of random linear codes. *IEEE Trans. Information Theory*, 57(2):718–725, 2011.

[49] Venkatesan Guruswami, Johan Håstad, Madhu Sudan, and David Zuckerman. Combinatorial bounds for list decoding. *IEEE Trans. Information Theory*, 48(5):1021–1034, 2002.

[50] Venkatesan Guruswami, Xiaoyu He, and Ray Li. The zero-rate threshold for adversarial bit-deletions is less than 1/2. In *62nd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2021*, pages 727–738. IEEE, 2021.

[51] Venkatesan Guruswami and Piotr Indyk. Expander-based constructions of efficiently decodable codes. In *42nd Annual Symposium on Foundations of Computer Science, FOCS 2001, 14-17 October 2001, Las Vegas, Nevada, USA*, pages 658–667, 2001.

[52] Venkatesan Guruswami and Piotr Indyk. Near-optimal linear-time codes for unique decoding and new list-decodable codes over smaller alphabets. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, pages 812–821, 2002.

[53] Venkatesan Guruswami and Piotr Indyk. Linear time encodable and list decodable codes. In *Proceedings of the thirty-fifth annual ACM symposium on Theory of computing*, pages 126–135, 2003.

[54] Venkatesan Guruswami and Piotr Indyk. Efficiently decodable codes meeting gilbert-varshamov bound for low rates. In *SODA*, volume 4, pages 756–757. Citeseer, 2004.

[55] Venkatesan Guruswami and Swastik Kopparty. Explicit subspace designs. *Combinatorica*, 36(2):161–185, 2016.

[56] 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*, pages 620–624. IEEE, 2016.

[57] Venkatesan Guruswami and Ray Li. Coding against deletions in oblivious and online models. *IEEE Trans. Inf. Theory*, 66(4):2352–2374, 2020.

[58] Venkatesan Guruswami, Ray Li, Jonathan Mosheiff, Nicolas Resch, Shashwat Silas, and Mary Wootters. Bounds for list-decoding and list-recovery of random linear codes. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2020*, volume 176, pages 9:1–9:21, 2020.

[59] Venkatesan Guruswami, Ray Li, Jonathan Mosheiff, Nicolas Resch, Shashwat Silas, and Mary Wootters. Bounds for list-decoding and list-recovery of random linear codes. *IEEE Trans. Inf. Theory*, 68(2):923–939, 2022.

[60] Venkatesan Guruswami, Jonathan Mosheiff, Nicolas Resch, Shashwat Silas, and Mary Wootters. Sharp threshold rates for random codes. In *12th Innovations in Theoretical Computer Science Conference*, 2021.

[61] Venkatesan Guruswami and Srivatsan Narayanan. Combinatorial limitations of average-radius list-decoding. *IEEE Trans. Information Theory*, 60(10):5827–5842, 2014.

[62] Venkatesan Guruswami and Nicolas Resch. On the list-decodability of random linear rank-metric codes. pages 1505–1509, 2018.

[63] Venkatesan Guruswami and Atri Rudra. Concatenated codes can achieve list-decoding capacity. In Shang-Hua Teng, editor, *Proceedings of the 19th Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2008*, pages 258–267. SIAM, 2008.

[64] Venkatesan Guruswami and Atri Rudra. Explicit codes achieving list decoding capacity: Error-correction with optimal redundancy. *IEEE Transactions on Information Theory*, 54(1):135–150, 2008.

[65] Venkatesan Guruswami, Atri Rudra, and Madhu Sudan. Essential coding theory. *Draft available at* http://cse.buffalo.edu/faculty/atri/courses/coding-theory/book/, 2019.

[66] Venkatesan Guruswami and Madhu Sudan. Improved decoding of Reed–Solomon and algebraic-geometry codes. *IEEE Transactions on Information Theory*, 45(6):1757–1767, 1999.

[67] Venkatesan Guruswami and Madhu Sudan. Extensions to the johnson bound. *Manuscript, February*, 2001.

