CODING FOR STORAGE AND TESTING

BY

SRILAKSHMI PATTABIRAMAN

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Electrical and Computer Engineering
in the Graduate College of the
University of Illinois Urbana-Champaign, 2021

Urbana, Illinois

Doctoral Committee:

Professor Olgica Milenkovic, Chair
Professor Iwan M. Duursma
Assistant Professor Ilan Shomorony
Associate Professor Lav R. Varshney
Professor Tandy Warnow

ABSTRACT

The problem of reconstructing strings from substring information has found many applications due to its importance in genomic data sequencing and DNA- and polymer-based data storage. Motivated by platforms that use chains of binary synthetic polymers as the recording media and read the content via tandem mass spectrometers, we propose new a family of codes that allows for both unique string reconstruction and correction of multiple mass errors.

We first consider the paradigm where the masses of substrings of the input string form the evidence set. We consider two approaches: The first approach pertains to asymmetric errors and the error-correction is achieved by introducing redundancy that scales linearly with the number of errors and logarithmically with the length of the string. The proposed construction allows for the string to be uniquely reconstructed based only on its erroneous substring composition multiset. The asymptotic code rate of the scheme is one, and decoding is accomplished via a simplified version of the Backtracking algorithm used for the Turnpike problem. For symmetric errors, we use a polynomial characterization of the mass information and adapt polynomial evaluation code constructions for this setting. In the process, we develop new efficient decoding algorithms for a constant number of composition errors.

The second part of this dissertation addresses a practical paradigm that requires reconstructing mixtures of strings based on the union of compositions of their prefixes and suffixes, generated by mass spectrometry devices. We describe new coding methods that allow for unique joint reconstruction of subsets of strings selected from a code and provide upper and lower bounds on the asymptotic rate of the underlying codebooks. Our code constructions combine properties of binary B_h and Dyck strings and can be extended to accommodate missing substrings in the pool.

In the final chapter of this dissertation, we focus on group testing. We

begin with a review of the gold-standard testing protocol for Covid-19, real-time, reverse transcription PCR, and its properties and associated measurement data such as amplification curves that can guide the development of appropriate and accurate adaptive group testing protocols. We then proceed to examine various off-the-shelf group testing methods for Covid-19, and identify their strengths and weaknesses for the application at hand. Finally, we present a collection of new analytical results for adaptive semiquantitative group testing with combinatorial priors, including performance bounds, algorithmic solutions, and noisy testing protocols. The worst-case paradigm extends and improves upon prior work on semiquantitative group testing with and without specialized PCR noise models.

To my family.

ACKNOWLEDGMENTS

I would like to express immense gratitude to my advisor Prof. Olgica Milenkovic for her guidance and support throughout my doctoral journey. I would also like to thank my doctoral committee members Prof. Iwan Duursma, Prof. Ilan Shomorony, Prof. Lav Varshney and Prof. Tandy Warnow for their feedback and guidance. I would also like to take this opportunity to thank all my collaborators - Dr. Ryan Gabrys, Dr. Abhishek Agarwal, Dr. João Ribeiro, Vishal Rana, Prof. Mahdi Cheraghchi, and Prof. Venkatesan Guruswami.

My time at the Coordinated Science Lab was wonderful primarily due to our faculty and my friends. My labmates have been a continuous source of support throughout these years. Thank you Pan, Jianhao, Eli, Chao, Roberto, João, Puoya, Vishal, Anu, Han, Mina and Rebecca! Thank you Surya, Vaishnavi, Ashok, Amish, Ben, Sourya, Saboo, Sid Satpathi, Sid, Jamila, Helmut, Anadi and Aditya for your friendship. My stay at Urbana has been lively and spirited thanks to Sneha, Yamuna, Chandana, Rucha, Souktik and Pranjal in addition to my friends listed above.

My love and regards to friends and family who have time and again hosted me at their homes. My day is brightened everytime I receive a text or call from a friend. Thank you Vaishu and Bhanu for being my cheer-squad. Thank you Aishwarya, Khurram, Anum, Megha and Arjun for always being a phone call away. Thank you Shoba aunty, Raman uncle, Akash and Shri for all your help and support over these years (and the amazing food). Thank you Ravi, Bande, Suzie, Gopal and Sudhu for your time, support and friendship. Here is to several more years!

This journey started with just a wish. To have it turn into a reality has been surreal. And it would not have been this way if not for my family. I would like to thank my grandparents for their love and support and my uncles, aunts and cousins for cheering me on. I would like to thank my sister-in-law for her love. Most of all, I would like to thank my parents

for their unwaivering strength, support and love, and my brother for always being there for me. I love you all immensely.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Current digital storage systems are facing numerous obstacles in terms of scaling the storage density and allowing for in-memory based computations [1]. To offer storage densities at nanoscale, several molecular storage paradigms have recently been put forward in [2–6]. One promising line of work with low storage cost and readout latency is [2], which proposes using synthetic polymers for storing user-defined information and reading the content via tandem mass spectrometry (MS/MS) techniques. More precisely, binary data is encoded using poly(phosphodiester)s, synthesized through automated phosphoramidite chemistry in such a way that the two bits 0 and 1 are represented by molecules of different masses that are stitched together into strings of fixed length. To read the encoded data, phosphate bonds are broken, and MS/MS readers are used to estimate the masses of the fragmented polymer and reconstruct the recorded string, as illustrated in the simplified scheme shown in Figure 1.1. Ideally, the masses of all prefixes and suffixes are recovered reliably, allowing one to read the message content by taking the differences of the increasing fragment masses and mapping them to the masses of the 0 or 1 symbol. Polymer synthesis is cost- and time-efficient and MS/MS sequencers are significantly faster than those designed for other macromolecules, such as DNA. Nevertheless, despite the fact that the masses of the polymers can be tuned to allow for more accurate mass discrimination, polymer-based storage systems still suffer from large read error-rates. This is due to the fact that MS/MS sequencing methods tend to produce peaks, representing the masses of the fragments that are buried in analogue noise due to atom disassociation during the fragmentation process and other sources of errors.

In an earlier line of work, the authors of [7] introduced the problem of *binary string reconstruction from its substring composition multiset* to address the issue of MS/MS readout analysis. The substring composition multiset of a binary string is obtained by writing out substrings of the string of all pos-

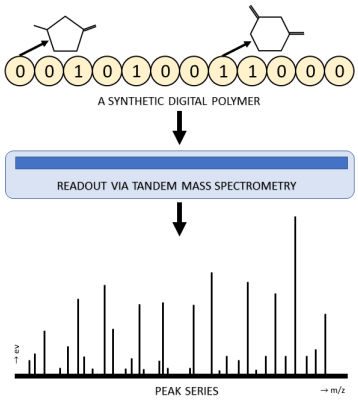


Figure 1.1: The scheme is adapted from [2]. The top figure depicts a binary string synthesized using phosphoramidite chemistry. The bottom image is an illustration of *peak series* or MS Spectrum obtained by MS/MS readout of the digital polymer. The peak series plots the charge at the detection plates (in eV) against the ratio of the mass number of the ion and its charge number (m/z). The charge normalization is often removed through calibration thereby allowing one to deal with masses only. Under ideal conditions, the peaks are supposed to correspond to the masses of string fragments, or more precisely, masses of prefixes and suffixes of the string. Due to measurement errors, spurious peaks arise and one needs to apply specialized signal processing techniques to identify the correct peaks.

sible lengths and then representing each substring by its composition. As an example, the string 101 contains three substrings of length one - 1, 0, and 1, two substrings of length 2 - 10 and 01, and one substring of length three - 101. The composition multiset of the substrings of length one equals $\{0, 1, 1\}$, the composition multiset of substrings of length two equals $\{0^1 1^1, 0^1 1^1\}$ and the composition multiset of substrings of length three equals $\{0^1 1^2\}$. Note that composition multisets ignore information about the actual order of the bits in the substrings and may hence be seen as only capturing the information about the "mass" or "weight" of the unordered substrings. Furthermore, the multiset information cannot distinguish between a string and its reversal, as well as some other nontrivial interleaved string settings. The problem addressed in [7] was to determine for which string lengths one can guarantee unique reconstruction from an error-free composition multiset, up to string reversal. The main results of [7, Theorem 17, 18, 20] demonstrate that binary strings of length $\leqslant 7$, one less than a prime or one less than twice a prime are uniquely reconstructable up to reversal. The work in [7] relies on

the two modeling assumptions:

Assumption 1. One can infer the composition of a polymer substring from its mass.

Assumption 2. When a polymer is broken down for mass spectrometry analysis, we observe the masses of all its substrings with identical frequency.

The masses of all binary substrings of an encoded polymer may be abstracted by the composition multiset of a string, provided that Assumption 1 holds. Assumption 2 slightly deviates from practical ion series measurements insofar as the latter only provides information about the masses of the prefixes and suffixes, while the proposed modification allows one to observe the masses of all substrings, but without a priori knowledge of their order. Under these modeling assumptions, Chapter 2 describes efficient encoding techniques such that the resultant string can be reconstructed from its composition multiset irrespective of the input string length k. Furthermore, we also design codes that can handle constant number of composition errors.

In Chapter 3, we refine our modeling assumptions:

Assumption 1. One can infer the composition of a polymer substring from its mass.

Assumption 2. When a polymer is broken down for mass spectrometry analysis, we observe the masses of all its prefixes and suffixes with identical frequency.

For a single string reconstruction from its prefix-suffix composition multiset, it suffices to distinguish the prefix compositions from the suffix compositions. However, if multiple strings are read simultaneously and the masses of prefixes and suffixes of the same length are confusable, the problem becomes complicated. We aim to find which combinations of coded binary strings can be distinguished from each other based on the union of their prefix-suffix masses and for which code rates is it possible to perform unique multistring reconstruction. For a given constant h, we seek the largest code of binary strings of a fixed length such that any $h' \leqslant h$ of the codestrings can be reconstructed from the union of the prefix-suffix composition multiset of the h' individual input string.

Chapter 4 pivots from the storage application of coding and focuses on group testing. In as little as ten months since the first case reported in the Hubei province of China, Covid-19 had rapidly spread across all continents except Antarctica [8]. The disease has caused more deaths than Ebola, SARS,

and the seasonal flu combined (reaching $5,000,000$ mortalities in November 2021), disrupted the global economy to an extent not seen since the Great Depression and altered the lives of hundreds of millions of people across the globe [9].

Many analyses associated with the Covid-19 pandemic have established that widespread population testing is key to effectively containing outbreaks of this and other infectious diseases. In May 2020, the United States was able to test around $150,000$ people per day, while countries that had managed to keep the outbreak under control, such as Germany and South Korea, had performed millions of tests during the same stage of the spread of the disease. To address the need for sustainable high-frequency population testing, a number of countries and states proposed and implemented *group testing schemes* in which genetic samples from different individuals are pooled together in a manner that incorporates thresholded real time reverse transcription polymerase chain reaction (RT-PCR) fluorescence signals into the testing scheme.

A number of recent reports suggest using Dorfman's or other mostly off-the-shelf GT schemes for Covid-19 testing [10–19]. Most of the proposed schemes do not incorporate relevant biological priors or exploit the highly specific measurement and noise properties of the RT-PCR method in their testing schemes. Chapter 4 argues that this is a significant detriment, as in order to properly execute the effort and avoid dangerous failures, testing schemes should be guided both by mathematical considerations as well as social, clinical, and genomic side information.

1.1 Dissertation Overview

This dissertation is in two parts: Chapters 2 and 3 focus on coding for polymer-based data storage. The two chapters highlight efficient coding and decoding techniques that pertain to string recovery from the corresponding evidence set as given by the readout mechanism. Chapter 4 deviates from the storage application and focuses on group testing for Covid-19. Adaptive testing schemes that take into consideration RT-PCR test outcomes and system specific noise are designed.

Chapters 2-4 are meant to be comprehensive and can be read in a stand-

alone manner. To that end, the introductions of Chapter 2 and Chapter 3 have minor overlaps.

1.2 Bibliographical Note

The following is a chapter-wise list of the publications that include the work presented herein:

Chapter 2

- S. Pattabiraman, R. Gabrys and O. Milenkovic, "Reconstruction and Error-Correction Codes for Polymer-Based Data Storage," in the proceedings of *Information Theory Workshop (ITW) August 2019.*
- R. Gabrys, S. Pattabiraman and O. Milenkovic,"Mass Error-Correction Codes for Polymer-Based Data Storage," in the proceedings of the *IEEE International Symposium on Information Theory (ISIT), 2020.*
- S. Pattabiraman, R. Gabrys and O. Milenkovic, "Coding for Polymer-Based Data Storage," *journal submission under review.*

Chapter 3

- R. Gabrys, S. Pattabiraman and O. Milenkovic, "Reconstructing Mixtures of Coded Strings from Prefix and Suffix Compositions," in the proceedings of *the 2020 Information Theory Workshop (ITW) April 2021.*
- R. Gabrys, S. Pattabiraman and O. Milenkovic, "Reconstruction of Sets of Strings from Prefix/Suffix Compositions," *journal submission under review.*

Chapter 4

- R. Gabrys, S. Pattabiraman, V. Rana, J. Ribeiro, M. Cheraghchi, V. Guruswami, O. Milenkovic, "AC-DC: Amplification Curve Diagnostics for Covid-19 Group Testing," *journal submission under review.*

CHAPTER 2

SINGLE CODED STRING RECONSTRUCTION

2.1 Introduction

Current digital storage systems are facing numerous obstacles in terms of scaling the storage density and allowing for in-memory based computations [1]. To offer storage densities at nanoscale, several molecular storage paradigms have recently been put forward in [2–6]. One promising line of work with low storage cost and readout latency is the work in [2], which proposes using synthetic polymers for storing user-defined information and reading the content via tandem mass spectrometry (MS/MS) techniques. More precisely, binary data is encoded using poly(phosphodiester)s, synthesized through automated phosphoramidite chemistry in such a way that the two bits 0 and 1 are represented by molecules of different masses that are stitched together into strings of fixed length. To read the encoded data, phosphate bonds are broken, and MS/MS readers are used to estimate the masses of the fragmented polymer and reconstruct the recorded string, as illustrated in the simplified scheme shown in Figure 2.1. Ideally, the masses of all prefixes and suffixes are recovered reliably, allowing one to read the message content by taking the differences of the increasing fragment masses and mapping them to the masses of the 0 or 1 symbol. Polymer synthesis is cost- and time-efficient and MS/MS sequencers are significantly faster than those designed for other macromolecules, such as DNA. Nevertheless, despite the fact that the masses of the polymers can be tuned to allow for more accurate mass discrimination, polymer-based storage systems still suffer from large read error-rates. This is due to the fact that MS/MS sequencing methods tend to produce peaks, representing the masses of the fragments that are buried in analogue noise due to atom disassociation during the fragmentation process and other sources of errors.

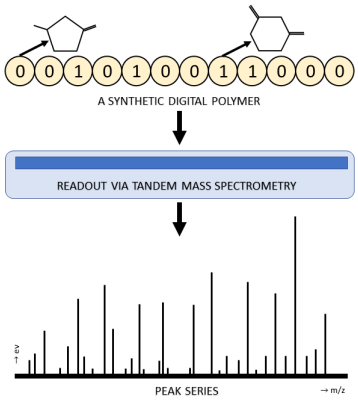


Figure 2.1: The scheme is adapted from [2]. The top figure depicts a binary string synthesized using phosphoramidite chemistry. The bottom image is an illustration of *peak series* or MS spectrum obtained by MS/MS readout of the digital polymer. The peak series plots the charge at the detection plates (in eV) against the ratio of the mass number of the ion and its charge number (m/z). The charge normalization is often removed through calibration thereby allowing one to deal with masses only. Under ideal conditions, the peaks are supposed to correspond to the masses of string fragments, or more precisely, masses of prefixes and suffixes of the string. Due to measurement errors, spurious peaks arise and one needs to apply specialized signal processing techniques to identify the correct peaks.

In an earlier line of work, the authors of [7] introduced the problem of *binary string reconstruction from its substring composition multiset* to address the issue of MS/MS readout analysis. The substring composition multiset of a binary string is obtained by writing out substrings of the string of all possible lengths and then representing each substring by its composition. As an example, the string 101 contains three substrings of length one - 1, 0, and 1, two substrings of length 2 - 10 and 01, and one substring of length three - 101. The composition multiset of the substrings of length one equals $\{0, 1, 1\}$, the composition multiset of substrings of length two equals $\{0^1 1^1, 0^1 1^1\}$ and the composition multiset of substrings of length three equals $\{0^1 1^2\}$. Note that composition multisets ignore information about the actual order of the bits in the substrings and may hence be seen as only capturing the information about the "mass" or "weight" of the unordered substrings. Furthermore, the multiset information cannot distinguish between a string and its reversal, as well as some other nontrivial interleaved string settings. The problem addressed in [7] was to determine for which string lengths one can guarantee

unique reconstruction from an error-free composition multiset, up to string reversal. The main results of [7, Theorem 17, 18, 20] demonstrate that binary strings of length $\leqslant 7$, one less than a prime or one less than twice a prime are uniquely reconstructable up to reversal.

For our line of work, we will rely on the two modeling assumptions first described in [7]:

Assumption 1. One can infer the composition of a polymer substring from its mass. *Assumption 2.* When a polymer is broken down for mass spectrometry analysis, we observe the masses of all its substrings with identical frequency.

The masses of all binary substrings of an encoded polymer may be abstracted by the composition multiset of a string, provided that Assumption 1 holds. Assumption 2 slightly deviates from practical ion series measurements insofar as the latter only provides information about the masses of the prefixes and suffixes, while the proposed modification allows one to observe the masses of all substrings, but without a priori knowledge of their order. Observe that one can make use of platforms that provide mass information for all substrings but such systems require more than one string disassociation and are hence harder to implement and more expensive.

Unlike the work in [7] which has solely focused on the problem of determining under which conditions unique string reconstruction is possible, we view the problem of multiset composition analysis from a coding-theoretic perspective and ask the following questions:

Q1. *Can one add asymptotically negligible redundancy to information strings in such a way that unique reconstruction is possible, independent of the length of the strings?* Since only strings of specific lengths are reconstructable up to reversals, we aim to devise an efficiently encodable and decodable scheme that encodes all strings of length $k \geqslant 1$ into strings of a larger length $n \geqslant k$ that are uniquely reconstructable for *all possible string lengths*. Furthermore, we do not allow for both a string and its reversal to be included in the codebook. One simple means for ensuring that a string is uniquely reconstructable up to reversal is to pad the string with 0s to obtain the shortest length of the form $\min\{p-1, 2q-1\}$, where p and q primes. For example, if $k > 89693$, it is known that there exists a prime p such that $k-1 < p-1 < \left(1 + \frac{1}{\ln^3 k}\right) k-1$. The result only holds for very large k that are beyond the reach of polymer chemistry. Bertrand's postulate [20] applies

to shorter lengths $k > 3$ but only guarantees that $k - 1 < p - 1 < 2k - 4$. This implies a possible coding rate loss of up to $1/2$. Note that eliminating reversals of strings reduces the codebook size by less than a half.

Q2. *Can one add asymptotically negligible redundancy to information strings in such a way that unique reconstruction is possible even in the presence of errors, independent on the length of the strings?* We focus on mass error models under which the composition (mass) of one substring is erroneously interpreted as a different composition (mass). In the asymmetric error model, no two errors can simultaneously affect the masses of two substrings of length i and $k - i + 1$, while in the symmetric error model such pairs are allowed. Clearly, the two models are the same when only one mass error is present. Furthermore, asymmetric errors are easily detectable even without added redundancy, while symmetric errors may not be automatically detectable. Symmetric errors tend to be correlated as they arise during the same fragmentation process, while asymmetric errors may be independent as they arise during two different fragmentation processes. It is therefore of interest to analyze both cases.

We answer both questions affirmatively by describing coding schemes that allow for both unique reconstruction and correction of multiple symmetric and asymmetric mass errors. For the case of asymmetric errors, encoding is performed by interleaving symmetric strings with shifted Catalan-Bertrand paths while decoding is accomplished through a modification of the backtracking decoding algorithm described in [7]. For symmetric errors, the proposed encoding and decoding procedures use the polynomial factorization approach of [7] and add redundancy in a fashion similar to that included in Reed-Solomon codes.

Both lines of work extend the existing literature in string reconstruction [21–24] and coded string reconstruction [25–28].

The organization of this chapter is as follows. Section 2.2 introduces the problem, the relevant terminology and notation. The topic of reconstruction codes, or code design for unique reconstruction, is addressed in Section 2.3. Asymmetric error-correction codes with unique reconstruction properties are addressed in Section 2.4, while symmetric error-correction code constructions are discussed in Section 2.5. The chapter concludes with a discussion of open problems in Section 2.6.

2.2 Problem Statement

Let $\mathbf{s} = s_1 s_2 \dots s_k$ be a binary string of length $k \geqslant 2$. A substring of $\mathbf{s}$ starting at i and ending at j, where $1 \leqslant i \leqslant j \leqslant k$, is denoted by $\mathbf{s}_i^j$, and is said to have *composition* $0^z 1^w$, where $0 \leqslant z, w \leqslant j - i + 1$ stand for the number of 0s and 1s in the substring, respectively. Let $c(\mathbf{s}_i^j)$ denote the composition of $\mathbf{s}_i^j$, $i \leqslant j$. A composition only conveys information about the weight of the substring, but not the particular order of the bits. Furthermore, let $C_l(\mathbf{s})$ stand for the multiset of compositions of substrings of $\mathbf{s}$ of length l, $1 \leqslant l \leqslant k$; clearly, this multiset contains $k-l+1$ compositions. For example, if $\mathbf{s} = 100101$, then the substrings of length two are $10, 00, 01, 10, 01$, so that $C_2(\mathbf{s}) = \{0^1 1^1, 0^2, 0^1 1^1, 0^1 1^1, 0^1 1^1\}$.

The multiset $C(\mathbf{s}) = \cup_{l=1}^{k} C_l(\mathbf{s})$ is termed the *composition multiset*. It is straightforward to see that the composition multisets of a string $\mathbf{s}$ and its reversal, $\mathbf{s}^r = s_k s_{k-1} \dots s_1$, are identical and hence these two strings are indistinguishable based on $C(\cdot)$. We define the *cumulative weight* of a composition multiset $C_l(\mathbf{s})$, with compositions of the form $0^z 1^w$, where $z+w = l$, as $w_l(\mathbf{s}) = \sum_{0^z 1^w \in C_l(\mathbf{s})} w$. Observe that $w_1(\mathbf{s}) = w_k(\mathbf{s})$, as both equal the weight of the string $\mathbf{s}$. More generally, one has $w_l(\mathbf{s}) = w_{k-l+1}(\mathbf{s})$, for all $1 \leqslant l \leqslant k$. This assertion can be proved by counting the objects of interest in two different ways. One may arrange all substrings of length ℓ row-wise. In this case, the columns represent strings of length $k - \ell + 1$. The weight counts of the rows have to be the same as those of the columns, so that $w_\ell = w_{k+1-\ell}$.

In our subsequent derivations, we also make use of the following notation. For a string $\mathbf{s} = s_1 s_2 \dots s_k$, we let $\sigma_i = \mathrm{wt}(s_i s_{k-i+1})$ for $i \leqslant \lfloor \frac{k}{2} \rfloor$, and for odd k, $\sigma_{\lceil \frac{k}{2} \rceil} = \mathrm{wt}(s_{\lceil \frac{k}{2} \rceil})$, where wt stands for the weight of the string. For our running example $\mathbf{s} = 100101$, $\sigma_1 = 2$, while $\sigma_2 = 0$ (see Figure 2.2). We use $\Sigma^{\lceil \frac{k}{2} \rceil}$ to denote the sequence $(\sigma_i)_{i \in [\lceil \frac{k}{2} \rceil]}$, where $[a] = \{1, \dots, a\}$.

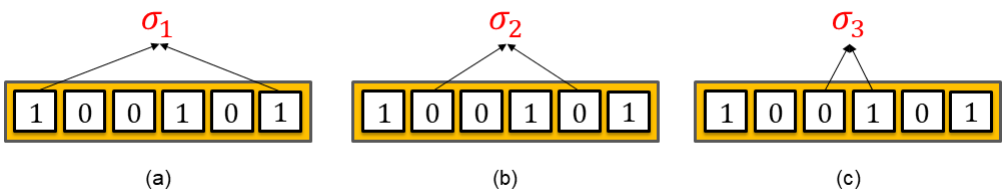


Figure 2.2: An illustration of $\Sigma^{\lceil \frac{k}{2} \rceil}$: For the string $\mathbf{s} = 100101$, $\Sigma^3 = [\sigma_1 = 2, \sigma_2 = 2, \sigma_3 = 1]$.

Whenever clear from the context, we omit the argument $\mathbf{s}$.

The problems of interest are as follows. The first problem pertains to reconstruction codes: A collection of binary strings of fixed length is called a **reconstruction code** if all the strings in the code can be reconstructed uniquely based on their multiset compositions. We seek reconstruction codes of small redundancy and consequently, large rate.

As part of the second problem, we consider **error-correcting reconstruction codes.** In this context, one is given a valid composition multiset of a string $\mathbf{s}$, $C(\mathbf{s})$. Within the multiset $C(\mathbf{s})$, some compositions may be arbitrarily corrupted to a composition of the same length. We refer to such errors as **composition errors**. For example, when $\mathbf{s} = 100101$, the multiset $C_2(\mathbf{s}) = \{0^1 1^1, 0^2, 0^1 1^1, 0^1 1^1, 0^1 1^1\}$ may be corrupted to $\tilde{C}_2(\mathbf{s}) = \{\mathbf{0^2}, 0^2, 0^1 1^1, 0^1 1^1, 0^1 1^1\}$, in which case we have a single composition error.

For the case where multiple composition errors may occur so that the symmetric difference between $C(\mathbf{s})$ and $\tilde{C}(\mathbf{s})$, denoted $C(\mathbf{s}) \triangle \tilde{C}(\mathbf{s})$, may contain more than one element, we will call the errors as asymmetric if the following condition holds: For each $i \in \{1, 2, \ldots, \lfloor \frac{n}{2} \rfloor\}$, if $\left| C_i(\mathbf{s}) \triangle \tilde{C}_i(\mathbf{s}) \right| \neq 0$, then $\left| C_{n-i+1}(\mathbf{s}) \triangle \tilde{C}_{n-i+1}(\mathbf{s}) \right| = 0$. In words, this means that the composition sets $C_i(\mathbf{s})$ and $C_{n-i+1}(\mathbf{s})$ cannot both be in error. For the case of symmetric errors, this condition need not hold (so that there are no restrictions on the structure of the composition errors), and asymmetric composition errors are a special case of symmetric composition errors. For the case where a single composition error occurs between $C(\mathbf{s})$ and $\tilde{C}(\mathbf{s})$, the single composition error is necessarily asymmetric (and therefore also symmetric).

Continuing from our previous example, the multisets $C_2(\mathbf{s})$ and $C_5(\mathbf{s})$ may be corrupted to $\tilde{C}_2(\mathbf{s}) = \{\mathbf{0^2}, 0^2, 0^1 1^1, 0^1 1^1, 0^1 1^1\}$ and $\tilde{C}_5(\mathbf{s}) = \{\mathbf{0^1 1^4}, 0^3 1^2\}$, in which case we say that we encountered an example of two *symmetric composition errors*, given that the substrings lengths 2 and 5 sum up to $k + 1 = 7$. Note that this example does not represent two asymmetric composition errors because an error has occurred in a composition of length $i = 2$ and also in composition of length $n - i + 1 = 6 - 2 + 1 = 5$.

Our main results are summarized below.

Theorem 1 establishes the existence of efficiently decodable reconstruction codes that have asymptotic rate one (proved in Section 2.3), while Theorem 2 establishes similar results for the case of reconstruction codes capable of

correcting one composition error (proved in Section 2.4.1).

Theorem 1. *There exist efficiently encodable and decodable reconstruction codes with information string-length k and redundancy at most $\frac{1}{2}\log(k)+5$.*

Theorem 2. *There exist efficiently encodable and decodable reconstruction codes with information string-length k capable of correcting a single composition error and redundancy at most $\frac{1}{2}\log(k)+13$.*

Theorems 3, 4 and 5 extend the results of Theorem 2 for the case of multiple composition errors, including both the asymmetric and symmetric case (proved in Sections 2.4.2, 2.5, and 2.5 respectively). The result in Theorem 3 demonstrates the existence of explicit asymmetric error-correcting reconstruction codes of asymptotic rate one that can be efficiently reconstructed for constant t. The result in Theorem 4 applies to symmetric errors. The best known redundancy is achieved using the construction supporting Theorem 5.

Theorem 3. *There exist efficiently encodable and decodable reconstruction codes with information string-length k capable of correcting a constant number of t asymmetric composition errors and redundancy $\mathcal{O}(t\log k)$. The decoding algorithm has complexity $\mathcal{O}(n^3\,2^t)$.*

Theorem 4. *There exist efficient symmetric t-error correcting reconstruction codes with information string-length k, redundancy $\mathcal{O}(t^2\log k)$ and decoding complexity $\mathcal{O}(n^3)$.*

Theorem 5. *There exist symmetric t-error correcting reconstruction codes with information string-length k, redundancy $\mathcal{O}(\log k+t)$ and decoding complexity $\mathcal{O}(n^{3+3t})$.*

2.2.1 Technical Background

For a string of length k, recall that $\sigma_i = \mathrm{wt}(s_i, s_{k+1-i})$, and that given C_1 one can compute $w_1 = \sum_{j=1}^{\lceil \frac{k}{2} \rceil} \sigma_j$. When $i = 2$, the bits at positions 1 and k contribute once to w_2, whereas the bits $2, \ldots, k-1$ all contribute twice to w_2.

Using C_2, we can obtain $\sigma_1 + 2\sum_{j=2}^{\lceil \frac{k}{2} \rceil} \sigma_j = w_2$. Generalizing this result for all

$C_i, i \leqslant \lceil \frac{k}{2} \rceil$ is straightforward, and gives the following equalities:

$$\frac{1}{i}\sigma_1 + \frac{2}{i}\sigma_2 + \cdots + \frac{i-1}{i}\sigma_{i-1} + \sigma_i + \sigma_{i+1} + \cdots + \sigma_{\lceil \frac{k}{2} \rceil} = \frac{1}{i}w_i. \tag{2.1}$$

The above system of $\lceil \frac{k}{2} \rceil$ linear equations with $\lceil \frac{k}{2} \rceil$ unknowns can be solved efficiently. Thus, for all error-free composition sets, one can find $\Sigma^{\lceil \frac{k}{2} \rceil}$.

Some of our code designs rely on the Backtracking algorithm [7], first used in the context of the Turnpike problem. We provide an example illustrating the operation of the algorithm. The composition multiset $C(\mathbf{s})$ of a string is given as the input to the algorithm, and its output is the set of all strings that have the same composition as $C(\mathbf{s})$.

Example 1. *Let* $\boldsymbol{s} = 1010001010$. *The sequence* $\Sigma^5 = (\sigma_1 = 1, \sigma_2 = 1, \sigma_3 = 1, \sigma_4 = 1, \sigma_5 = 0)$ *can be uniquely determined from the composition multiset. This follows from* $w_1 = \sum_{i=1}^{5} \sigma_i$ *and* $i\,w_1 - w_i = \sum_{j=1}^{i-1}(i-j)\sigma_j$, *for* $i = 1, \ldots, \lceil \frac{k}{2} \rceil$. *Solving the system of equations produces* Σ^5.

The Backtracking algorithm starts by determining the first and the last bit of the string and then proceeds to place the remaining bits in an inward fashion. Since $\sigma_1 = \mathrm{wt}(s_1 s_{10})$ *is known, and since a string and its reversal have the same composition multiset, the first and the last bits are placed arbitrarily. In our example, without loss of generality, the Backtracking algorithm sets* $s_1 = 1$ *and* $s_{10} = 0$ *(see Figure 2.3).*

Let ℓ_r *be the length of the reconstructed prefix/suffix pair. Backtracking produces a multiset of all compositions that are jointly determined by the reconstructed prefix and suffix of length* $\ell_r = 1$, $\boldsymbol{s}_1^1 = 1$, $\boldsymbol{s}_{10}^{10} = 0$ *and* Σ^5. *Denote this multiset by* $\mathcal{T}_{\ell_r=1}$.

Note that $\sigma_5 = 0$ *implies that the composition of* $\boldsymbol{s}_5^6$ *is* 0^2. *Similarly,* $\sigma_4 = 1$ *and* $\sigma_5 = 0$ *imply that the composition of* $\boldsymbol{s}_4^7$ *is* $0^3 1$. *Thus, using the information in* Σ^5 *alone one can reconstruct the following compositions:* $0^6 1^4, 0^5 1^3, 0^4 1^2, 0^3 1^1, 0^2$. *Note that compositions of substrings of the form* $\boldsymbol{s}_i^j$ *can be reconstructed provided that* i, j *satisfy: (1)* $i \leqslant j \leqslant \ell_r$ *or (2)* $k+1-\ell_r \leqslant i \leqslant j$ *or (3)* $i \leqslant \ell_r + 1$ *and* $j \geqslant k - \ell_r$. *Thus, the composition* $0^5 1^4$ *of* $\boldsymbol{s}_1^9$ *and the composition* $0^6 1^3$ *of* $\boldsymbol{s}_2^{10}$ *can both be reconstructed as well. Consequently,* $\mathcal{T}_1 = \{0^6 1^4, 0^5 1^4, 0^6 1^3, 0^5 1^3, 0^4 1^2, 0^3 1^1, 0^2, 0, 1\}$.

In the next step, the Backtracking algorithm tries to determine the bits s_2 *and* s_9. *First, recall that* $\sigma_2 = 1$ *is known. The algorithm determines*

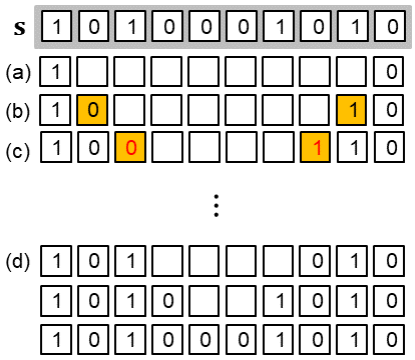


Figure 2.3: Illustration of the Backtracking algorithm for the string $\mathbf{s} = 1010001010$. (a) Backtracking algorithm begins by reconstructing s_1 and s_{10}. (b) When the weight of the reconstructed prefix ($\mathrm{wt}(1) = 1$) is not equal to that of the reconstructed suffix ($\mathrm{wt}(0) = 0$), the backtracking algorithm reconstructs s_2 and s_9 correctly. (c) However, when the weight of the reconstructed prefix ($\mathrm{wt}(10) = 1$) is equal to that of the suffix ($\mathrm{wt}(10) = 1$), the backtracking algorithm *guesses* the bits s_3 and s_8. (d) When the reconstructed string is at odds with the evidence composition multiset, the backtracking algorithm *backtracks* to its first guess and corrects it.

the compositions of the two longest substrings in the multiset $C \setminus \mathcal{T}_1$ *to be* $\{0^5 1^3, 0^5 1^3\}$. *Observe that these compositions must be those of the substrings* $\boldsymbol{s}_1^8$ *and* $\boldsymbol{s}_3^{10}$ *(although inconsequential for this example, it is still important to note that in general one does not know which one of the two largest compositions in* $C \setminus \mathcal{T}_1$ *correspond to the prefix). Hence, the compositions of the prefix-suffix pair* $\{\boldsymbol{s}_1^2, \boldsymbol{s}_9^{10}\}$ *equal* $\{01, 01\}$.

Since the weight of the reconstructed prefix is not equal to the weight of the reconstructed suffix, i.e., $\mathrm{wt}(\boldsymbol{s}_1^1) = 1 \neq 0 = \mathrm{wt}(\boldsymbol{s}_{10}^{10})$, *the Backtracking algorithm outputs* $s_2 = 0, s_9 = 1$. *This follows from fact that given that the reconstructed prefix-suffix pair have a weight mismatch, setting* $(s_2 = 0, s_9 = 1)$, *or setting* $(s_2 = 1, s_9 = 0)$ *leads to different prefix-suffix compositions. As a result,* $\{1^2, 0^2\} \neq \{01, 01\}$. *The algorithm completes this iteration by updating* $\mathcal{T}$ *to* $\mathcal{T}_{\ell_r=2} = \{0^6 1^4, 0^5 1^4, 0^6 1^3, 0^5 1^3, 0^5 1^3, 0^5 1^3, 0^4 1^3, 0^5 1^2, 0^4 1^2, 0^3 1^1, 0^2, 01, 01, 0, 1, 0, 1\}$.

In the next iteration, following the same steps described above, the compositions of the prefix-suffix pair of length 3 *are found to be* $\{01^2, 0^2 1\}$. *However, since* $\mathrm{wt}(\boldsymbol{s}_1^2) = \mathrm{wt}(\boldsymbol{s}_9^{10})$, *the Backtracking algorithm cannot determine the bits* s_3, s_8. *Thus, whenever* $\mathrm{wt}(\boldsymbol{s}_1^{\ell_r}) = \mathrm{wt}(\boldsymbol{s}_{k+1-\ell_r}^k)$ *and* $\sigma_{\ell+1} = 1$, *the algo-*

rithm guesses *the bits* $s_{\ell_r+1}, s_{n-\ell_r}$. *However, if* $\mathrm{wt}(\boldsymbol{s}_1^{\ell_r}) = \mathrm{wt}(\boldsymbol{s}_{k+1-\ell_r}^{k})$ *and* $\sigma_{\ell+1} \in \{0, 2\}$, *then the reconstruction of bits* s_{ℓ_r+1} *and* $s_{n-\ell_r}$ *is straightforward. For example, guessing that* $s_3 = 0$, *and* $s_7 = 1$ *leads to an error. The error is detected by encountering a multiset* $\mathcal{T}_{\ell_r}$ *that is incompatible with the composition multiset* C *of the given string. Upon detection of an error, the algorithm backtracks to the last position where it guessed the bit assignment, changes its guess and restarts the algorithm from that iteration. In our example, this leads to* $s_3 = 1$ *and* $s_8 = 0$, *and one hence obtains the reconstructed string* 1010001010. □

Note that if $\Sigma^{\lfloor \frac{k}{2} \rfloor} \in \{0, 2\}^{\lfloor \frac{k}{2} \rfloor}$, then the string reconstruction is straightforward (see Figure 2.4).

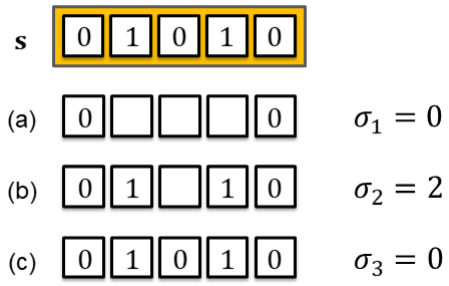


Figure 2.4: Illustration of the Backtracking algorithm for the string $\mathbf{s} = 01010$.

The complexity of the Backtracking algorithm is summarized in the following theorem.

Theorem. [7, Theorem 32] *Let*

$$\ell_s \stackrel{def}{=} |\{i \leqslant \lfloor \frac{n}{2} \rfloor : \mathrm{wt}(\boldsymbol{s}_1^i) = \mathrm{wt}(\boldsymbol{s}_{n+1-i}^n) \text{ and } s_{i+1} \neq s_{n-i}\}|,$$

$$E_s \stackrel{def}{=} \{\boldsymbol{v} : C(\boldsymbol{v}) = C(\boldsymbol{s})\},\ \ell_s^* \stackrel{def}{=} \max_{u \in E_s} \ell_u.$$

For a given input $C(\boldsymbol{s})$ *and* ℓ_s, *the Backtracking algorithm outputs a set of strings that contains* $\boldsymbol{s}$ *in time* $\mathcal{O}(2^{\ell_s} n^2 \log(n))$. *Furthermore,* E_s *can be recovered in time* $\mathcal{O}(2^{\ell_s^*} n^2 \log(n))$.

If a string has a length that does not allow for unique reconstruction up to reversal, the algorithm returns a set of strings and in the process backtracks multiple times. Backtracking is possible even when the string is uniquely

reconstructable, but a condition that ensures that the algorithm does not backtrack is that no prefix has a matching suffix of the same length and same weight. If the algorithm does not backtrack, the string has to be unique. This observation is crucial for our subsequent constructions and it motivates the use of Catalan-Bertrand paths discussed in what follows.