[68] Venkatesan Guruswami, Christopher Umans, and Salil Vadhan. Unbalanced expanders and randomness extractors from Parvaresh–Vardy codes. *Journal of the ACM*, 56(4):1–34, 2009.

[69] Venkatesan Guruswami and Salil P. Vadhan. A lower bound on list size for list decoding. *IEEE Trans. Inf. Theory*, 56(11):5681–5688, 2010.

[70] Venkatesan Guruswami and Carol Wang. Deletion codes in the high-noise and high-rate regimes. *IEEE Trans. Information Theory*, 63(4):1961–1970, 2017.

[71] Venkatesan Guruswami and Chaoping Xing. Folded codes from function field towers and improved optimal rate list decoding. In *Proceedings of the forty-fourth annual ACM symposium on Theory of computing*, pages 339–350. ACM, 2012.

[72] Venkatesan Guruswami and Chaoping Xing. List decoding Reed-Solomon, Algebraic-Geometric, and Gabidulin subcodes up to the Singleton bound. In *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*, pages 843–852. ACM, 2013.

[73] Bernhard Haeupler. Optimal document exchange and new codes for insertions and deletions. In *Proceedings of the 60th IEEE Annual Symposium on Foundations of Computer Science*, pages 334–347, 2019.

[74] Bernhard Haeupler and Amirbehshad Shahrasbi. Synchronization strings: explicit constructions, local decoding, and applications. In *STOC'18—Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, pages 841–854, 2018.

[75] Bernhard Haeupler and Amirbehshad Shahrasbi. Synchronization strings and codes for insertions and deletions - A survey. *IEEE Trans. Inf. Theory*, 67(6):3190–3206, 2021.

[76] Iftach Haitner, Yuval Ishai, Eran Omri, and Ronen Shaltiel. Parallel hashing via list recoverability. In *Annual Cryptology Conference*, pages 173–190. Springer, 2015.

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

[78] Hiep Hàn, Marcos Kiwi, and Matías Pavez-Signé. Quasi-random words and limits of word sequences. In *Latin American Symposium on Theoretical Informatics*, pages 491–503. Springer, 2021.

[79] Brett Hemenway, Noga Ron-Zewi, and Mary Wootters. Local list recovery of high-rate tensor codes & applications. In *58th Annual IEEE Symposium on Foundations of Computer Science*, 2017.

[80] Brett Hemenway and Mary Wootters. Linear-time list recovery of high-rate expander codes. In *International Colloquium on Automata, Languages, and Programming*, pages 701–712. Springer, 2015.

[81] Piotr Indyk, Hung Q Ngo, and Atri Rudra. Efficiently decodable non-adaptive group testing. In *Proceedings of the twenty-first annual ACM-SIAM symposium on Discrete Algorithms*, pages 1126–1142. SIAM, 2010.

[82] Selmer Johnson. A new upper bound for error-correcting codes. *IRE Transactions on Information Theory*, 8(3):203–207, 1962.

[83] Stasys Jukna. *Extremal combinatorics: with applications in computer science*. Texts in Theoretical Computer Science. An EATCS Series. Springer, Heidelberg, second edition, 2011.

[84] Ian A Kash, Michael Mitzenmacher, Justin Thaler, and Jonathan Ullman. On the zero-error capacity threshold for deletion channels. In *2011 Information Theory and Applications Workshop*, pages 1–5. IEEE, 2011.

[85] Marcos Kiwi, Martin Loebl, and Jiří Matoušek. Expected length of the longest common subsequence for large alphabets. *Advances in Mathematics*, 197:480–498, November 2004.

[86] S. Kopparty, N. Ron-Zewi, S. Saraf, and M. Wootters. Improved decoding of folded Reed-Solomon and multiplicity codes. In *2018 IEEE 59th Annual Symposium on Foundations of Computer Science*, pages 212–223. IEEE, 2018.

[87] Swastik Kopparty. List-decoding multiplicity codes. *Theory of Computing*, 11(1):149–182, 2015.