Theorem 6. (Whitworth [1878] , Bertrand [1887]) *Among all strings comprised of a 0s and b 1s, where $a \geqslant b$, there are $\binom{a+b}{a} - \binom{a+b}{a+1}$ strings in which every prefix has at least as many 0s as 1s. Note that when $a = b = h$,*

$$\binom{a+b}{a} - \binom{a+b}{a+1} = \frac{1}{h+1}\binom{2h}{h} = C_h.$$

The number C_h is known as the h^{th} Catalan number. *Note that the central binomial coefficient $\binom{2h}{h}$ also counts the number of strings of length $2h$ whose every prefix has at least as many 0s as 1s. Furthermore, note that the scaled central binomial coefficient $\frac{1}{2}\binom{2h}{h}$ counts the number of strings of length $2h$ whose every prefix contains strictly more 0s than 1s.*

The second part of Theorem 6 is proved in Appendix A.1.

Strings that have the property that their every prefix contains *strictly* more 0s than 1s are henceforth referred to as *Catalan-Bertrand* strings (see Figure 2.5).

We also find the following bounds on the central binomial coefficient useful in our subsequent derivations.

Proposition 1. The central binomial coefficient may be bounded [29] as:

$$\frac{2^{2h}}{\sqrt{\pi(h+1)}} \leqslant \binom{2h}{h} \leqslant \frac{2^{2h}}{\sqrt{\pi h}}, \ \forall\, h \geqslant 1. \tag{2.2}$$

2.3 Reconstruction Codes

We describe next a family of efficiently encodable and decodable reconstruction codes that map strings of any length k into strings of length $n \leqslant k + \frac{1}{2}\log(k) + 5$.

Recall that for a given string of length n, the system of $\lceil \frac{n}{2} \rceil$ linear equations with $\lceil \frac{n}{2} \rceil$ unknowns given by (2.1) can be solved efficiently. Thus, for all

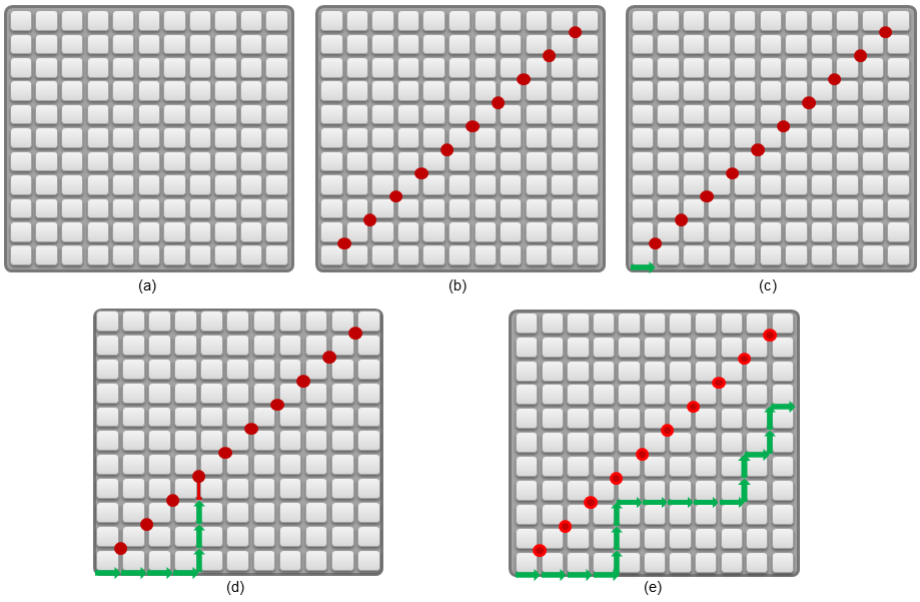


Figure 2.5: Catalan-Bertrand strings: (a) Consider a $n \times n$ grid whose (b) diagonal is blocked. A horizontal step in the grid represents bit 0 in its corresponding binary string, while a vertical step represents bit 1. (c) All Catalan-Bertrand strings start with a horizontal step and never meet the diagonal. Subfigure (d) is an example of a forbidden path, while (e) is an example of a path that corresponds to a valid Catalan-Bertrand string.

error-free composition sets, one can find $\Sigma^{\lceil \frac{n}{2} \rceil}$. Therefore, the problem of interest is to determine $\mathbf{s}$ given $\Sigma^{\lceil \frac{n}{2} \rceil}$ and $C(\mathbf{s})$. Note that when $\text{wt}(\mathbf{s}_1^i) \neq \text{wt}(\mathbf{s}_{n+1-i}^n)$, [7, Lemma 31] asserts that $C(\mathbf{s}), \mathbf{s}_1^i$, and $\mathbf{s}_{n-i+1}^n$ determine the ordered pair (s_{i+1}, s_{n-i}).

The previous lemma [7, Lemma 31] will be used to guide our construction of a reconstructible code based on Catalan-Bertrand strings.

Claim 1. *An asymptotic rate* 1 *reconstruction code can be constructed with Catalan strings.*

Claim 1 provides a simple construction for a family of reconstructible codes. We aim to construct codes with improved block rate over the Catalan strings and to that end we proceed as follows. Let $I \subseteq [n]$. The string formed by concatenating bits at positions in I in-order is denoted by $\mathbf{s}_I$. We define a

reconstruction code $\mathcal{S}_R(n)$ of even length n as follows:

$$\begin{aligned}\mathcal{S}_R(n) =& \{\mathbf{s} \in \{0,1\}^n, s_1 = 0, s_n = 1, \text{ and} \\ & \exists\, I \subseteq \{1, 2, \ldots, n-1, n\} \text{ such that} \\ & \qquad \text{for all } i \in I, s_i \neq s_{n+1-i}, \\ & \qquad \text{for all } i \notin I, s_i = s_{n+1-i}, \\ & \mathbf{s}_{[\frac{n}{2}]\cap I} \text{ is a Catalan-Bertrand string}\}.\end{aligned} \tag{2.3}$$

See Figure 2.6 for an example.

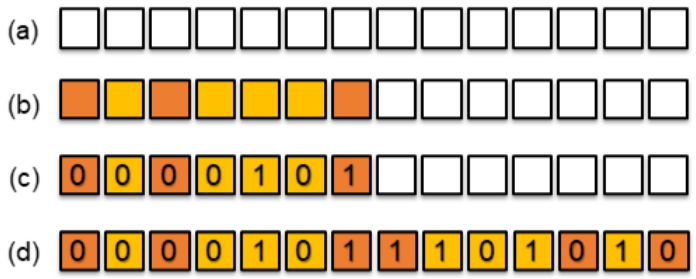


Figure 2.6: Construction of a codestring in $\mathcal{S}_R(14)$: The first 7 indices of the string (a) are partitioned into two (b). Consider the string $s_1 s_3 s_7$ formed by the in-place concatenation of the indices in the orange set. No restriction is placed on this string: $s_1 s_3 s_7 \in \{0,1\}^3$. However, string $s_2 s_4 s_5 s_6$ formed by the in-place concatenation of the indices in the yellow set is restricted to Catalan-Bertrand strings. (d) To complete the string, the partition is then mirrored. The substring $\mathbf{s}_8^{14}$ is such that the bits at the orange indices retain their mirror values, while those at the yellow indices complement them.

We define a reconstruction code $\mathcal{S}_R(n)$ of even length n as follows:
For odd n, we define the codebook as

$$\mathcal{S}_R(n) = \cup_{\mathbf{s}\in\mathcal{S}_R(n-1)} \{\mathbf{s}_1^{\frac{n-1}{2}}\ 0\ \mathbf{s}_{\frac{n+1}{2}}^{n-1},\ \mathbf{s}_1^{\frac{n-1}{2}}\ 1\ \mathbf{s}_{\frac{n+1}{2}}^{n-1}\}.$$

The following proposition is an immediate consequence of the above construction.

Lemma 1. *Consider a string $\boldsymbol{s} \in \mathcal{S}_R(n)$. For all prefix-suffix pairs of length $1 \leqslant j \leqslant \frac{n}{2}$, one has $\mathrm{wt}(\boldsymbol{s}_1^j) \neq \mathrm{wt}(\boldsymbol{s}_{n+1-j}^n)$.*

The proof of Theorem 1 follows from the fact that $\mathcal{S}_R(n)$ is a reconstruction

code, which may be easily established from the guarantees for the Backtracking algorithm and Lemma 1.

For n even, the size of $\mathcal{S}_R(n)$ may be bounded as:

$$|\mathcal{S}_R(n)| = \sum_{i=0}^{\frac{n}{2}-1} \binom{\frac{n}{2}-1}{i} 2^{\frac{n}{2}-1-i} \binom{i}{\lfloor \frac{i}{2} \rfloor} \geqslant \frac{2^{n-3}}{\sqrt{\pi n}}\,.$$

The first equality follows from the description of the codebook, while the second inequality follows from Proposition 1 and the binomial theorem. For odd n, $|\mathcal{S}_R(n)| = 2|\mathcal{S}_R(n-1)| \geqslant \frac{2^{n-3}}{\sqrt{\pi n}}$. Further details are provided in Appendix A.2. As $2^k \leqslant |\mathcal{S}_R(n)|$, simple algebraic manipulation reveals that the redundancy of the reconstruction code for information lengths k is at most $1/2 \log(k) + 5$ for all $k \geqslant 8$.

Given an information string of length k, the encoding algorithm returns a reconstructable string of length n. The encoding algorithm that accompanies our reconstruction codebook (a bijective map between the set of all information strings of length k and a subset of the reconstructable strings of length n) can be implemented using simple lexicographical rankings of Catalan-Bertrand strings and symmetric strings. This encoding technique requires $\mathcal{O}(n^2)$ time (see Appendix A.3 for details). However, as described in [30], there exist other ordering-based constructions for Catalan strings that may be used to further increase the efficiency of the encoding algorithm. The Backtracking algorithm reconstructs the coded string in $\mathcal{O}(n^3)$ time. The coded string is then mapped to the information string via the inverse encoding map, which takes an additive $\mathcal{O}(n^2)$ time. Thus, the overall reconstruction time remains $\mathcal{O}(n^3)$. This concludes the proof of Theorem 1.

2.4 Error-Correcting Reconstruction Codes: The Asymmetric Setting

For clarity of exposition, we will start with a discussion of single error-correcting reconstruction codes, as they illustrate the use of Catalan-Bertrand paths and are conceptually easy to extend for the case of multiple composition errors. Our reconstruction codes with composition error-correcting capabilities are derived using the interleaving procedure described in the

previous section, and they require adding an additional logarithmic number of redundant bits to recover the sequence $\Sigma^{\lceil \frac{n}{2} \rceil}$.

2.4.1 Single Error-Correcting Reconstruction Codes

We begin with an intuitive discussion that leads to Claim 2.

For a given codestring from $\mathcal{S}_R(n)$, the backtracking algorithm identifies a confusable string only if the composition of a prefix or a suffix is altered. Moreover, note that when the composition of a prefix (or a suffix) is drastically altered, the erroneous prefix (or a suffix) can be identified. This in turn implies that the string can be reconstructed. Thus, only certain kinds of errors may produce confusable strings. To see that, consider the following. Assume that we have reconstructed the prefix (and suffix) of length i and $\sigma_{i+1} = 1$. There are two valid compositions each for a prefix and a suffix. There are only 4 ways to modify the prefix-suffix composition pair such that the erroneous composition cannot be identified readily. (There are 4 valid compositions for the prefix and suffix of length $i+1$. There are 4 ways to pick a prefix-suffix pair such that they do not together correspond to any prefix-suffix substring pair; however, individually, they are all valid.) Since there are $n-1$ prefix-suffix pairs, there are at most $4(n-1)$ strings confusable with the given string.

To construct the codebook $\mathcal{S}_C^{(1)}(n)$ that can correct one error, pick any string $\mathbf{s} \in \mathcal{S}_R(n)$ and add to $\mathcal{S}_C^{(1)}(n)$. Remove all strings $\hat{\mathbf{s}} \in \mathcal{S}_R(n)$ that are at distance 2 as identified by the Backtracking algorithm. We repeat this procedure until $\mathcal{S}_R(n)$ is empty. Thus, $|\mathcal{S}_C^{(1)}(n)| \geqslant \frac{1}{4n}|\mathcal{S}_R(n)|$.

Claim 2. *There exists single error-correcting reconstruction codes with asymptotic rate* 1.

Let $\mathcal{S}_R(n-2)$ be the code of odd length $n-2$, $\lceil \frac{n}{2} \rceil$ divisible by three, as described in the previous section. Then, a single (symmetric or asymmetric) composition error-correcting code of length n, $\mathcal{S}_C^{(1)}(n)$, can be constructed by adding two bits to each string in $\mathcal{S}_R(n-2)$ and subsequently fixing the value of one additional bit. These three redundant bits allow us to uniquely recover the sequence $\Sigma^{\lceil \frac{n}{2} \rceil}$ in the presence of a single composition error. As will be seen from our subsequent derivations, given $\Sigma^{\lceil \frac{n}{2} \rceil}$ and the erroneous composition set of $\mathbf{s}$, one can reconstruct $\mathbf{s}$.

To prove Theorem 2, let $\tilde{C}(\mathbf{s})$ denote the set obtained by introducing a single error in the composition set $C(\mathbf{s})$ of a string $\mathbf{s}$ of length n. Recall that w_j stands for the cumulative weight of compositions of length j in C and that $w_j = w_{n-j+1}$. Let $\tilde{w}_j$ denote the cumulative weight of compositions in $\tilde{C}_j$. It is straightforward to prove the following proposition.

Proposition 2. *Let $j \in \left[\lceil \frac{n}{2} \rceil\right]$. Then,*

$$w_j = jw_1 - \sum_{i=1}^{j-1} i\,\sigma_{j-i},$$

which implies

$$jw_1 - \sum_{i=2}^{j-1} i\,\sigma_{j-i} \leqslant w_j \leqslant jw_1 - \sum_{i=2}^{j-1} i\,\sigma_{j-i} + 2.$$

Proof. Note that $w_j = \sum_{i=1}^{j-1} i\sigma_i + \sum_{i=j}^{\lceil \frac{n}{2} \rceil} j\sigma_i$. Since $w_1 = \sum_{i=1}^{\lceil \frac{n}{2} \rceil} \sigma_i$, we have

$$\begin{aligned} w_j =& \sigma_1 + 2\sigma_2 + \cdots + (j-1)\sigma_{j-1} + j\sum_{i=j}^{\lceil \frac{n}{2} \rceil} \sigma_j \\ =& j\left(\sum_{i=1}^{\lceil \frac{n}{2} \rceil} \sigma_i\right) - \sigma_{j-1} - 2\sigma_{j-2} - \cdots - (j-1)\sigma_1. \\ =& jw_1 - \sum_{i=1}^{j-1} i\sigma_{j-i}. \end{aligned}$$

□

This result immediately implies the next proposition.

Proposition 3. *Let $j \in \left[\lceil \frac{n}{2} \rceil\right]$ and suppose that we are given $w_1, \sigma_1, \ldots, \sigma_{j-2}$. Then, the value $w_j \bmod 3$ uniquely determines w_j.*

We also need the following three propositions.

Proposition 4. *Given $\mathrm{wt}(\boldsymbol{s}) \bmod 2$, $\tilde{w}_n$ and $\tilde{w}_1$, one can recover w_1.*

Proof. If $\tilde{w}_n = \tilde{w}_1$, then clearly $w_1 = \tilde{w}_n = \tilde{w}_1$. Hence, suppose that $\tilde{w}_n \neq \tilde{w}_1$ and observe that $|\tilde{w}_1 - w_1| \leqslant 1$. The last inequality follows since at most

one composition error is allowed. If $\tilde{w}_1 \bmod 2 = \mathrm{wt}(\mathbf{s}) \bmod 2$, then $w_1 = \tilde{w}_1$; otherwise, $w_1 = \tilde{w}_n$. □

Proposition 5. *Suppose that n is odd and that either $\lceil \frac{n}{2} \rceil + 1$ or $\lceil \frac{n}{2} \rceil$ is divisible by 3. Assume that $\boldsymbol{s} = s_1 \dots s_{\lceil \frac{n}{2} \rceil} \dots s_n$, and let $\boldsymbol{s}' = s_1 \dots 1 - s_{\lceil \frac{n}{2} \rceil} \dots s_n$. Then,*

$$\sum_{i=1}^{\lceil \frac{n}{2} \rceil} w_i(\boldsymbol{s}) \equiv \sum_{i=1}^{\lceil \frac{n}{2} \rceil} w_i(\boldsymbol{s}') \bmod 3.$$

Proof. Suppose that $s_{\lceil \frac{n}{2} \rceil} = 1$. Then, the bit $s_{\lceil \frac{n}{2} \rceil}$ contributes $\lceil \frac{n}{2} \rceil$ to $w_{\lceil \frac{n}{2} \rceil}$ and $\lceil \frac{n}{2} \rceil - 1$ to $w_{\lceil \frac{n}{2} \rceil - 1}$.

In summary, if $s_{\lceil \frac{n}{2} \rceil} = 1$, then

$$\sum_{i=1}^{\lceil \frac{n}{2} \rceil} w_i(\mathbf{s}) = \sum_{i=1}^{\lceil \frac{n}{2} \rceil} w_i(\mathbf{s}') + \frac{\lceil \frac{n}{2} \rceil \left(\lceil \frac{n}{2} \rceil + 1 \right)}{2}.$$

The result follows if either $\lceil \frac{n}{2} \rceil + 1$ or $\lceil \frac{n}{2} \rceil$ is divisible by 3. □

Proposition 6. *For odd n, if $s_1 \dots s_{\lceil \frac{n}{2} \rceil} \dots s_n \in \mathcal{S}_R(n)$, then $s_1 \dots 1 - s_{\lceil \frac{n}{2} \rceil} \dots s_n \in \mathcal{S}_R(n)$.*

The proof of the proposition is straightforward, as it follows from the definition of the reconstruction set $\mathcal{S}_R(n)$.

For odd n such that $\lceil \frac{n}{2} \rceil \equiv 0 \mod 3$ our code is defined as follows:

$$\begin{aligned} \mathcal{S}_C^{(1)}(n) = \Big\{ \mathbf{s} = s_1 s_2 s_3 \dots s_{\lceil \frac{n}{2} \rceil} \dots s_{n-2} s_{n-1} s_n \in \{0,1\}^n : \\ s_1 \mathbf{s}_3^{n-2} s_n = s_1 s_3 s_4 s_5 \dots s_{n-4} s_{n-3} s_{n-2} s_n \in \mathcal{S}_R(n-2), \\ \mathrm{wt}(\mathbf{s}) \bmod 2 = 0, \\ \sum_{i=1}^{\lceil \frac{n}{2} \rceil} w_i(\mathbf{s}) \equiv 0 \bmod 3, \text{ where } s_2 \leqslant s_{n-1} \Big\}. \end{aligned}$$

The size of the code $\mathcal{S}_C^{(1)}(n)$ is $\frac{|\mathcal{S}_R(n-2)|}{2}$, which follows from the second constraint that $s_1 s_3 \dots s_{n-2} s_n \in \mathcal{S}_R(n-2)$, along with Proposition 6. To construct a string in $\mathcal{S}_C^{(1)}(n)$, we first fix s_2 and s_{n-1} so that $\sum_{i=1}^{\lceil \frac{n}{2} \rceil} w_i(\mathbf{s}) \equiv 0 \bmod 3$. Then, we choose $s_{\lceil \frac{n}{2} \rceil}$ to satisfy $\mathrm{wt}(\mathbf{s}) \equiv 0 \bmod 2$. From Propositions 5 and 6, the resulting string belongs to $\mathcal{S}_C^{(1)}(n)$. The next proposition

shows that for certain codelengths, there exists values for s_2 and s_{n-1} that always allow for the constraints to be satisfied.

Proposition 7. *When $\lceil \frac{n}{2} \rceil$ is divisible by 3, then for any $\mathbf{x} = x_1 \dots x_{n-2} \in \{0,1\}^{n-2}$, there exists $s_2 s_{n-1} \in \{0,1\}^2$ so that*

$$\sum_{i=1}^{\lceil \frac{n}{2} \rceil} w_i\Big(x_1 s_2 x_2 x_3 \dots x_{n-3} s_{n-1} x_{n-2}\Big) \equiv 0 \bmod 3,$$

where $s_2 \leqslant s_{n-1}$.

Proof. Let $\mathbf{s} = x_1 s_2 x_2 x_3 \dots x_{n-3} s_{n-1} x_{n-2}$. Clearly, the elements s_2 and s_{n-1} appear in exactly one composition from $C_1(\mathbf{s})$ (recall that $C_i(\mathbf{s})$ denotes the set of compositions of $\mathbf{s}$ of length i). Furthermore, s_2 and s_{n-1} each appear twice in every set $C_i(\mathbf{s})$, where $\lceil \frac{n}{2} \rceil \geqslant i \geqslant 2$. Therefore the symbol s_2 appears in

$$2\left(\lceil \frac{n}{2} \rceil - 1\right) + 1$$

compositions from $C_1(\mathbf{s}) \cup C_2(\mathbf{s}) \cup \dots \cup C_{\lceil \frac{n}{2} \rceil}(\mathbf{s})$, and, by symmetry, the symbol s_{n-1} appears $2\left(\lceil \frac{n}{2} \rceil - 1\right) + 1$ times as well.
Suppose $\sum_{i=1}^{\lceil n/2 \rceil} w_i(s) \equiv a \bmod 3$ when $(s_2, s_{n-1}) = (0,0)$. Then, more generally if $(s_2, s_{n-1}) = (c_1, c_2) \in \{0,1\}^2$ where (c_1, c_2) are not necessarily equal to $(0,0)$ we have

$$\begin{aligned}\sum_{i=1}^{\lceil \frac{n}{2} \rceil} w_i(\mathbf{s}) &\equiv a + c_1\Big(2\big(\lceil \frac{n}{2} \rceil - 1\big) + 1\Big) + c_2\Big(2\big(\lceil \frac{n}{2} \rceil - 1\big) + 1\Big) \bmod 3 \\ &\equiv a - c_1 - c_2 \bmod 3.\end{aligned}$$

Since for the case $(c_1, c_2) = (0,0)$, $\sum_{i=1}^{\lceil \frac{n}{2} \rceil} w_i(\mathbf{s}) \equiv a \bmod 3$, for $(c_1, c_2) = (0,1)$, $\sum_{i=1}^{\lceil \frac{n}{2} \rceil} w_i(\mathbf{s}) \equiv a-1 \bmod 3$, and for $(c_1, c_2) = (1,1)$, $\sum_{i=1}^{\lceil \frac{n}{2} \rceil} w_i(\mathbf{s}) \equiv a-2 \bmod 3$. This completes the proof. □

For the next lemma, recall that $\tilde{C}(\mathbf{s})$ is the result of a single composition error in $C(\mathbf{s})$.

Lemma 2. *Suppose that $\boldsymbol{s} \in \mathcal{S}_C^{(1)}(n)$ where $\lceil \frac{n}{2} \rceil$ is divisible by 3. Then, given $\tilde{C}(\boldsymbol{s})$, one can recover $\Sigma^{\lceil \frac{n}{2} \rceil}$.*

Proof. In order to prove the claim, we show that given $\tilde{C}(\mathbf{s})$, one can recover $w_1, w_2, \ldots, w_n$, which we know uniquely determine $\Sigma^{n/2}$ according to (2.1). Let j be such that $\tilde{w}_j \neq \tilde{w}_{n+1-j}$. Since at most one single composition error is allowed, there exists at most one such j. It is straightforward to see that due to symmetry, either $\tilde{w}_j \neq w_j = w_{n+1-j}$ or $\tilde{w}_{n+1-j} \neq w_j = w_{n+1-j}$. Since $\mathrm{wt}(\mathbf{s}) \bmod 2 = 0$ by construction, it follows that we can determine w_1 based on Proposition 4. In addition, using the first identity from Proposition 2, it follows that we can recover $\sigma_1, \sigma_2, \ldots, \sigma_{j-2}$. Also, using the constraint $\sum_{i=1}^{\lceil \frac{n}{2} \rceil} w_i(\mathbf{s}) \equiv 0 \bmod 3$, we can recover $w_j \bmod 3$. Then, according to Proposition 3, we can recover w_j along with $w_1, \ldots, w_n$. One case left to consider is when $\tilde{w}_i = \tilde{w}_{n+1-i}$, for all $i \in [\lfloor \frac{n}{2} \rfloor]$. In this case, $\tilde{w}_{\lceil \frac{n}{2} \rceil} \neq w_{\lceil \frac{n}{2} \rceil}$. Applying Proposition 3 allows us to determine $w_{\lceil \frac{n}{2} \rceil}$ for this case as well. This completes the proof. □

Next, recall that $\mathcal{T}_i$ stands for the set of compositions of all substrings $\mathbf{s}_j^l$ for which $j < l \leqslant i$, or $n+1-i \leqslant j < l$, or $j \leqslant i+1$ *and* $n-i \leqslant l$, or $l = n+1-j$.

Let the two strings $\mathbf{s}$ and $\mathbf{v}$ be such that $\mathbf{s}_1^j = \mathbf{v}_1^j$ and $\mathbf{s}_{n+1-j}^n = \mathbf{v}_{n+1-j}^n$ and either $s_{j+1} \neq v_{j+1}$ or $s_{n-j} \neq v_{n-j}$. Then we say that the *longest prefix-suffix pair* shared by the two strings has length j.

Before we proceed to prove that $\mathcal{S}_C^{(1)}(n)$ is a single error-correcting code, we provide two illustrative examples - one for the case where the error occurs in a composition of length (size) $j \leqslant \lfloor \frac{n}{2} \rfloor$, and another for the case where the error occurs in a composition of length (size) $j \geqslant \lceil \frac{n}{2} \rceil$.

Example 2. *Let* $n = 11$ *and consider* $\boldsymbol{s} = 00001111111 \in \mathcal{S}_C^{(1)}(n)$. *Let the composition multiset with one composition error be* $\tilde{C}(\boldsymbol{s}) = (C(\boldsymbol{s}) \cup \{1^4\}) \setminus \{0^4\}$. *Given* $\tilde{C}(\boldsymbol{s})$, *by Lemma 2, we can infer* $\Sigma^6 = (1,1,1,1,2,1)$. *The Backtracking algorithm readily reconstructs up to* $s_1 s_2 s_3 = 000$, *and* $s_9 s_{10} s_{11} = 111$. *For further details on the Backtracking algorithm, refer to Example 1 and [7]. Next, observe that* $w_4 \neq w_8$, *and that the two largest compositions in* $\tilde{C}(\boldsymbol{s}) \setminus \mathcal{T}_3 = \{0^4 1^3, 1^7\}$ *are compatible with the reconstructed prefix, suffix and the constraints imposed by* Σ^6. *Thus, the Backtracking algorithm proceeds by reconstructing* $s_4 s_8 = 01$, *and computing* $\mathcal{T}_4$. *Note that for this string, due to the constraints imposed by* σ_5, *and* σ_6, *the string is immediately reconstructed as* 00001111111. *The Backtracking algorithm finds that* $0^4 \in \mathcal{T}_4$, *but* $0^4 \notin \tilde{C}(\boldsymbol{s})$. *However, this one single incompatibility is expected*

in the given error setting. In general, for the case of a single composition error, if the error occurs in a composition that corresponds to a substring of length $\leqslant \lfloor \frac{n}{2} \rfloor$*, it does not affect the Backtracking algorithm.* □

Example 3. *Let* $n = 11$ *and once again consider* $\boldsymbol{s} = 00001111111 \in \mathcal{S}_C^{(1)}(n)$*. Let the composition multiset with one composition error be* $\tilde{C}(\boldsymbol{s}) = (C(\boldsymbol{s}) \cup \{01^6\}) \setminus \{1^7\}$*. Given* $\tilde{C}(\boldsymbol{s})$*, by Lemma 2, we can infer* $\Sigma^6 = (1, 1, 1, 1, 2, 1)$*. The Backtracking algorithm readily reconstructs up to* $s_1 s_2 s_3 = 000$*, and* $s_9 s_{10} s_{11} = 111$*. Note that* $w_4 \neq w_8$*, and that the two largest compositions in* $\tilde{C}(\boldsymbol{s}) \setminus \mathcal{T}_3$ *equal* $\{0^4 1^3, 01^6\}$*, implying* $s_4 s_8 = 00$*. Clearly, one of these two compositions is erroneous as* $\sigma_4 = 1$*. Hence, the two possibilities for the bits* $s_4 s_8$ *are* 10 *and* 01*. If the Backtracking algorithm assigns* $s_1 s_2 s_3 s_4 = 0001$*, and* $s_8 s_9 s_{10} s_{11} = 0111$*, then note that while* $\mathcal{T}_4$ *contains only one* 1^4*,* $\tilde{C}(\boldsymbol{s})$ *contains four* 1^4*. In particular, the number of distinct elements in the sets* $\tilde{C}(\boldsymbol{s})$ *and* $\mathcal{T}_4$ *due to incorrect bit assignments is strictly greater than one. Thus, if the Backtracking algorithm erroneously reconstructs the bits* $s_4 s_8$*, it backtracks and assigns* $s_4 s_8 = 01$ *instead. As mentioned in Example 2, due to the constraints imposed by* σ_5*, and* σ_6*, the string is then correctly reconstructed as* 00001111111*.* □

Lemma 3. *Let* $\boldsymbol{s} \in \mathcal{S}_C^{(1)}(n)$*. Given* $\tilde{C}(\boldsymbol{s})$*, one can uniquely reconstruct the string* $\boldsymbol{s}$*.*

Proof. Let j denote the index of the composition multiset C_j that contains an error. As shown in the example, single composition errors that occur in a composition of a substring of length $j \leqslant \lfloor \frac{n}{2} \rfloor$ do not affect the reconstruction process, since the Backtracking algorithm only makes use of information provided by compositions of substrings of length $\geqslant \lceil \frac{n}{2} \rceil$. As the Backtracking algorithm progresses, the erroneous composition is identified through a comparison of the erroneous observed composition multiset and the iteratively constructed set $\mathcal{T}_\ell$, as explained in the above examples. Errors that happen for $j \leqslant \lfloor \frac{n}{2} \rfloor$ are easily identified and automatically corrected by the Backtracking algorithm. From Lemma 2, $\Sigma^{\lceil \frac{n}{2} \rceil}$ may be determined in an error-free manner. Using the obtained $\Sigma^{\lceil \frac{n}{2} \rceil}$, we run the Backtracking algorithm and in the process, we possibly run into incompatible compositions for $j \geqslant \lceil \frac{n}{2} \rceil$. Note that when $j = \lceil \frac{n}{2} \rceil$, the Backtracking algorithm has reconstructed the entire string. Given an already reconstructed prefix-suffix pair of length i,

$\mathbf{s}_1^i, \mathbf{s}_{n-i+1}^n$, we make use of the two largest cardinality compositions in $\tilde{C}(\mathbf{s}) \setminus \mathcal{T}_i$ to reconstruct the bits s_{i+1} and s_{n-i}. If $\sigma_{i+1} \in \{0, 2\}$, s_{i+1} and s_{n-i} can be determined immediately. Also, given $\Sigma^{\lceil \frac{n}{2} \rceil}$, if the Backtracking algorithm can correctly determine which of the above two compositions corresponds to a prefix and a suffix, then s_{i+1} and s_{n-i} can be uniquely identified. Otherwise, the Backtracking algorithm halts and performs a guess. Thus, in summary, an error occurring in a composition in C_j affects the reconstruction of the bits indexed by $i+1$ and $n-i$, where i is such that $j+i+1=n$.

Consider the case where the incompatibility manifests itself through $\mathcal{T}_i \not\subset \tilde{C}$, where $j = n-i-1$. Here, we identify the element that is in $\mathcal{T}_i$ but not in $\tilde{C}_j$, and add its weight to $\tilde{w}_j$ and compare it with $\tilde{w}_{n+1-j}$; this allows us to identify the erroneous composition. The assumption is that a composition corresponding to a substring of length j is erroneous. Clearly, $\tilde{C}_j = (C_j \setminus \{c_{i_1}\}) \cup \{c_{i_2}\}$, for some compositions indexed by i_1 and i_2, where c_{i_1} corresponds to the original, correct composition, while c_{i_2} corresponds to the erroneous composition. Since $\mathcal{T}_i$ contains some composition c_{i_3} of a substring of length j that is not present in $\tilde{C}_j$, it must be that $c_{i_3} = c_{i_1}$. Thus, we have $\mathrm{wt}(c_{i_2}) = \tilde{w}_j + \mathrm{wt}(c_{i_1}) - \tilde{w}_{n+1-j}$ and the erroneous composition can be identified and corrected. Next, suppose on the contrary that $\mathcal{T}_i \subset \tilde{C}$. In this case, consider the two largest compositions in $\tilde{C} \setminus \mathcal{T}_i$. The two largest compositions in $\tilde{C} \setminus \mathcal{T}_i$ are the compositions of a prefix-suffix pair of length j.

Since we have reconstructed the prefix and suffix of length i, and we know that $\sigma_{i+1} = 1$, the prefix-suffix pair is either $(\mathbf{s}_1^i 0, 1\mathbf{s}_{n+1-i}^n)$ or $(\mathbf{s}_1^i 1, 0\mathbf{s}_{n+1-i}^n)$. To show that only one of the constructed prefix-suffix pairs is valid/compatible, it suffices to show the following: For any two distinct strings $\mathbf{s}, \mathbf{v} \in \mathcal{S}_C^{(1)}(n)$ that have the same $\Sigma^{\lceil \frac{n}{2} \rceil}$, and are such that the longest prefix-suffix pairs shared by them is of length i, one has $|C(\mathbf{s}) \setminus C(\mathbf{v})| \geqslant 3$. Note that it now follows from Lemma 2 that for all strings $\mathbf{s} \in \mathcal{S}_C^{(1)}(n)$, $\Sigma^{\lceil \frac{n}{2} \rceil}$ can be determined. Thus, if two strings $\mathbf{u}, \mathbf{v} \in \mathcal{S}_C^{(1)}(n)$ share the same $\Sigma^{\lceil \frac{n}{2} \rceil}$ sequence, and $\mathbf{u}_1^{i+1} = \mathbf{s}_1^i 0, \mathbf{u}_{n-i}^n = 1\mathbf{s}_{n+1-i}^n$, and $\mathbf{v}_1^{i+1} = \mathbf{s}_1^i 1, \mathbf{v}_{n-i}^n = 0\mathbf{s}_{n+1-i}^n$, then $\tilde{C}(\mathbf{u}) = \tilde{C}(\mathbf{v})$ only if $|C(\mathbf{u}) \setminus C(\mathbf{v})| \leqslant 2$. Thus, $|C(\mathbf{u}) \setminus C(\mathbf{v})| \geqslant 3$ implies that $\tilde{C}(\mathbf{u}) \neq \tilde{C}(\mathbf{v})$. Observe that since $\mathbf{v} \in \mathcal{S}_C^{(1)}(n)$, the number of 0s in $c(\mathbf{s}_1^i)$ is at least by two larger than the number of 0s in $c(\mathbf{s}_{n+1-i}^n)$.

Let us assume that on the contrary, there are two strings $\mathbf{s}, \mathbf{v}$ such that $|C(\mathbf{s}) \setminus C(\mathbf{v})| \leqslant 2$, and that they differ only in their respective C_j sets.

Since the prefixes and suffixes of the strings of length $i = n - j - 1$ are identical, we let $s_1, \ldots, s_i$ and $s_{n+1-i}, \ldots, s_n$ denote the first and last i bits of both strings. Let $c(\mathbf{s})$ denote the composition of the string $\mathbf{s}$. Furthermore, let $c(\mathbf{s}_i^j)$ denote the composition of $\mathbf{s}_i^j$, $i \leqslant j$ and let $c = c(\mathbf{s}_{i+3}^{n-i-2})$.

When $n = 2(i+1)+1$, the strings differ in two compositions in C_{n-1-i} due to the assumption that the longest prefix-suffix pair shared by the two strings $\mathbf{s}$ and $\mathbf{v}$ is of length i. Since $\mathbf{s}$ and $\mathbf{v}$ share the same $\Sigma^{\lceil \frac{n}{2} \rceil}$, let $\mathbf{s}_{\lceil \frac{n}{2} \rceil} = \mathbf{v}_{\lceil \frac{n}{2} \rceil} = b$. Therefore, $\mathbf{s} = \mathbf{s}_1^i 0 b 1 \mathbf{s}_{n+1-i}^n$, $\mathbf{v} = \mathbf{s}_1^i 1 b 0 \mathbf{s}_{n+1-i}^n$. Observe that for the cases $b = 0$ and $b = 1$, $|C_{j-1}(\mathbf{s}) \setminus C_{j-1}(\mathbf{v})| \geqslant 1$. Thus, $|C(\mathbf{s}) \setminus C(\mathbf{v})| \geqslant 3$.

When $n \geqslant 2(i+1)+3$ and $\sigma_{i+2} = 1$, we let s_+ stand for the $(i+2)^{\text{th}}$ bit in the string $\mathbf{s}$, and v_+ stand for the $(i+2)^{\text{th}}$ bit of string $\mathbf{v}$. Figure 2.7 illustrates this setting. When $\sigma_{i+2} \in \{0, 2\}$, we let b denote the $(i+2)^{\text{th}}$ bits of the two strings, which are identical. Next, we determine conditions under which $C_{j-1}(\mathbf{s}) = C_{j-1}(\mathbf{v})$. We know that if the compositions of the two strings differ by three or more, the two strings cannot be confused under the single composition error model. Due to the constraints imposed by the very construction of the string, we know that $|C_j(\mathbf{s}) \setminus C_j(\mathbf{v})| = 2$. Thus, for the two strings not to be confusable under the given error model, $|C_\ell(\mathbf{s}) \setminus C_\ell(\mathbf{v})| > 0$ for some $\ell \in [n] \setminus \{j\}$. We show that a specific ℓ satisfying the previous inequality equals $j-1$, i.e., $|C_{j-1}(\mathbf{s}) \setminus C_{j-1}(\mathbf{v})| > 0$. Note that the compositions of substrings of length $n-i-2$ that contain the bits $i+1, \ldots, n-i$ are identical for the two strings.

Figure 2.7: Two strings $\mathbf{s}$ and $\mathbf{v}$ that satisfy the assumptions used in the proof.

Case 1: $\sigma_{i+2} = 1$. With a slight abuse of notation, we choose to write compositions as sets containing both bits and other compositions. On the left-hand side of the equation below, the compositions correspond to the substrings of $\mathbf{s}$ of length $n-i-2$ that *may* differ for the two strings. The right-hand side of the equation corresponds to the same entities in $\mathbf{v}$. If the

equation holds, then the multisets $C_{j-1}(\mathbf{s})$ and $C_{j-1}(\mathbf{v})$ are equal.

$$\left\{\begin{array}{l}\{c(\mathbf{s}_1^i), 0, s_+, c\},\\ \{c(\mathbf{s}_2^i), 0, s_+, c, 1-s_+\},\\ \{c(\mathbf{s}_{j+2}^n), 1, 1-s_+, c\},\\ \{c(\mathbf{s}_{j+2}^{n-1}), 1, 1-s_+, c, s_+\}\end{array}\right\}$$
$$= \left\{\begin{array}{l}\{c(\mathbf{s}_1^i), 1, v_+, c\},\\ \{c(\mathbf{s}_2^i), 1, v_+, c, 1-v_+\},\\ \{c(\mathbf{s}_{j+2}^n), 0, 1-v_+, c\},\\ \{c(\mathbf{s}_{j+2}^{n-1}), 0, 1-v_+, c, v_+\}\end{array}\right\}.$$

The exhaustive case-by-case arguments show that the above set equality is never true.

Case 2: $\sigma_{i+2} \in \{0, 2\}$ Similar reasoning leads to a set equality condition in which s_+ and v_+ are replaced by b. Once again, it can be shown by an exhaustive case-by-case analysis that the set equality never holds, independently on the choice of b. This implies that the composition sets $C_{j-1}(\mathbf{s})$ and $C_{j-1}(\mathbf{v})$ differ, which in turn implies that the composition multisets of the two strings are at distance $\geqslant 3$. □

Recall that when $\lceil \frac{n}{2} \rceil \mod 3 = 0$, the size of the code $\mathcal{S}_C^{(1)}(n)$ is $\frac{|\mathcal{S}_R(n-2)|}{2}$. In addition to the redundancy required to construct the reconstruction code, we require one bit to ensure $n-3$ is even, three bits to fix s_2, s_{n-1} and $s_{\lceil \frac{n}{2} \rceil}$, and four bits to ensure that $\lceil \frac{n}{2} \rceil$ is divisible by three. Thus, $\mathcal{S}_C^{(1)}(n)$ requires $\frac{1}{2}\log k + 13$ redundant bits.