[88] Swastik Kopparty, Nicolas Resch, Noga Ron-Zewi, Shubhangi Saraf, and Shashwat Silas. On list recovery of high-rate tensor codes. *IEEE Transactions on Information Theory*, 67(1):296–316, 2020.

[89] Swastik Kopparty, Noga Ron-Zewi, Shubhangi Saraf, and Mary Wootters. Improved decoding of folded reed-solomon and multiplicity codes. In *2018 IEEE 59th Annual Symposium on Foundations of Computer Science*, pages 212–223. IEEE, 2018.

[90] Andreas Lenz, Paul H. Siegel, Antonia Wachter-Zeh, and Eitan Yaakobi. Coding over sets for DNA storage. *IEEE Trans. Inf. Theory*, 66(4):2331–2351, 2020.

[91] Vladmir I. Levenshtein. Binary codes capable of correcting deletions, insertions and reversals. *Soviet Physics Dokl. (English Translation)*, 10(8):707–710, 1966.

[92] Ray Li and Mary Wootters. Improved list-decodability of random linear binary codes. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2018)*, 2018.

[93] Ray Li and Mary Wootters. Improved list-decodability of random linear binary codes. *IEEE Trans. Inf. Theory*, 67(3):1522–1536, 2021.

[94] Tomasz Luczak. A note on the sharp concentration of the chromatic number of random graphs. *Combinatorica*, 11(3):295–297, 1991.

[95] George S Lueker. Improved bounds on the average length of longest common subsequences. *Journal of the ACM*, 56(3):1–38, 2009.

[96] Ben Lund and Aditya Potukuchi. On the list recoverability of randomly punctured codes. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX-/RANDOM 2020)*, volume 176, pages 30:1–30:11, 2020.

[97] Florence Jessie MacWilliams and Neil James Alexander Sloane. *The theory of error-correcting codes*. Elsevier, 1977.

[98] David W Matula. Employee party problem. In *Notices of the American Mathematical Society*, volume 19, pages A382–A382, 1972.

[99] Hugues Mercier, Vijay K. Bhargava, and Vahid Tarokh. A survey of error-correcting codes for channels with symbol synchronization errors. *IEEE Commun. Surv. Tutorials*, 12(1):87–96, 2010.

[100] Michael Mitzenmacher. A survey of results for deletion channels and related synchronization channels. *Probability Surveys*, 6:1–33, 2009.

[101] Jonathan Mosheiff, Nicolas Resch, Noga Ron-Zewi, Shashwat Silas, and Mary Wootters. LDPC codes achieve list decoding capacity. In *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020*, pages 458–469. IEEE, 2020.

[102] Crispin St. J. A. Nash-Williams. Edge-disjoint spanning trees of finite graphs. *Journal of the London Mathematical Society*, 1(1):445–450, 1961.

[103] Hung Q Ngo, Ely Porat, and Atri Rudra. Efficiently decodable error-correcting list disjunct matrices and applications. In *International Colloquium on Automata, Languages, and Programming*, pages 557–568. Springer, 2011.

[104] Morris Plotkin. Binary codes with specified minimum distance. *IRE Transactions on Information Theory*, 6(4):445–450, 1960.

[105] William H Press, John A Hawkins, Stephen K Jones, Jeffrey M Schaub, and Ilya J Finkelstein. HEDGES error-correcting code for dna storage corrects indels and allows sequence constraints. *Proceedings of the National Academy of Sciences*, 117(31):18489–18496, 2020.

[106] Irving S. Reed and Gustave Solomon. Polynomial codes over certain finite fields. *Journal of the Society for Industrial and Applied Mathematics*, 8(2):300–304, 1960.

[107] Oliver Riordan and Nicholas Wormald. The diameter of sparse random graphs. *Combinatorics, Probability and Computing*, 19(5-6):835–926, 2010.

[108] Atri Rudra and Mary Wootters. Every list-decodable code for high noise has abundant near-optimal rate puncturings. In *Proceedings of the Forty-Sixth annual ACM Symposium on Theory of Computing*, pages 764–773. ACM, 2014.