The backtracking string reconstruction process based on an erroneous composition set is straightforward: It takes $\mathcal{O}(n^2)$ time to compute the $\mathcal{T}_k$ multiset, and backtracking performs $\mathcal{O}(n)$ steps. Thus, the decoding algorithm can computes the original string in $\mathcal{O}(n^3)$ time.

2.4.2 Multiple Error-Correcting Reconstruction Codes: The Asymmetric Case

We consider an error model in which each of the multisets $C_i \cup C_{n+1-i}$, $i \in [n]$ is allowed to contain at most one composition error and the total number of

errors is at most t. The codes described in what follows add asymptotically negligible redundancy to the information strings to correct a fixed number of t asymmetric errors. To construct the codes, we generalize the approach used in the previous section for correcting a single error.

We start with the description of a *t-shifted* reconstruction code of even length m, denoted by $\mathcal{S}_R^{(t)}(m)$ and defined below.

$$\begin{aligned}\mathcal{S}_R^{(t)}(m) =\{&\mathbf{s} \in \{0,1\}^m, \mathbf{s}_1^t = \mathbf{0}, \mathbf{s}_{m-t+1}^m = \mathbf{1}, \text{ and} \\ &\exists\, I \subseteq \{t+1, \ldots, m-t\} \text{ such that} \\ &\qquad \forall\, i \in I, s_i \neq s_{m+1-i}, \\ &\qquad \text{and } \forall\, i \notin I, s_i = s_{m+1-i}, \\ &\mathbf{s}_{[m/2]\cap I} \text{ is a Catalan-Bertrand string}\}.\end{aligned} \tag{2.4}$$

We refer to strings of the form $\mathbf{s}_1^t \mathbf{s}_{[m/2]\cap I}$ as *t-shifted* Catalan-Bertrand strings. For a $\mathbf{s} \in \mathcal{S}_R^{(t)}(m)$, every prefix of length i where $\frac{m}{2} \geqslant i \geqslant t+1$, has at least $t+1$ more 0s than its corresponding suffix of the same length.

Lemma 4. *Let $\mathbf{s}, \mathbf{v} \in \mathcal{S}_R^{(t)}(m)$ share the same $\Sigma^{\frac{m}{2}}$ sequence and satisfy $|C_j(\mathbf{s}) \setminus C_j(\mathbf{v})| \leqslant 2$ for all $j \in [m]$. If the longest prefix-suffix pair shared by $\mathbf{s}$ and $\mathbf{v}$ is of length i, then their corresponding composition multisets $C_{m-i-1}, C_{m-i-2}, \ldots, C_{m-i-t}, C_{m-i-t-1}$ each differ in at least 2 compositions.*

We defer the proof to Appendix A.4.

Corollary 1. *Let $\mathbf{s} \in \mathcal{S}_R^{(t)}(m)$, and let $\tilde{C}(\mathbf{s})$ be the composition multiset $C(\mathbf{s})$ corrupted by at most t asymmetric errors. Then, given the correct $\Sigma^{\frac{m}{2}}$ sequence, the string $\mathbf{s}$ can be uniquely reconstructed from $\tilde{C}(\mathbf{s})$.*

Proof. The result immediately follows from Lemma 4. □

Henceforth, we use $\mathcal{S}_{CA}^{(t)}(n)$ to denote an asymmetric t-error-correcting reconstruction code. Strings $\mathbf{s} \in \mathcal{S}_{CA}^{(t)}(n)$ are constructed by adding $n-m$ redundancy bits to a string $\mathbf{s}' \in \mathcal{S}_R^{(t)}(m)$ of even length in such a way that the $\Sigma^{\lceil \frac{n}{2} \rceil}$ sequence can be recovered even in the presence of t asymmetric errors.

Claim 3. *Let $\mathbf{s}$ be an arbitrary string of even length n and let $\tilde{C}(\mathbf{s})$ denote the composition multiset $C(\mathbf{s})$ corrupted by t asymmetric errors. Then, at least $\frac{n}{2} - 3t$ elements in $(\sigma_1, \sigma_2, \ldots, \sigma_{\frac{n}{2}})$ can be determined based on $\tilde{C}(\mathbf{s})$.*

Proof. The claim is a consequence of a simple analysis of the set of linear equations in (2.1). Clearly, w_i is unknown whenever $C_i \cup C_{n+1-i}$ contains an error. Therefore, if we have t errors we only have $\frac{n}{2} - t$ linear equations that involve $\frac{n}{2}$ variables. From this system of $\frac{n}{2} - t$ linear equations we form a new system of linear equations by subtracting equation (2.1) with index i from the equation (2.1) with index $i + 1$. Note that for all values of i such that w_{i-1}, w_i and w_{i+1} are known, the value of σ_i can be found from the new system of equations. Thus, the derived system of equations allows one to infer at least $\frac{n}{2} - 3t$ elements of the $\Sigma^{\frac{n}{2}}$ sequence. Note that all the expressions above assume that n is even. For odd n, $\lceil \frac{n}{2} \rceil$ should be used instead. □

We illustrate the above claim with an example. If w_3, w_4 and w_5 are known then using the linear equations corresponding to $i = 3$ and $i = 4$, one can infer $\sum_{k=4}^{\frac{n}{2}} \sigma_k$ and using the linear equations corresponding to $i = 4$ and $i = 5$, one can infer $\sum_{k=5}^{\frac{n}{2}} \sigma_k$. Thus, one can determine $\sigma_4 = \sum_{k=4}^{\frac{n}{2}} \sigma_k - \sum_{k=5}^{\frac{n}{2}} \sigma_k$.

Thus, to recover the entire $\Sigma^{\frac{n}{2}}$ sequence, it suffices to take the $\Sigma^{\frac{n}{2}}$ string from a systematic Reed-Solomon code over the alphabet $\{0, 1, 2\}$ that can correct up to $3t$ erasures.

Thus, the codestrings $\mathbf{s} \in \mathcal{S}_{CA}^{(t)}(n)$ are constructed via the following procedure:

- Pick a string $\mathbf{s}' = \mathbf{s}_1'^{\frac{m}{2}} \mathbf{s}'^{m}_{\frac{m}{2}+1} \in \mathcal{S}_R^{(t)}(m)$.
- Using a systematic Reed-Solomon code over the alphabet $\{0, 1, 2\}$ that can correct up to $3t$ erasures, the $\Sigma^{\frac{m}{2}}$ sequence is mapped to $\Sigma^{\frac{n}{2}}$. Note that the sequence $(\sigma_{\frac{m}{2}+1}, \ldots, \sigma_{\frac{n}{2}})$ is appended to $\Sigma^{\frac{m}{2}}$.
- A string $\mathbf{b}$ of length $n-m$ is created using the sequence $(\sigma_{\frac{m}{2}+1}, \ldots, \sigma_{\frac{n}{2}})$ as follows. For all $k \in \left[\frac{n-m}{2}\right]$:

$$b_k b_{n-m+1-k} = \begin{cases} 00, & \text{if } \sigma_{\frac{m}{2}+k} = 0; \\ 01, & \text{if } \sigma_{\frac{m}{2}+k} = 1; \\ 11, & \text{if } \sigma_{\frac{m}{2}+k} = 2. \end{cases}$$

- A codestring $\mathbf{s} \in \mathcal{S}_{CA}^{(t)}(n)$ is obtained by concatenating the strings $\mathbf{s}'$ and $\mathbf{b}$, namely $\mathbf{s} = \mathbf{s}_1'^{\frac{m}{2}}\, \mathbf{b}_1^{n-m}\, \mathbf{s}'^{m}_{\frac{m}{2}+1}$.

Given $\tilde{C}(\boldsymbol{s})$, the composition multiset $C(\boldsymbol{s})$ corrupted by t asymmetric errors, the string $\mathbf{s}$ can be uniquely reconstructed via the the following four-step procedure:

- Construct the linear system of equations governed by (2.1) using the erroneous composition multiset.
- Solve for the σ_i values that can be inferred from the linear system.
- Infer the correct $\Sigma^{\frac{n}{2}}$ sequence using an efficient polynomial evaluation decoder.
- Reconstruct the string $\mathbf{s}$ using the Backtracking algorithm.

The procedure described above requires $\frac{1}{2}\log n + 6$ redundant bits to ensure the Catalan-Bertrand string structure of even length, $2t$ redundant bits for the t-shifted structure and $3t\log n$ redundant bits to correct erasures in the $\Sigma^{\frac{n}{2}}$ sequence. Thus, the number of redundant bits r required is $\left(\frac{1}{2}+3t\right)\log n + 2t + 6$. Furthermore, r does not exceed $\left(\frac{1}{2}+3t\right)\log k + 2t + 7 + \left(\frac{1}{2}+3t\right)\frac{1}{\kappa}$, where κ is supremum over all $\kappa > 0$ such that $n \geqslant (1+\kappa)\left(\left(\frac{1}{2}+3t\right)\log n + 2t + 7\right)$.

Recall that the Backtracking algorithm takes $\mathcal{O}(n^3)$ time to reconstruct the string (it takes $\mathcal{O}(n^2)$ time to find the longest compositions in the set $\mathcal{T}_\ell \setminus C$, and reconstruct the bits $s_{\ell+1}s_{n-\ell}$; and, there are $\frac{n}{2}$ such pairs to be reconstructed). With a slight abuse of notation, we say that index i corresponds to an asymmetric error if a single composition error occurred in $C_i \cup C_{n+1-i}$. Now assume that the indices $i, i+1, \ldots, j$ $(j \geqslant i)$ correspond to composition lengths that contain asymmetric errors, and that $C_{i-1}, C_{n+2-i}, C_{j+1}, C_{n-j}$ are error-free. Note that the proof of Lemma 4 established that the Backtracking algorithm can reconstruct the correct substrings $\mathbf{s}_i^j\mathbf{s}_{n+1-j}^{n+1-i}$ before proceeding to reconstruct the bits s_{j+1}, s_{n-j}. Thus, every contiguous burst of errors of length t' causes an additional $\mathcal{O}(n^2 2^{t'})$ reconstruction time delay. Thus, the worst case reconstruction time is $\mathcal{O}(n^2 2^t)$.

Combining this result with that of Corollary 1 establishes Theorem 3.

2.5 Multiple Error-Correcting Reconstruction Codes: The Symmetric Case

We now turn our attention to designing reconstruction codes capable of correcting symmetric composition errors. The proposed method leverages a polynomial formulation of the composition reconstruction problem first described in [7]. The main result is a constructive proof for the existence of codes with $\mathcal{O}(t^2 \log k)$ bits of redundancy capable of correcting t symmetric composition errors.

To this end, we first review the results of [7] describing how to formulate the string reconstruction problem in terms of bivariate polynomial factorization.

For a string $\mathbf{s} \in \{0,1\}^n$, let $P_{\mathbf{s}}(x,y)$ be a bivariate polynomial of degree n with coefficients in $\{0,1\}$ such that $P_{\mathbf{s}}(x,y)$ contains exactly one term with total degree $i \in \{0,1,\ldots,n\}$. If $\mathbf{s} = s_1 \ldots s_n$ and if $\Big(P_{\mathbf{s}}(x,y)\Big)_i$ denotes the unique term of total degree i, then $\Big(P_{\mathbf{s}}(x,y)\Big)_0 = 1$, and

$$\Big(P_{\mathbf{s}}(x,y)\Big)_i = \begin{cases} y\Big(P_{\mathbf{s}}(x,y)\Big)_{i-1}, & \text{if } s_i = 0, \\ x\Big(P_{\mathbf{s}}(x,y)\Big)_{i-1}, & \text{if } s_i = 1. \end{cases}$$

In words, we use y to denote the bit 0 and x to denote the bit 1 and then summarize the composition of all prefixes of the string $\mathbf{s}$ in polynomial form. As a simple example, for $\mathbf{s} = 0100$ we have $P_{\mathbf{s}}(x,y) = 1 + y + xy + xy^2 + xy^3$. To see why this is true, we start with the free coefficient 1, then add y to indicate that the prefix of length one of the string equals 0, add xy to indicate that the prefix of length two contains one 0 and one 1, add xy^2 to indicate that the prefix of length three contains two 0s and one 1 and so on.

We also introduce another bivariate polynomial $S_{\mathbf{s}}(x,y)$ to describe the composition multiset $C(\mathbf{s})$ in a manner similar to $P_{\mathbf{s}}(x,y)$. In particular, we now associate each composition with a monomial in which the symbol y represents the bit 0 and the symbol x with the bit 1. As an example, for $\mathbf{s} = 0100$ we have

$$C(\mathbf{s}) = \Big\{0, 1, 0, 0, 01, 01, 0^2, 0^2 1, 0^2 1, 0^3 1\Big\},$$

and

$$S_{\mathbf{s}}(x,y) = x + 3y + 2xy + y^2 + 2xy^2 + xy^3,$$

where the first two terms in $S_{\mathbf{s}}(x,y)$ indicate that the composition multiset contains one substring 1 and three substrings 0; the next three terms indicate that the string contains two substrings with one 1 and one 0 and one substring with two 0s. The remaining terms are interpreted similarly.

The key identity from [7] is of the form

$$P_{\mathbf{s}}(x,y)\, P_{\mathbf{s}}\left(\frac{1}{x},\frac{1}{y}\right) = (n+1) + S_{\mathbf{s}}(x,y) + S_{\mathbf{s}}\left(\frac{1}{x},\frac{1}{y}\right). \tag{2.5}$$

Given a bivariate polynomial $f(x,y)$, we use $f^*(x,y)$ to denote its reciprocal polynomial, defined as

$$f^*(x,y) = x^{\deg_x(f)} y^{\deg_y(f)} f\left(\frac{1}{x},\frac{1}{y}\right),$$

where $\deg_x(f)$ denotes the x-degree of $f(x,y)$ and $\deg_y(f)$ denotes its y-degree. For simplicity, we hence write $d_x = \deg_x(P_{\mathbf{s}})$ and $d_y = \deg_y(P_{\mathbf{s}})$. Using the notion of the reciprocal polynomial we can rewrite the expression in (2.5) as:

$$P_{\mathbf{s}}(x,y)\, P^*_{\mathbf{s}}(x,y) = x^{d_x} y^{d_y}\ (n+1+S_{\mathbf{s}}(x,y)) + S^*_{\mathbf{s}}(x,y). \tag{2.6}$$

Note that if $\tilde{C}(\mathbf{s})$ is the composition multiset resulting from t symmetric composition errors in $C(\mathbf{s})$ and $\tilde{S}_{\mathbf{s}}(x,y)$ is the polynomial representation of $\tilde{C}(\mathbf{s})$ while $S_{\mathbf{s}}(x,y)$ is the polynomial representation of $C(\mathbf{s})$, then

$$\tilde{S}_{\mathbf{s}}(x,y) = S_{\mathbf{s}}(x,y) + E(x,y),$$

where $E(x,y)$ has at most $2t$ nonzero coefficients. Note that the coefficients of $E(x,y)$ lie in $\{-t, -t+1, \ldots, -1, 0, 1, \ldots, t-1, t\}$. A composition error corresponds to removing a multinomial e_t from $S_{\mathbf{s}}(x,y)$ and adding a different multinomial e_f. Thus, $-e_t$, and $+e_f$ are addends in $E(x,y)$. Since up to t errors are possible, the coefficients of every multinomial in $E(x,y)$ are integers in $\{-t, -t+1, \ldots, -1, 0, 1, \ldots, t-1, t\}$. If every multinomial removed from or added to $S_{\mathbf{s}}(x,y)$ is unique, then there are $2t$ terms in $E(x,y)$. Otherwise, the number of multinomials is less than $2t$. Our first result relates $\tilde{S}_{\mathbf{s}}(x,y)$

and $P_{\mathbf{s}}(x, y)$.

Claim 4. *Suppose that* $\mathrm{wt}(\boldsymbol{s}) \bmod (2t+1) = c_w$ *for some* $c_w \in \{0, 1, \ldots, 2t\}$. *Then, given* $\tilde{S}_{\boldsymbol{s}}(x, y)$ *and* c_w *one can generate*

$$P_{\boldsymbol{s}}(x, y)\, P^*_{\boldsymbol{s}}(x, y) + \tilde{E}(x, y),$$

where the polynomial $\tilde{E}(x, y)$ *has at most* $4t$ *terms.*

Proof. First, recall that $\tilde{S}_{\mathbf{s}}(x, y) = S_{\mathbf{s}}(x, y) + E(x, y)$ where $E(x, y)$ has at most $2t$ nonzero coefficients. Given c_w, we can easily determine the exact degrees d_x and d_y of the polynomial encoding of $\mathbf{s}$: In the error-free case, the sum of all compositions of length 1 (i.e., the sum of the bits of the string) equals $\mathrm{wt}(\mathbf{s}) = d_x$. When the composition multiset is erroneous, we can only observe $\tilde{d}_x$, which takes a value in the set $\{d_x - t, d_x - t + 1, \ldots, d_x, d_x + 1, d_x + 2, \ldots, d_x + t - 1, d_x + t\}$. Equivalently, we know that

$$d_x \in \{\tilde{d}_x - t, \tilde{d}_x - t + 1, \ldots, \tilde{d}_x, \tilde{d}_x + 1, \tilde{d}_x + 2, \ldots, \tilde{d}_x + t - 1, \tilde{d}_x + t\}.$$

Since $d_x \equiv c_w \mod (2t+1)$, exactly one value in the set $\{\tilde{d}_x - t, \tilde{d}_x - t + 1, \ldots, \tilde{d}_x, \tilde{d}_x + 1, \tilde{d}_x + 2, \ldots, \tilde{d}_x + t - 1, \tilde{d}_x + t\}$ will satisfy this condition. Hence, d_w can be inferred exactly, and since $d_y = n - d_x$, the same conclusion holds for d_y.

Next, we form $P_{\mathbf{s}}(x, y)\, P^*_{\mathbf{s}}(x, y)$ as follows:

$$\begin{aligned}
& x^{d_x} y^{d_y} \left(n + 1 + \tilde{S}_{\mathbf{s}}(x, y) + \tilde{S}_{\mathbf{s}}\left(\frac{1}{x}, \frac{1}{y}\right)\right) \\
& = x^{d_x} y^{d_y} (n+1) + x^{d_x} y^{d_y} \times \\
& \left(S_{\mathbf{s}}(x, y) + E(x, y) + S_{\mathbf{s}}\left(\frac{1}{x}, \frac{1}{y}\right) + E\left(\frac{1}{x}, \frac{1}{y}\right)\right) \\
& = P_{\mathbf{s}}(x, y)\, P^*_{\mathbf{s}}(x, y) + x^{d_x} y^{d_y} \left(E(x, y) + E\left(\frac{1}{x}, \frac{1}{y}\right)\right) \\
& = P_{\mathbf{s}}(x, y)\, P^*_{\mathbf{s}}(x, y) + \tilde{E}(x, y),
\end{aligned}$$

where $\tilde{E}(x, y) = x^{d_x} y^{d_y} \left(E(x, y) + E\left(\frac{1}{x}, \frac{1}{y}\right)\right)$ has at most $4t$ nonzero coefficients, which proves the desired result. □

Let $\mathbb{F}_q$ be a finite field of order q, where q is an odd prime. Let $\alpha \in \mathbb{F}_q$ be a primitive element of the field. For a polynomial $f(x) \in \mathbb{F}_q[x]$, let $\mathcal{R}(f)$ denote

the set of its roots. We find the following result useful for our subsequent derivations.