[109] Atri Rudra and Mary Wootters. Every list-decodable code for high noise has abundant near-optimal rate puncturings. In *Proceedings of the 46th Annual ACM Symposium on Theory of Computing*, STOC 2014, pages 764–773, 2014.

[110] Atri Rudra and Mary Wootters. It'll probably work out: Improved list-decoding through random operations. In *Proceedings of the 2015 Conference on Innovations in Theoretical Computer Science*, pages 287–296, 2015.

[111] Atri Rudra and Mary Wootters. Average-radius list-recovery of random linear codes. In *Proceedings of the 2018 ACM-SIAM Symposium on Discrete Algorithms, SODA*, 2018.

[112] Leonard J. Schulman and David Zuckerman. Asymptotically good codes correcting insertions, deletions, and transpositions. *IEEE Transactions on Information Theory*, 45(7):2552–2557, 1999.

[113] Chong Shangguan and Itzhak Tamo. Combinatorial list-decoding of Reed-Solomon codes beyond the Johnson radius. In *Proceedings of the 52nd Annual ACM Symposium on Theory of Computing*, STOC 2020, pages 538–551, 2020.

[114] Claude Elwood Shannon. A mathematical theory of communication. *The Bell system technical journal*, 27(3):379–423, 1948.

[115] Jin Sima and Jehoshua Bruck. On optimal k-deletion correcting codes. *IEEE Trans. Inf. Theory*, 67(6):3360–3375, 2021.

[116] Jin Sima, Ryan Gabrys, and Jehoshua Bruck. Syndrome compression for optimal redundancy codes. In *IEEE International Symposium on Information Theory*, pages 751–756, 2020.

[117] Richard Singleton. Maximum distance q-nary codes. *IEEE Transactions on Information Theory*, 10(2):116–118, 1964.

[118] Sundara Rajan Srinivasavaradhan, Sivakanth Gopi, Henry D. Pfister, and Sergey Yekhanin. Trellis BMA: coded trace reconstruction on IDS channels for DNA storage. In *IEEE International Symposium on Information Theory, ISIT 2021*, pages 2453–2458. IEEE, 2021.

[119] Madhu Sudan. List decoding: algorithms and applications. *SIGACT News*, 31(1):16–27, 2000.

[120] Madhu Sudan, Luca Trevisan, and Salil Vadhan. Pseudorandom generators without the xor lemma. *Journal of Computer and System Sciences*, 62(2):236–266, 2001.

[121] William T. Tutte. On the problem of decomposing a graph into n connected factors. *Journal of the London Mathematical Society*, 1(1):221–230, 1961.

[122] Jeffrey D. Ullman. On the capabilities of codes to correct synchronization errors. *IEEE Transactions on Information Theory*, 13(1):95–105, 1967.

[123] Salil P. Vadhan. Pseudorandomness. *Foundations and Trends in Theoretical Computer Science*, 7(1-3):1–336, 2012.

[124] Rom Rubenovich Varshamov. Estimate of the number of signals in error correcting codes. *Docklady Akad. Nauk, SSSR*, 117:739–741, 1957.

[125] Carol Wang. *Beyond unique decoding: topics in error-correcting codes*. PhD thesis, Carnegie Mellon University, 2015.

[126] Mary Wootters. On the list decodability of random linear codes with large error rates. In *Symposium on Theory of Computing Conference, STOC'13*, pages 853–860, 2013.

[127] Jack Wozencraft. List decoding. *Quarter Progress Report*, 48:90–95, 1958.

[128] Yihan Zhang, Amitalok J Budkuley, and Sidharth Jaggi. Generalized list decoding. In *2020 Information Theory and Applications Workshop*, pages 51–1. IEEE, 2020.

[129] Victor Vasilievich Zyablov and Mark Semenovich Pinsker. List concatenated decoding. *Problemy Peredachi Informatsii*, 17(4):29–33, 1981.