Theorem 7. *([31, Ch. 5]) Assume that $E(x) \in \mathbb{F}_q[x]$ has at most t nonzero coefficients. Then, $E(x)$ can be uniquely determined in $\mathcal{O}(n^2)$ time given $E(\alpha^t), E(\alpha^{t-1}), \ldots, E(\alpha^0), E(\alpha^{-1}), \ldots, E(\alpha^{-t})$.*

2.5.1 The Code Construction

Our approach to constructing a symmetric t-error-correcting code of length n, denoted by $\mathcal{S}_{CS}^{(t)}(n)$, relies on the fact that $\tilde{E}(x, y)$ may be written as:

$$\begin{aligned}\tilde{E}(x,y) =&(a_{i_1,1}y^{j_{i_1,1}} + \cdots + a_{i_1,m_{i_1}}y^{j_{i_1,m_{i_1}}})x^{i_1} + \\ &(a_{i_2,1}y^{j_{i_2,1}} + \cdots + a_{i_2,m_{i_2}}y^{j_{i_2,m_{i_2}}})x^{i_2} + \\ &\vdots \\ &(a_{i_h,1}y^{j_{i_h,1}} + \cdots + a_{i_h,m_{i_h}}y^{j_{i_h,m_{i_h}}})x^{i_h},\end{aligned} \tag{2.7}$$

where each $a_{i,j} \in \{-1, 1\}$, $h \leqslant 4t$ and the total number of nonzero terms is $\leqslant 4t$. Since $\tilde{E}(x, y)$ is restricted to have at most $4t$ nonzero terms, each of the polynomials $(a_{i_\ell,1}y^{j_{i_\ell,1}} + \cdots + a_{i_\ell,m_{i_\ell}}y^{j_{i_\ell,m_{i_\ell}}})$ can contain at most $4t$ nonzero terms. Consequently, one has $m_{i_\ell} \leqslant 4t$ for all $\ell \in \{1, 2, \ldots, h\}$.

Based on the previous observations we are ready to introduce our first code construction. We assume that $P_{\mathbf{s}}(x, y)$ is a bivariate polynomial over the field $\mathbb{F}_q$, where q is the smallest prime $\geqslant 2n + 1$. Clearly, for a $P_{\mathbf{s}}(x, y) \in \mathbb{I}[x, y]$ over the set integers $\mathbb{I}$, one can obtain $P_{\mathbf{s}}(x, y) \in \mathbb{F}_q[x, y]$ by simply reducing $P_{\mathbf{s}}(x, y)$ modulo q.

Lemma 5. *Let*

$$\begin{aligned}\mathcal{C} = \{\boldsymbol{s} \in \{0,1\}^n \ s.t. \ &\mathrm{wt}(\boldsymbol{s}) \bmod 2t+1 = 0, \\ &\{1, \alpha, \alpha^2, \ldots, \alpha^{4t}\} \subseteq \mathcal{R}(P_{\boldsymbol{s}}(x, 1)), \\ &\{1, \alpha, \alpha^2, \ldots, \alpha^{4t}\} \subseteq \mathcal{R}(P_{\boldsymbol{s}}(x, \alpha)), \\ &\vdots \\ &\{1, \alpha, \alpha^2, \ldots, \alpha^{4t}\} \subseteq \mathcal{R}(P_{\boldsymbol{s}}(x, \alpha^{4t}))\}.\end{aligned}$$

Then, $\mathcal{C}$ is a symmetric t-error-correcting code.

Proof. We prove the claim by describing a decoding algorithm that for any given $\tilde{S}_{\mathbf{s}}(x,y)$, which is the result of at most t composition errors occurring in $S_{\mathbf{s}}(x,y)$, uniquely recovers $S_{\mathbf{s}}(x,y)$.

Since there are at most t erroneous compositions in $\tilde{S}_{\mathbf{s}}(x,y)$, one can determine $\mathrm{wt}(\mathbf{s})$ by summing up the length-one compositions (i.e., the bits) in $\tilde{S}_{\mathbf{s}}(x,y)$ along with the fact that $\mathrm{wt}(\mathbf{s}) \bmod 2t+1 = 0$. Therefore, from Claim 4, we can construct the polynomial

$$F(x,y) = P_{\mathbf{s}}(x,y)\, P^*_{\mathbf{s}}(x,y) + \tilde{E}(x,y), \tag{2.8}$$

where $\tilde{E}(x,y)$ has at most $4t$ nonzero coefficients.

Suppose that $\beta, \beta' \in \mathbb{F}_q$. First, observe that if $P_{\mathbf{s}}(\beta,\beta')\, P^*_{\mathbf{s}}(\beta,\beta') = 0$, then $P_{\mathbf{s}}(\frac{1}{\beta},\frac{1}{\beta'})\, P^*_{\mathbf{s}}(\frac{1}{\beta},\frac{1}{\beta'}) = 0$ which immediately follows from the definition of $P^*_{\mathbf{s}}(x,y)$. Thus, if (β,β') is a root of $P_{\mathbf{s}}(\cdot,\cdot)$, then so is (β^{-1},β'^{-1}). Since $\{1,\alpha,\alpha^2,\ldots,\alpha^{4t}\} \subseteq \mathcal{R}(P_{\mathbf{s}}(\alpha^{\ell_1},y))$ for all $\ell_1 \in \{0,1,\ldots,4t\}$, and similarly $\{1,\alpha,\alpha^2,\ldots,\alpha^{4t}\} \subseteq \mathcal{R}(P_{\mathbf{s}}(x,\alpha^{\ell_2}))$ for all $\ell_2 \in \{0,1,\ldots,4t\}$, it follows that $F(\alpha^{\ell_1},\alpha^{\ell_2}) = \tilde{E}(\alpha^{\ell_1},\alpha^{\ell_2})$. Hence, we have:

$$\begin{aligned}
\tilde{E}(\alpha^{\ell_1},\alpha^{\ell_2}) = & \\
&\big(a_{i_1,1}\alpha^{\ell_2\times j_{i_1,1}} + \cdots + a_{i_1,m_{i_1}}\alpha^{\ell_2\times j_{i_1,m_{i_1}}}\big)\alpha^{\ell_1\times i_1} \\
+\,&\big(a_{i_2,1}\alpha^{\ell_2\times j_{i_2,1}} + \cdots + a_{i_2,m_{i_2}}\alpha^{\ell_2\times j_{i_2,m_{i_2}}}\big)\alpha^{\ell_1\times i_2} \\
&\qquad\qquad\vdots \\
+\,&\big(a_{i_{i_h},1}\alpha^{\ell_2\times j_{i_{i_h},1}} + \cdots + a_{i_{i_h},m_{i_{i_h}}}\alpha^{\ell_2\times j_{i_{i_h},m_{i_{i_h}}}}\big)\alpha^{\ell_1\times i_{i_h}},
\end{aligned}$$

for $\ell_1,\ell_2 \in \{0,1,\ldots,4t,-1,-2,\ldots,-4t\}$. From Theorem 7, for any fixed ℓ_2 we know the evaluations $\tilde{E}(\alpha^{\ell_1},\alpha^{\ell_2})$ for $\ell_1 \in \{0,1,\ldots,4t,-1,-2,\ldots,-4t\}$, so that we can recover the polynomials

$$\begin{aligned}
\tilde{E}(x,\alpha^{\ell_2}) = &\big(a_{i_1,1}\alpha^{\ell_2\times j_{i_1,1}} + \cdots + a_{i_1,m_{i_1}}\alpha^{\ell_2\times j_{i_1,m_{i_1}}}\big)x^{i_1} \\
+\,&\big(a_{i_2,1}\alpha^{\ell_2\times j_{i_2,1}} + \cdots + a_{i_2,m_{i_2}}\alpha^{\ell_2\times j_{i_2,m_{i_2}}}\big)x^{i_2} \\
&\qquad\qquad\vdots \\
+\,&\big(a_{i_{i_h},1}\alpha^{\ell_2\times j_{i_{i_h},1}} + \cdots + a_{j_{i_{i_h},m_{i_{i_h}}}}\alpha^{\ell_2\times j_{i_{i_h},m_{i_{i_h}}}}\big)x^{i_{i_h}},
\end{aligned} \tag{2.9}$$

using a decoder for a cyclic Reed-Solomon code of complexity $\mathcal{O}(n^2)$.

Let

$$M_{i_\ell}(y) = a_{i_\ell,1} y^{j_{i_\ell,1}} + \cdots + a_{i_\ell,m_{i_\ell}} y^{j_{i_\ell,m_{i_\ell}}}$$

be the polynomial multiplier of x^{i_ℓ} in $\tilde{E}(x,y)$. From the previous discussion, we know that the maximum number of nonzero terms in $M_{i_\ell}(y)$ is $4t$. Using (2.9), we can determine $M_{i_\ell}(\alpha^{\ell_2})$ for $\ell_2 \in \{0,1,2,\ldots,4t,-1,-2,\ldots,-4t\}$. Due to Theorem 7, this implies that we can recover $M_{i_\ell}(y)$ for $\ell \in \{1,2,\ldots,h\}$ once again using a decoder for a Reed-Solomon code. Since

$$\tilde{E}(x,y) = M_{i_1}(y)x^{i_1} + M_{i_2}(y)x^{i_2} + \cdots + M_{i_h}(y)x^{i_h},$$

we can determine $E(x,y)$ by noting the following: 1) Given $\mathrm{wt}(\mathbf{s}) \mod (2t+1)$, we can recover $\mathrm{wt}(\mathbf{s})$ from the erroneous composition multiset, from which d_x and $d_y = n - d_x$ can be determined. 2) Since d_x, d_y are known, and $\tilde{E}(x,y) = x^{d_x}y^{d_y}\left(E(x,y) + E\left(\frac{1}{x},\frac{1}{y}\right)\right)$, $E(x,y)$ can be determined. Subsequently we can reconstruct $S_{\mathbf{s}}(x,y)$ given $\tilde{S}_{\mathbf{s}}(x,y)$. □

The following corollary is an immediate consequence of Lemma 5.

Corollary 2. *Let*

$$\mathcal{C} = \{\boldsymbol{s} \in \{0,1\}^n \ s.t. \ P_{\boldsymbol{s}}(\alpha^{\ell_1},\alpha^{\ell_2}) = a_{\ell_1,\ell_2}, \\ \mathrm{wt}(\boldsymbol{s}) \equiv a \bmod 2t+1\},$$

for all $\ell_1,\ell_2 \in \{0,1,\ldots,4t\}$, $a \in \{0,1,\ldots,2t\}$, *and where* $(a_{\ell_1,\ell_2})_{\ell_1=0,\ell_2=0}^{4t}$ *is an arbitrary vector from* $\mathbb{F}_q^{(4t+1)^2}$. *Then,* $\mathcal{C}$ *can correct* t *symmetric composition errors.*

2.5.2 A Systematic Encoder $\mathcal{E}_{t,n}$

We construct next a systematic encoder $\mathcal{E}_{t,n}$ for the previously proposed codes.

Let r be the number of redundant bits in the proposed code construction. We will show in Theorem 4 that for all n, one requires a redundancy that

does not exceed

$$4\Big[(4t+1)^2(\log(2n+1)+1)+\log(2t+1) \\ +t\log\left((4t+1)^2(\log(2n+1)+1)+\log(2t+1)\right)\Big] \\ +\frac{1}{2}\log(n)+5.$$

One can show that r does not exceed $156t^2\log 8n$. Thus, $r=\mathcal{O}(t^2\log n)$. Furthermore, r does not exceed $156t^2\log 8k+156t^2\left(\frac{1}{\kappa}\right)$, where κ is supremum over all $\kappa>0$ such that $n\geqslant 156(1+\kappa)(t^2\log 8n+1)$. To express the redundancy in terms of the information length k, we upper bound $ct^2\log n$, where c is a constant, as follows. First, we write

$$ct^2\log n=ct^2\left(\log k+\log\left(\frac{n-k+k}{k}\right)\right).$$

Then, we upper bound the term $\log\left(\frac{n-k+k}{k}\right)$ using the Taylor series for $\log(1+x)$ and the linear term involved to arrive at $\log\left(\frac{n-k+k}{k}\right)<\frac{n-k}{k}$. For k_0 large enough and for all $k\geqslant k_0$, $\frac{n-k}{k}ct^2$ can be upper bound by a constant independent of n and k under the given parameter assumptions.

The encoder $\mathcal{E}_{t,n}$ takes as input the string $\mathbf{u}\in\{0,1\}^{n-\hat{r}}$, where $\hat{r}>0$ is a redundancy to be precisely specified later, and it produces a string $\mathbf{s}$. The evaluations of the polynomial $P_{\mathbf{s}}(x,y)$ is stored in

$$\left(w_1,w_2,\ldots,w_{\frac{\hat{r}}{2}}\right)\bmod 2,$$

where we recall that w_i stands for the cumulative weight of compositions of length i in $C(\mathbf{s})$.

Let $\mathcal{E}_t:\{0,1\}^m\to\{0,1\}^{m+t\log m}$ be a systematic encoder for a code with minimum Hamming distance $2t+1$ that inputs a string of length m and outputs a string of length $m+t\log m$. We will use this encoder with $m=(4t+1)^2(1+\log(2n+1))+\log(2t+1)$. Clearly, such a code exists since binary BCH codes of odd minimum distance have the desired set of parameters.

Encoder $\mathcal{E}_{t,n}:\{0,1\}^{n-\hat{r}}\to\{0,1\}^n$.

Input String $\mathbf{u}\in\{0,1\}^{n-\hat{r}}$.

Output Symmetric t-error-correcting codestring $\mathbf{s}\in\{0,1\}^n$.

1. Let $\alpha\in\mathbb{F}_q$ be a primitive element and let q be an odd prime $\geqslant 2n+1$.

For $\ell_1, \ell_2 \in \{0, 1, \ldots, 4t\}$, set $a_{\ell_1,\ell_2} = P_{\mathbf{u}}(\alpha^{\ell_1}, \alpha^{\ell_2})$, $\mathbf{a} = (a_{\ell_1,\ell_2})_{\ell_1=0,\ell_2=0}^{4t}$. Let $a = \mathrm{wt}(\mathbf{u}) \bmod 2t+1$.

2. Let $\bar{\mathbf{s}} = \mathcal{E}_{\mathbf{t}}(a, \mathbf{a}) \in \{0,1\}^{\frac{\hat{r}}{4}}$.

3. For $j \in \{1, 2, \ldots, \frac{\hat{r}}{2}\}$, define $\mathbf{z} = (z_1 \ldots z_{\frac{\hat{r}}{2}})$ as

$$z_j = \begin{cases} \sum_{i=1}^{j-1} z_i \bmod 2, & \text{if } j \text{ is odd and } \bar{s}_{\frac{j+1}{2}} = 0, \\ \sum_{i=1}^{j-1} z_i + 1 \bmod 2, & \text{if } j \text{ is odd and } \bar{s}_{\frac{j+1}{2}} = 1, \\ 0, & \text{if } j \text{ is even.} \end{cases}$$

4. Set $\mathbf{s} = \mathbf{0}\,\mathbf{u}\,\mathbf{z} \in \{0,1\}^n$, where $\mathbf{0}$ is an all-zero string of length $\frac{\hat{r}}{2}$.

The t-error-correcting code $\mathcal{S}_{CS}^{(t)}(n)$ is generated by the following two-step procedure:

- An information string of length k is first encoded using the reconstruction code $\mathcal{S}_R$, resulting in the string $\mathbf{u} \in \mathcal{S}_R(n - \hat{r})$.
- The string $\mathbf{u}$ is passed through the encoder $\mathcal{E}_{t,n}$, resulting in the codestring $\mathbf{s} = \mathcal{E}_{t,n}(\mathbf{u}) \in \mathcal{S}_{CS}^{(t)}(n)$.

Based on the above analysis, we set $\hat{r}$ to be the smallest integer $\geqslant r - \left(\frac{1}{2}\log(n) + 5\right)$ that is divisible by 4.

The redundancy of the code may be calculated as follows:

1. Since $q \geqslant 2n+1$, every α_{ℓ_1,ℓ_2}, $\ell_1, \ell_2 \in \{0, 1, \ldots 4t\}$ requires at most $1 + \log(2n+1)$ (due to the fact that given any positive integer x, there exits a prime number between x and $2x$).

2. Note that a requires $\log 2t + 1$ bits of redundancy. Thus, $\frac{\hat{r}}{4}$ is at most

$$\begin{aligned} &(4t+1)^2(1 + \log(2n+1)) + \log(2t+1) \\ &+ t\log((4t+1)^2(1 + \log(2n+1)) + \log(2t+1)). \end{aligned}$$

3. As already observed, the reconstructable string $\mathbf{u}$ requires at most $\frac{1}{2}\log n + 5$ bits of redundancy.

The redundancy of the encoder $\mathcal{E}_{t,n}$ is $\mathcal{O}(t^2 \log n)$ bits.

We find the following claims useful in our subsequent derivations.

Claim 5. *At Step 3 of the encoding procedure, for odd $j \in [\frac{\hat{r}}{2}]$, one has*

$$\bar{s}_{\frac{j+1}{2}} = \sum_{i=1}^{j} z_i \bmod 2. \tag{2.10}$$

This claim obviously follows from the definition of the string $\mathbf{z}$.

Recall next that for a string $\mathbf{s} \in \{0,1\}^n$, its $\Sigma^{\lceil \frac{n}{2} \rceil}$ sequence $(\sigma_1, \sigma_2, \ldots, \sigma_{\lceil \frac{n}{2} \rceil}) \in \{0,1,2\}^{\lceil \frac{n}{2} \rceil}$ equals $\sigma_i = s_i + s_{n+1-i}$. As a result of Step 4 of encoding with $\mathcal{E}_{t,n}$, we have the next result.

Claim 6. *For $j \in [\frac{\hat{r}}{2}]$,*

$$z_j = \sigma_{\frac{\hat{r}}{2}+1-j}.$$

The next claim connects the quantities w_i and $\bar{\mathbf{s}}$, defined in Step 2 of the encoding procedure.

Claim 7. *For $j \in \frac{\hat{r}}{4}$, the following holds*

$$w_{2j} \equiv \bar{s}_j \bmod 2.$$

Proof. The result is a consequence of the observation that

$$\begin{aligned} w_{2j} \equiv & 2jw_1 - (2j-1)\sigma_1 - (2j-2)\sigma_2 - \cdots - \sigma_{2j-1} \bmod 2 \\ \equiv & \sigma_1 + \sigma_3 + \cdots + \sigma_{2j-1} \bmod 2, \end{aligned}$$

where the first line follows from Equation (2.1). From Claims 5 and 6, and the previous observation, and the fact that we set $z_j = 0$ for even values of j in Step 3 of the encoding procedure, we have

$$w_{2j} \equiv \sum_{i=1}^{2j-1} \sigma_j \equiv \sum_{i=1}^{2j-1} z_j \equiv \bar{s}_j \bmod 2.$$

□

The next result will be used to prove the main finding regarding symmetric error-correction codes, as stated in Theorem 4.

Lemma 6. *The collection of strings*

$$\mathcal{C} = \left\{ \boldsymbol{s} : \boldsymbol{s} = \mathcal{E}_{t,n}(\boldsymbol{u}), \boldsymbol{u} \in \{0,1\}^{n-\hat{r}} \right\}$$

constitutes a symmetric t-error-correcting code.

Proof. In order to prove the result, we will describe how to recover $S_{\mathbf{s}}(x,y)$ given $\tilde{S}_{\mathbf{s}}(x,y)$, where $\tilde{S}_{\mathbf{s}}(x,y)$ is the result of at most t composition errors in $S_{\mathbf{s}}(x,y)$ for a codestring generated according to $\mathcal{E}_{t,n}(\mathbf{u}) = \mathbf{s}$.

We begin by forming the string

$$\tilde{\mathbf{w}} = \left(\tilde{w}_2, \tilde{w}_4, \ldots, \tilde{w}_{\frac{\hat{r}}{2}} \right).$$

One can obtain $\tilde{\mathbf{w}}$ from $\tilde{S}_{\mathbf{s}}(x,y)$ by summing up the 1s in all compositions of length two to get $\tilde{w}_2$, summing up the 1s in all compositions of length four to get $\tilde{w}_4$, and so on. For simplicity, let $\mathbf{w} = \left(w_2, w_4, \ldots, w_{\frac{\hat{r}}{2}} \right)$ for the string $\mathbf{s}$.

Since there are at most t composition errors in $\tilde{S}_{\mathbf{s}}(x,y)$, it follows that

$$d_H\left(\mathbf{w} \bmod 2, \tilde{\mathbf{w}} \bmod 2 \right) \leqslant t.$$

From Claim 7, since $\mathbf{w}$ mod 2 belongs to a code with minimum Hamming distance $2t+1$, we can recover $\mathbf{w}$ mod 2 from $\tilde{\mathbf{w}}$ mod 2. Then, given $\mathbf{w}$ mod 2, we can recover $\bar{\mathbf{s}}$ from Step 2 of the encoding procedure, and from $\bar{\mathbf{s}}$ we can determine $a = \mathrm{wt}(\mathbf{u}) \mod (2t+1)$. Using $\bar{\mathbf{s}}$, it is also straightforward to determine $\mathbf{z}$ from Step 3 of the encoding procedure. Thus, $\mathrm{wt}(\mathbf{z})$ is determined accurately as well. One can then easily determine the exact (yet potentially erroneous) weight of $\mathbf{u}$, since $\mathrm{wt}(\mathbf{u}) = \mathrm{wt}(\mathbf{s}) - \mathrm{wt}(\mathbf{z})$. Given $\tilde{\mathrm{wt}}(\mathbf{s})$, as determined from the sum of all compositions of substrings of length one, since we know (1) $\left| \left[\tilde{\mathrm{wt}}(\mathbf{s}) - \mathrm{wt}(\mathbf{z}) \right] - \mathrm{wt}(\mathbf{u}) \right| \leqslant t$, and (2) $a = \mathrm{wt}(\mathbf{u}) \mod (2t+1)$, we can infer $\mathrm{wt}(\mathbf{u})$ exactly. Subsequently, we can recover

$$\mathrm{wt}(\mathbf{s}) = \mathrm{wt}(\mathbf{u}) + \mathrm{wt}(\mathbf{z}),$$

and from $\mathrm{wt}(\mathbf{s})$, we can determine d_x and d_y, the x and y degrees of the polynomial $P_{\mathbf{s}}(x,y)$.

Next, we turn our attention to recovering the evaluations of the polynomial $P_{\mathbf{s}}(\alpha^{\ell_1}, \alpha^{\ell_2})$ for $\ell_1, \ell_2 \in \{0, 1, \ldots, 4t\}$. These, along with $\mathrm{wt}(\mathbf{s})$, suffice

according to Lemma 5 to recover $\mathbf{s}$. From $\bar{\mathbf{s}}$, we can determine $P_{\mathbf{u}}(\alpha^{\ell_1}, \alpha^{\ell_2})$ according to Steps 1 and 2 of the encoding procedure.

Let $d_{x,\mathbf{u}} = \deg_x(P_{\mathbf{u}}(x,y))$ and $d_{y,\mathbf{u}} = \deg_y(P_{\mathbf{u}}(x,y))$.

First, note that

$$\begin{aligned} P_{\mathbf{s}}(x,y) = P_{\mathbf{0}}(x,y) + y^{\frac{\hat{r}}{2}}(P_{\mathbf{u}}(x,y)-1) \\ + x^{d_{x,\mathbf{u}}} y^{\frac{\hat{r}}{2}+d_{y,\mathbf{u}}} (P_{\mathbf{z}}(x,y)-1). \end{aligned}$$

Therefore, since $\mathbf{z}$ is already known, we have

$$\begin{aligned} P_{\mathbf{s}}(\alpha^{\ell_1}, \alpha^{\ell_2}) = P_{\mathbf{0}}(\alpha^{\ell_1}, \alpha^{\ell_2}) + \alpha^{\ell_2 \times \frac{\hat{r}}{2}}(P_{\mathbf{u}}(\alpha^{\ell_1}, \alpha^{\ell_2})-1) \\ + \alpha^{\ell_1 \times d_{x,\mathbf{u}}} \alpha^{\ell_2 \times (\frac{\hat{r}}{2}+d_{y,\mathbf{u}})} (P_{\mathbf{z}}(\alpha^{\ell_1}, \alpha^{\ell_2})-1). \end{aligned}$$

The proof of the claim now follows from Corollary 2. □

We are left with the task of reconstructing the string $\mathbf{s}$ from its correct composition multiset $C(\mathbf{s})$. Recall that if all pairs of prefixes and suffixes of the same length are such that their weights differ, the string can be reconstructed efficiently by the Backtracking algorithm. Also, recall that the string $\mathbf{s}$ is obtained by concatenating three strings, *i.e.*, $\mathbf{s} = \mathbf{0}\,\mathbf{u}\,\mathbf{z}$. The prefix of length $\frac{\hat{r}}{2}$ is fixed to be all zeros and can therefore be reconstructed immediately. Lemma 6 allows one to recover the suffix $\mathbf{z}$. Since $\mathbf{u} \in \S_R(n-\hat{r})$, every prefix of length $\leqslant \lfloor \frac{n}{2} \rfloor$ has strictly more 0s than its corresponding suffix of the same length. Thus, the Backtracking algorithm can efficiently reconstruct the correct string $\mathbf{s}$. This establishes the result of Theorem 4.

We conclude our exposition by describing another family of uniquely reconstructable codes that can correct up to t composition errors in $C(\mathbf{s})$. These codes rely on the use of *Catalan paths*. Recall that Catalan paths of length $2h$ may be represented by binary strings that have the property that every prefix has at least as many 0s as 1s and the weight of the strings is h.

Let $\mathcal{P}(2h) \subset \{0,1\}^{2h}$ denote the set of Catalan strings of even length $2h$. It is well known that the codebook $\mathcal{P}(2h)$ has approximately $\frac{3}{2}\log h$ bits of redundancy, which follows directly from the expression for the Catalan number $C_h = \frac{1}{h+1}\binom{2h}{h}$.

The main differences between the polynomial construction and the Catalan-based designs are that the former has a *larger order of redundancy* ($\mathcal{O}(t^2 \log n)$ compared to $\mathcal{O}(\log n + t)$) but also has *an efficient decoding algorithm*. At

this point, no algorithm scaling efficiently with both n and t is known for the Catalan-based construction.

The basic idea behind the construction is simple and it imposes two constraints on the underlying codestrings:

1. **The Catalan string constraint**: This constraint requires that the codestrings be Catalan.

2. **Parity symbols**: The codestrings need to include $4t + 1$ 0s in the prefix and $4t + 1$ 1s in the suffix.

Intuitively, the fixed prefixes of 0s and suffixes of 1s, as well as the balancing property of Catalan strings, ensure that for at least $4t + 1$ choices of ℓ, the compositions multisets $C_\ell(\mathbf{s})$ and $C_\ell(\mathbf{v})$ of two distinct codestrings $\mathbf{s}$ and $\mathbf{v}$ differ in at least one composition.

Throughout our subsequent exposition, due to the heavy use of subscripts and superscripts, we write $-i$ instead of $n - i + 1$ for all indices used.

Let

$$\begin{aligned}\mathcal{C}(n,t) = \Big\{ & \mathbf{s} \in \{0,1\}^n \,:\, s_1 \,\ldots\, s_{4t+1} = 0\,0\,\ldots 0, \\ & s_{-4t-1}\, s_{-4t} \,\ldots\, s_{-1} = 1\,1 \ldots\, 1, \\ & s_{4t+2}\, s_{4t+3} \,\ldots\, s_{-4t-2} \in \mathcal{P}(n - 2(4t+1)) \Big\},\end{aligned} \tag{2.11}$$

where n is even.

We show next that $\mathcal{C}(n,t)$ is a t symmetric composition error-correcting code with $\mathcal{O}(\log n + t)$ bits of redundancy. This redundancy is significantly improved compared to that of the previously described polynomial evaluation construction.

Henceforth, $S_1 \triangle S_2 = (S_1 \setminus S_2) \cup (S_2 \setminus S_1)$ is used to denote the symmetric difference of two sets S_1 and S_2.

Theorem 8. *The code $\mathcal{C}(n,t)$ can correct t composition errors.*

Proof. We prove the result by showing that any pair of distinct codestrings $\mathbf{s}, \mathbf{v} \in \mathcal{C}$ satisfies

$$|C(\mathbf{v}) \triangle C(\mathbf{s})| \geqslant 4t + 1,$$

which implies the desired result.

Suppose that i is the smallest integer such that either $s_i \neq v_i$ or $s_{-i} \neq v_{-i}$. Since the first and last $4t+1$ bits of each codestring are identical and since every Catalan string begins with a 0 and ends with a 1, we have $i \geqslant 4t+3$.

Next, assume that $s_{-i} \neq v_{-i}, s_i = v_i$. The cases $s_i \neq v_i, s_{-i} = v_{-i}$ and $s_i \neq v_i, s_{-i} \neq v_{-i}$ can be proven similarly by considering the reversals of the strings $\mathbf{s}$ and $\mathbf{v}$.

Consider the compositions of the following two substrings:

$$\mathbf{s}_1^{-i-1} = s_1\, s_2\, \ldots s_{-i-1},$$
$$\mathbf{v}_1^{-i-1} = v_1\, v_2\, \ldots v_{-i-1}.$$

We claim that $\mathrm{wt}(\mathbf{s}_1^{-i-1}) \neq \mathrm{wt}(\mathbf{v}_1^{-i-1})$, which implies $\mathrm{c}(\mathbf{s}_1^{-i-1}) \neq \mathrm{c}(\mathbf{v}_1^{-i-1})$. This follows from the Catalan constraint, which ensures that $\mathrm{wt}(\mathbf{s}) = \mathrm{wt}(\mathbf{v})$, the assumptions that $s_{-i} \neq v_{-i}$, $\mathbf{s}_{-i+1}^{-1} = \mathbf{v}_{-i+1}^{-1}$, $\mathbf{s}_1^{i-1} = \mathbf{v}_1^{i-1}$, and from the choice of i.

As a result, we have

$$\mathrm{wt}(\mathbf{s}_i^{-i}) = \mathrm{wt}(\mathbf{v}_i^{-i}).$$

Next, we establish that $c(\mathbf{s}_1^{-i-1}) \in C(\mathbf{v}) \triangle C(\mathbf{s})$. For any $1 < j \leqslant i+1$, we have the following equality that holds for substrings of $\mathbf{s}$ of length $n-i$:

$$\mathrm{wt}(\mathbf{s}_j^{-(i-j+2)}) = \mathrm{wt}(\mathbf{s}_j^{i-1}) + \mathrm{wt}(\mathbf{s}_i^{-i}) + \mathrm{wt}(\mathbf{s}_{-i+1}^{-(i-j+2)}).$$

To prove this result, we consider the strings of length $n-i$ that are in the symmetric difference $C(\mathbf{v}) \triangle C(\mathbf{s})$. In particular, we consider the following three cases:

1. $j \leqslant i-1$,
2. $j = i$,
3. $j = i+1$.

Clearly, for the first case it holds that

$$\mathrm{wt}(\mathbf{s}_j^{-(i-j+2)}) = \mathrm{wt}(\mathbf{v}_j^{-(i-j+2)}).$$

For the second case, due to the constraints that $s_{4t+2}\, s_{4t+3}\, \ldots\, s_{-4t-2} \in \mathcal{P}(n-$

$2(4t+1))$, $s_1 \ldots s_{4t+1} = 00 \ldots 0$ and $s_{-4t-1}\, s_{-4t} \ldots s_{-1} = 11 \ldots 1$, it follows that $\mathbf{s}_1^{-i-1}$ contains more 0s than 1s, but $\mathbf{v}_i^{-2}$ contains more 1s than 0s. A similar argument may be used for the third case, and it can be shown in this case that $\mathbf{v}_{i+1}^{-1}$ also contains more 1s than 0s, which implies that $c(\mathbf{s}_1^{-i-1}) \in C(\mathbf{v}) \triangle C(\mathbf{s})$, as desired. In other words, we consider substrings of length $n-i$ (because $\mathbf{s}_1^{-i-1}$ has length $n-i$), of the form $\mathbf{s}_j^{-(i-j+2)}$. For the case where $1 < j \leqslant i-1$, the substrings of length $n-i$ in $\mathbf{v}$ and $\mathbf{s}$ have the same compositions, since $\mathrm{wt}(\mathbf{s}_j^{-(i-j+2)}) = \mathrm{wt}(\mathbf{v}_j^{-(i-j+2)})$. Thus, these substrings do not affect the compositions in $C(\mathbf{v})\Delta C(\mathbf{s})$. This covers the first case, Case 1. Hence it remains to show that $c(\mathbf{s}_1^{-i-1}) \neq c(\mathbf{v}_j^{-i-j+2})$ for Cases 2 and 3 (when $i = j$ and $i = j+1$). For the case $i = j$, we have $c(\mathbf{s}_1^{-(i-1)}) \neq c(\mathbf{v}_i^{-2})$, since $\mathbf{s}_1^{-(i-1)}$ has more 0s than 1s, whereas $\mathbf{v}_i^{-2}$ has more 1s than 0s. For the case $j = i+1$, $c(\mathbf{v}_{i+1}^{-1})$ also has more 1s than 0s. This completes the claim that for $l = 1$, $c(\mathbf{s}_l^{-i-1}) = c(\mathbf{s}_1^{-i-1}) \in C(\mathbf{v})\Delta C(\mathbf{s})$. The case $l \geqslant 2$ can be analyzed similarly.

Based on the discussion above, it is straightforward to identify additional substrings whose compositions lie in the symmetric difference of $C(\mathbf{s})$ and $C(\mathbf{v})$. In particular, if we can show that for every $l \in \{2, 3, 4, \ldots, 4t+1\}$ one of the following two claims is true:

1. $c(\mathbf{s}_l^{-i-1}) \in C(\mathbf{v}) \triangle C(\mathbf{s})$, or
2. $c(\mathbf{v}_l^{-i-1}) \in C(\mathbf{v}) \triangle C(\mathbf{s})$,

then $|C(\mathbf{s}) \triangle C(\mathbf{v})| \geqslant 4t+1$.

For $l \in \{2, 3, 4, \ldots, 4t+1\}$, it is straightforward to see that

$$\mathrm{wt}(\mathbf{s}_l^{-i-1}) \neq \mathrm{wt}(\mathbf{v}_l^{-i-1}).$$

Without loss of generality, we may assume that $\mathrm{wt}(\mathbf{s}_l^{-i-1}) < \mathrm{wt}(\mathbf{v}_l^{-i-1})$. Then $c(\mathbf{s}_l^{-i-1}) \in C(\mathbf{v})\triangle C(\mathbf{s})$. Similarly as before, for any $l < j \leqslant i+l$, the following holds for substrings of $\mathbf{s}$ of length $n-i-l+1$:

$$\mathrm{wt}(\mathbf{s}_j^{-(i-j+l+1)}) = \mathrm{wt}(\mathbf{s}_j^{i-1}) + \mathrm{wt}(\mathbf{s}_i^{-i}) + \mathrm{wt}(\mathbf{s}_{-i+1}^{-(i-j+l+1)}).$$

If $j \leqslant i-1$, we have

$$\begin{aligned}\mathrm{wt}(\mathbf{s}_j^{-(i-j+l+1)}) =& \mathrm{wt}(\mathbf{s}_j^{i-1}) + \mathrm{wt}(\mathbf{s}_i^{-i}) + \mathrm{wt}(\mathbf{s}_{-i+1}^{-(i-j+l+1)}) \\ =& \mathrm{wt}(\mathbf{v}_j^{i-1}) + \mathrm{wt}(\mathbf{v}_i^{-i}) + \mathrm{wt}(\mathbf{v}_{-i+1}^{-(i-j+l+1)}) \\ =& \mathrm{wt}(\mathbf{v}_j^{-(i-j+l+1)}).\end{aligned}$$

For the case $j \geqslant i \geqslant 4t+3$, note that $\mathbf{s}_l^{-i-1}$ contains more zeros than ones but for $j > i-1$, the substring $\mathbf{v}_j^{-(i-j+l+1)}$ contains at least as many 1s as 0s. Therefore, for any $j > i-1$,

$$c(\mathbf{s}_l^{-i-1}) \neq c(\mathbf{v}_j^{-(i-j+l+1)}).$$

We are left with analyzing the compositions of substrings of length $n-i-l+1$ in $\mathbf{v}$ to the left of $\mathbf{v}_l^{-i-1}$. Since every codestring in $\mathcal{C}(n,t)$ starts with $4t+1$ 0s, it follows that for any $j < l$

$$\mathrm{wt}(\mathbf{v}_j^{-(i-j+l+1)}) \leqslant \mathrm{wt}(\mathbf{v}_{j-1}^{-(i-(j-1)+l+1)}).$$

Furthermore, since $\mathrm{wt}(\mathbf{s}_l^{-i-1}) < \mathrm{wt}(\mathbf{v}_l^{-i-1})$, it follows that for any $j < l$,

$$\mathrm{wt}(\mathbf{s}_l^{-i-1}) < \mathrm{wt}(\mathbf{v}_j^{-(i-j+l+1)}).$$

Thus, $c(\mathbf{s}_l^{-i-1}) \in C(\mathbf{v}) \triangle C(\mathbf{s})$. This completes the proof. □

The result of Theorem 4 may be used to prove Theorem 5 since the number of redundant bits, $\mathcal{O}(\log k + t)$, is a direct consequence of the code construction described in Equation (2.11).

The reconstruction time for the described codes for a constant number of errors t is polynomial in n. To see this, consider the $\binom{\binom{n+1}{2}}{t}$ possible choices for errors in distinct compositions. Each composition can be corrupted in at most n different ways (for the composition corresponding to the whole string this number equals n). Thus, given an erroneous composition multiset $\tilde{C}(\mathbf{s})$, there are at most $\binom{\binom{n+1}{2}}{t} n^t$ candidate true composition multisets $\{\tilde{C}^1(\mathbf{s}), \tilde{C}^2(\mathbf{s}), \ldots \tilde{C}^m(\mathbf{s})\}$, where $m = \mathcal{O}(n^{3t})$. Thus, by reconstructing the strings as given by the compositions $\{\tilde{C}^1(\mathbf{s}), \tilde{C}^2(\mathbf{s}), \ldots \tilde{C}^m(\mathbf{s})\}$ using the Backtracking algorithm, we can recover the string $\mathbf{s}$ in $\mathcal{O}(n^{3+3t})$ time.

2.6 Open Problems

Many combinatorial and coding-theoretic problems related to mass error-correcting codes remain open and are listed below.

- In Sections 2.3, 2.4 and 2.5 we showed that the number of redundant bits sufficient for unique and efficient reconstruction without errors and in the presence of a constant number of t errors equals $\mathcal{O}(\log k)$ and $\mathcal{O}(t^2 \log k)$, respectively. Lower bounds on the number of redundant bits are still unknown.

- The decoding algorithm used in the proof of Theorem 3 is efficient only if the number of errors, t is a constant. We are unaware of string reconstruction algorithms that are efficient both in t and n.

- We addressed the string reconstruction problem when the errors are either asymmetric or symmetric. However, MS/MS errors are often bursty and context-dependent. Thus, studying other error models is of interest.

- Several problems outlined in [7] also remain open. We restate two of those problems for completeness: (1) Improve the upper and lower bounds on the number of *confusable* strings. (2) Determine explicit polynomial-time algorithm for string reconstruction problems, the existence of which was established in [32–35].

CHAPTER 3

MULTIPLE CODED STRING RECONSTRUCTION

3.1 Introduction

Modern digital data storage systems are facing fundamental storage density limits and to address the emerging needs for large volume archiving, it is of importance to identify new nanoscale recording media. Recently proposed DNA-based data storage paradigms [2–6, 36–38] offer storage densities that are orders of magnitude higher than those of flash and optical recorders but the systems often come with a prohibitively high cost and slow and error-prone read/write platforms. To mitigate the issues associated with potentially ambiguous data reconstruction and to correct a diverse type of errors inherent to DNA sequencing technologies, several new coding solutions that aid in string assembly, dealing with asymmetries in the readout channel, and reconciliation of multiple string evidence sets were introduced in [25, 28, 39–43] (see also the related and follow-up lines of work [44–49]).

As an alternative to DNA-based data storage systems, polymer-based data storage systems [2, 36] are particularly attractive due to their low cost [2]. In such platforms, two molecules of significantly different masses are synthesized to represent the bits 0 and 1, respectively. The molecules are used as building blocks in the sequential process of recording user-defined information content. The obtained synthetic polymers are read by tandem mass (MS/MS) spectrometers. A mass spectrometer breaks multiple copies of the polymer uniformly at random, thereby creating prefixes and suffixes of the string of various lengths. The readout system outputs masses of these prefixes and suffixes. If the masses of all prefixes from a single string are accounted for and error-free, reconstruction is straightforward. But if multiple strings are read simultaneously and the masses of prefixes and suffixes of the same length are confusable, the problem becomes significantly more complicated.

It is currently not known which combinations of coded binary strings can be distinguished from each other based on prefix-suffix masses and for which code rates is it possible to perform unique multistring reconstruction.

In a related research direction, the problem of reconstructing a string from an abstraction of its MS/MS output was considered in [7], under the name *string reconstruction from its substring composition multiset*. The *composition* of a binary string is the number of 0s and the number of 1s in the string. For example, the composition of 001 equals $0^2 1^1$, indicating that 001 contains two 0s and one 1, without revealing the order of the bits. The substring composition multiset $C(\mathbf{s})$ of a string $\mathbf{s}$ is the multiset of compositions of all possible substrings of the string $\mathbf{s}$. As an illustration, the set of all substrings of 001 equals $\{0, 0, 1, 00, 01, 001\}$, and the substring composition multiset of 001 equals $\{0^1, 0^1, 1^1, 0^2, 0^1 1^1, 0^2 1^1\}$. Two modeling assumptions are used for the purpose of rigorous mathematical analysis of this problem [7] and in subsequent works [50–52]: (a) Using MS/MS measurements, one can uniquely infer the composition of a polymer substring from its mass. (b) When a polymer is broken down for mass spectrometry analysis, the masses of all its substrings are observed with identical frequencies.

Under the above modeling assumptions, the authors of [7] established that strings are uniquely reconstructable up to reversal provided that the length of the strings n is one less than a prime or one less than twice a prime, or whenever $n \leqslant 7$. The work [50–52] demonstrated that at most logarithmic code redundancy can ensure unique reconstruction of single strings drawn from codebooks based on Bertrand-Catalan strings or Reed-Solomon-like constructions.

However, the assumption that MS/MS output measurements include masses of all substrings is often not true in practice, as breaking the string in one rather than two locations is easier to perform. In the former case, one is presented with masses of the prefixes and suffixes. Thus, for the string 001, one would observe the multiset $\{0^1, \cancel{0^1}, 1^1, 0^2, 0^1 1^1, 0^2 1^1\}$. Furthermore, in practice the contents of multiple strings are often read simultaneously, which complicates the matter even further as it is not known a priori which prefixes and suffixes are associated with a given string.

The problem addressed in this work may be formally stated as follows. We seek the size of the largest code $C(h)$ of binary strings of a fixed length n with a property we refer to as *h-unique reconstructability*. For any subcol-

lection $\mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_{\bar{h}}$ of $\bar{h} \leqslant h$ strings from $C(h)$, one is presented with the union $\mathcal{M}(\mathbf{s}_1) \cup \mathcal{M}(\mathbf{s}_2) \cup \cdots \cup \mathcal{M}(\mathbf{s}_{\bar{h}})$ of the prefix-suffix composition multisets, $\mathcal{M}(\mathbf{s}_i)$, $i = 1, \ldots, \bar{h}$, of the individual strings $\mathbf{s}_i$, $i = 1, \ldots, h$. The prefix-suffix composition multiset $\mathcal{M}(\mathbf{s})$ of a string $\mathbf{s}$ captures the weights of prefixes and suffixes of the string $\mathbf{s}$ of all lengths. Unique reconstruction refers to the property of being able to distinguish all possible h'-unions and unambiguously determine the identity of the strings in the collection. Our main result provides a construction for $C(h)$ that asymptotically approaches $1/h$, under certain mild parameter constraints. The proofs of our results rely on the use of Dyck and binary B_h strings. For the latter, only constructions and bounds pertaining to $h = 2$ and $h = n$ have been known in the literature [53–55], while we provide new results for arbitrary even values of h. We also introduce a simple scheme for combating missing prefix-suffix errors in the pool and motivate the study of a number of new error-control coding problems associated with mixture reconstructions.

This chapter is organized as follows. Section 3.3 introduces the problem, as well as the relevant terminology and notation. Section 3.4 describes the code constructions and the corresponding lower-bound analysis for the code rate. Upper bounds are presented in Section 3.5. Error-control coding schemes are described in Section 3.6, along with open problems.

3.2 A Short Note on Single String Reconstruction

Before we get to the essence of this chapter, we shall briefly discuss single string reconstruction from the prefix-suffix composition multiset.

3.2.1 Reconstruction Codes

Consider the problem of string reconstruction from prefix-suffix composition multiset. At iteration i, the backtracking algorithm employed therein first identified the longest prefix-suffix pair not in the set $\mathcal{T}_i$. However, for the problem of single string reconstruction from the prefix-suffix composition multiset, one readily has access to the same. Thus, the backtracking algorithm as described in the previous chapter can be employed here as well. Furthermore, the backtracking algorithm guesses only when the weight of

the reconstructed prefix is the same as that of the suffix. Thus, the following codebook $\mathcal{S}_{R,p-s}(n)$ is a single string prefix-suffix reconstruction code as well.

For even n,

$$\begin{aligned}\mathcal{S}_{R,p-s}(n) =&\{\mathbf{s} \in \{0,1\}^n, s_1 = 0, s_n = 1, \text{ and} \\ &\exists\, I \subseteq \{1,2,\ldots,n-1,n\} \text{ such that} \\ &\qquad\qquad \text{for all } i \in I, s_i \neq s_{n+1-i}, \\ &\qquad\qquad \text{for all } i \notin I, s_i = s_{n+1-i}, \\ &\mathbf{s}_{[\frac{n}{2}]\cap I} \text{ is a Catalan-Bertrand string}\}.\end{aligned} \tag{3.1}$$

For odd n,

$$\mathcal{S}_{R,p-s}(n) = \cup_{\mathbf{s}\in\mathcal{S}_{R,p-s}(n-1)}\{\mathbf{s}_1^{\frac{n-1}{2}}\ 0\ \mathbf{s}_{\frac{n+1}{2}}^{n-1},\ \mathbf{s}_1^{\frac{n-1}{2}}\ 1\ \mathbf{s}_{\frac{n+1}{2}}^{n-1}\}.$$

Claim 8. *There exists single string prefix-suffix reconstruction codes of length n with redundancy $\mathcal{O}(\log n)$.*

3.2.2 Error-Correction Codes

Perhaps the simplest of errors are that of erasures, as the location of the errors can be inferred. Consider a string $\mathbf{s} \in \mathcal{S}_{R,p-s}(n)$. Clearly the set of prefix (and correspondingly its suffix) compositions can be separated from the prefix-suffix composition multiset.

Suppose a single prefix composition multiset is erased. Clearly from the set of the suffix compositions, the string can be reconstructed. Thus, due to inherent redundancy present in the evidence set one need not encode redundancy to correct a single erasure. In the same spirit, if at most t erasures occur in the prefix-suffix composition multiset, it suffices to ensure that prefix and suffix composition multisets are such that even in the presence of at most $\lfloor \frac{t}{2} \rfloor$ erasures, one can reconstruct the string.

Consider the following coding technique:

- Given a string $\mathbf{s} \in \{0,1\}^n$, construct $I(\mathbf{s}) = s_1(s_1+s_2)(s_1+s_2+s_3)\ldots(s_1+s_2+\cdots+s_n) = I(\mathbf{s})_1 I(\mathbf{s})_2 \ldots I(\mathbf{s})_n$, where the addition is over $\mathbb{F}_2$.

- Encode $I(\mathbf{s})$ using a BCH code such that the resulting string $I(\mathbf{s})R'(\mathbf{s})$,

where $R'(\mathbf{s})$ denotes the redundancy bits, is of length $m-1$ and can correct $\lfloor \frac{t}{2} \rfloor$ erasures.

- Let $R(\mathbf{s})_1 = R'(\mathbf{s})_1 + I(\mathbf{s})_1$; $R(\mathbf{s})_{2i} = \bar{R(\mathbf{s})}_{2i} = 1 - R(\mathbf{s})_{2i}$, and $R(\mathbf{s})_{2i+1} = R'(\mathbf{s})_i + R'(\mathbf{s})_{i+1} + R(\mathbf{s})_{2i}$, for all $i \in [m-n-1]$. $R(\mathbf{s})_m = R(\mathbf{s})_1 + R(\mathbf{s})_2 + \cdots + R(\mathbf{s})_{m-1}$.
- Encode $\mathbf{s}$ as $R(\mathbf{s})\mathbf{s}R(\mathbf{s})$.

Such a code can correct up to t erasures in the prefix-suffix composition multiset.

Claim 9. *There exist single string prefix-suffix reconstruction and t-erasure error correction codes of length n with redundancy $\mathcal{O}(t \log n)$.*

We now consider substitution errors. Consider a composition $0^z 1^w$ corresponding to a substring (prefix or substring) of length i. A substitution error occurs if $0^z 1^w$ is modified as $0^{z'} 1^{w'}$ such that $z + w = z' + w'$. Now, consider the case where at most t composition substitution errors are allowed to occur in the prefix-suffix composition multiset. Although it is true that either the prefix or the suffix composition set has at most $\lfloor \frac{t}{2} \rfloor$ errors, it is not always possible to discern the set with the fewer errors. Thus, the following simple modification to the scheme described above corrects for t composition errors.

Consider the following coding technique:

- Given a string $\mathbf{s} \in \{0,1\}^n$, construct $I(\mathbf{s}) = s_1(s_1+s_2)(s_1+s_2+s_3)\ldots(s_1+s_2+\cdots+s_n) = I(\mathbf{s})_1 I(\mathbf{s})_2 \ldots I(\mathbf{s})_n$, where the addition is over $\mathbb{F}_2$.
- Encode $I(\mathbf{s})$ using a BCH code such that the resulting string $I(\mathbf{s})R'(\mathbf{s})$, where $R'(\mathbf{s})$ denotes the redundancy bits, is of length $m-1$ and can correct t substitution errors.
- Let $R(\mathbf{s})_1 = R'(\mathbf{s})_1 + I(\mathbf{s})_1$; $R(\mathbf{s})_{2i} = \bar{R(\mathbf{s})}_{2i} = 1 - R(\mathbf{s})_{2i}$, and $R(\mathbf{s})_{2i+1} = R'(\mathbf{s})_i + R'(\mathbf{s})_{i+1} + R(\mathbf{s})_{2i}$, for all $i \in [m-n-1]$. $R(\mathbf{s})_m = R(\mathbf{s})_1 + R(\mathbf{s})_2 + \cdots + R(\mathbf{s})_{m-1}$.
- Encode $\mathbf{s}$ as $R(\mathbf{s})\mathbf{s}R(\mathbf{s})$.

Claim 10. *There exist single string prefix-suffix reconstruction and t-substitution error correction codes of length n with redundancy $\mathcal{O}(t \log n)$.*

3.3 Problem Statement and Preliminaries

We start by introducing the relevant notation. Let $\mathbf{s} = s_1 \dots s_n \in \{0,1\}^n$ be a binary string of length n and let $\mathcal{M}(\mathbf{s})$ denote the composition multiset of all prefixes and suffixes of $\mathbf{s}$. For example, if $\mathbf{s} = 01101$, then

$$\mathcal{M}(\mathbf{s}) = \Big\{0, 01, 01^2, 0^2 1^2, 0^2 1^3, 1, 01, 01^2, 01^3, 0^2 1^3\Big\}.$$

We denote the set of prefix and suffix compositions of $\mathbf{s}$ as $\mathcal{M}_p(\mathbf{s})$ and $\mathcal{M}_s(\mathbf{s})$, respectively. For the above string, $\mathcal{M}_p(\mathbf{s}) = \{0, 01, 01^2, 0^2 1^2, 0^2 1^3\}$ and $\mathcal{M}_s(\mathbf{s}) = \{1, 01, 01^2, 01^3, 0^2 1^3\}$.

We seek to design a binary codebook $C(n,h) \subseteq \{0,1\}^n$ so that for any collection of distinct strings $\mathbf{s}_1, \mathbf{s}_2, \dots, \mathbf{s}_{\bar{h}} \in C(n,h)$ with $\bar{h} \leqslant h$, the multiset

$$\mathcal{M}(\mathbf{s}_1) \cup \mathcal{M}(\mathbf{s}_2) \cup \dots \cup \mathcal{M}(\mathbf{s}_{\bar{h}}) \tag{3.2}$$

uniquely determines the individual strings in the collection. We refer to a code that satisfies (3.2) as an ***h*-multicomposition code**, or an ***h*-MC code**. For simplicity, we often use $\mathcal{M}(S)$ to describe the multi-composition set for $S = \{\mathbf{s}_1, \mathbf{s}_2, \dots, \mathbf{s}_h\}$. We also say that $C_p(n,h) \subseteq \{0,1\}^n$ is an h-prefix code if for any two distinct sets of size $\leqslant h$, say $S_1, S_2 \subseteq C_p$, $\mathcal{M}_p(S_1) \neq \mathcal{M}_p(S_2)$.

The next claim establishes a useful connection between our problem and the related problem of determining binary strings based on their real-valued sum.

Claim 11. *Given $\mathcal{M}_p(\mathbf{s}_1) \cup \mathcal{M}_p(\mathbf{s}_2) \cup \dots \cup \mathcal{M}_p(\mathbf{s}_h)$, one can determine the real-valued sum $\mathbf{s}_1 + \mathbf{s}_2 + \dots + \mathbf{s}_h$.*

Proof. We prove the result for $h = 2$ as the generalization is straightforward. Suppose that $\mathbf{s}_1, \mathbf{s}_2 \in \{0,1\}^n$. Then, given $\mathcal{M}_p(\mathbf{s}_1) \cup \mathcal{M}_p(\mathbf{s}_2)$, let n_i denote the total number of ones in the two compositions of prefixes of length i in the multiset (i.e., sum of their weights). It is straightforward to see that $\mathbf{s}_1 + \mathbf{s}_2 = t_1 t_2 \dots t_n$, where $t_i = n_i - n_{i-1}$, with $n_0 = 0$. □

Example 4. *As an illustrative example, consider the strings $\mathbf{s}_1 = 110100$ and $\mathbf{s}_2 = 101010$, for which we have $\mathbf{s}_1 + \mathbf{s}_2 = 211110$. Clearly, $(\mathbf{s}_1 + \mathbf{s}_2)_1 = 2$, which we obtained by summing the compositions of prefixes of length one,*

i.e., $1+1=2$. *Next, it is easy to see that* $(\mathbf{s}_1+\mathbf{s}_2)_2 = (\mathbf{s}_1+\mathbf{s}_2)_1^2 - (\mathbf{s}_1+\mathbf{s}_2)_1$, *where for simplicity of notation we used* $(\mathbf{s}_1+\mathbf{s}_2)_1^2$ *to denote the sum of the weights of the prefixes of length two. A straightforward calculation reveals that* $(\mathbf{s}_1+\mathbf{s}_2)_2 = (2+1)-2 = 1$. *The other positions in the sum of the strings can be determined similarly.* □

The above claim provides a useful connection between our problem and the problem of designing binary B_h sequences. A *binary* B_h *sequence* is a set $\mathcal{S}_h(n)$ of binary strings of fixed length n such that for any two distinct subsets of strings in $\mathcal{S}_h(n)$, say $\mathcal{S} = \{\mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_{\bar{h}_1}\} \neq \mathcal{T} = \{\mathbf{t}_1, \mathbf{t}_2, \ldots, \mathbf{t}_{\bar{h}_2}\}$, where $\bar{h}_1, \bar{h}_2 \leqslant h$, one has

$$\sum_{i=1}^{\bar{h}_1} \mathbf{s}_i \neq \sum_{j=1}^{\bar{h}_2} \mathbf{t}_j. \tag{3.3}$$

Here, addition is performed over the reals. To avoid possible confusion with the naming convention involving sequences of sequences, we henceforth refer to the above entity as collection of binary B_h sequences or binary B_h codes of length n.

Example 5. *Consider the set* $\mathcal{S}_2(6) = \{110100, 101010, 110010\}$. *It is easy to verify that the real-valued sums of pairs of strings in* $\mathcal{S}_2(6)$ *are distinct. Thus,* $\mathcal{S}_2(6)$ *is a binary* B_2 *code.*

However, $\mathcal{S}_2'(6) = \{110100, 101010, 110010, 101100\}$ *is not a binary* B_2 *code since* $110100 + 101010 = 110010 + 101100 = 211110$. □

Based on Claim 11, it is easy to identify two sufficient conditions for a set of strings to be an h-MC code:

1. **Condition 1:** One can recover $\mathcal{M}_p(\mathbf{s}_1) \cup \cdots \cup \mathcal{M}_p(\mathbf{s}_h)$ from $\mathcal{M}(\mathbf{s}_1) \cup \cdots \cup \mathcal{M}(\mathbf{s}_h)$, for any choice of h distinct codestrings $\mathbf{s}_1, \ldots, \mathbf{s}_h$; and

2. **Condition 2:** The codestrings $\mathbf{s}_1, \ldots, \mathbf{s}_h$ belong to a binary B_h code $\mathcal{S}_h(n)$.

These observations will be used to construct h-MC codes in Section 3.4. Note that the condition that the codestrings in an MC code belong to a B_h-code is not necessary. For example, consider the case $\mathbf{s}_1 = 011$, $\mathbf{s}_2 = 000$, $\mathbf{s}_3 = 001$, $\mathbf{s}_4 = 010$. Then, $\mathbf{s}_1 + \mathbf{s}_2 = 011 = \mathbf{s}_3 + \mathbf{s}_4$, but $01^2 \in \mathcal{M}(\mathbf{s}_1) \cup \mathcal{M}(\mathbf{s}_2)$

and $01^2 \notin \mathcal{M}(\mathbf{s}_3) \cup \mathcal{M}(\mathbf{s}_4)$, so that $\{\mathbf{s}_1, \mathbf{s}_2\}$ and $\{\mathbf{s}_3, \mathbf{s}_4\}$ are not confusable. However, a direct consequence of Claim 11 is that the maximum size of a B_h code is at most the maximum size of a h-prefix code.

The best currently known upper bound on the rate of binary B_2 codes is .5753 [56]. For $h > 2$ such that $h \neq n$, we are unaware of any other bounds on the rate of binary B_h codes other the ones presented in this work. We show next that for sufficiently large code lengths, the maximum rate of an h-MC code is at least $\frac{1}{h}$.

3.4 A Constructive Lower Bound for h-MC Codes

We start with a binary B_h code and introduce redundancy into the underlying strings to ensure that given the multi-composition set of at most h strings, one can separate the prefixes from the suffixes. Then, given the set of prefixes, one can use the same idea in Claim 11 to recover the sum of the h codestrings and hence the codestrings themselves.

Let $\mathcal{S}_h(n) \subseteq \mathbb{F}_2^n$ be a B_h code over $\mathbb{F}_2^n$. It is well known that $\mathcal{S}_h(n)$ can be constructed using the columns of a parity-check matrix of a code with minimum Hamming distance $\geqslant 2h+1$. Using this construction, we have

$$\lim_{n\to\infty} \frac{1}{n} \log |\mathcal{S}_h(n)| = \frac{1}{h}.$$

For our problem and the underlying approach for solving it, we will also make use of Dyck strings: A string $\mathbf{s} \in \mathbb{F}_2^N$ of even length N is a Dyck string if its weight satisfies $\mathrm{wt}(\mathbf{s}) = \frac{N}{2}$, and for $i \in [N-1]$,

$$\mathrm{wt}(s_1 s_2 \ldots s_i) \geqslant \left\lceil \frac{i}{2} \right\rceil. \tag{3.4}$$

The approach for generating the code $C(N,h) \subseteq \mathbb{F}_2^N$ is to ensure that it satisfies the following two properties:

1. A string $\mathbf{s} \in C(N,h)$ is a Dyck string;

2. The set $C(N,h)$ is a binary B_h code of length N.

The first property ensures that the mixtures of prefixes and suffixes can be partitioned into two sets, one containing all the prefixes and another

containing all the suffixes. The second property ensures that given the prefix set (or, alternatively, the suffix set) one can recover the codestrings using the simple observation that the prefixes uniquely determine the real-valued sum of the strings in the mixture. We illustrate these observations with an example.

Example 6. *Consider the binary B_2 code $\mathcal{S}_2(6) = \{110100, 101010, 110010\}$. Clearly, all three strings are Dyck strings as their prefixes of any length contain at least as many ones as zeros.*

Next, write $\mathbf{s}_1 = 110100$ and $\mathbf{s}_2 = 101010$, so that $\mathcal{M}(\mathbf{s}_1) \cup \mathcal{M}(\mathbf{s}_2) = \{1, 1, 01,\ 1^2,\ 01^2,\ 01^2, 0^2 1^2,\ 01^3, 0^2 1^3, 0^2 1^3,\ 0^3 1^3, 0^3 1^3,\ 0^3 1^3,\ 0^3 1^3,\ 0^3 1^2,\ 0^3 1^2, 0^2 1^2,\ 0^3 1, 0^2 1, 0^2 1,\ 01, 0^2, 0, 0\}$. Since $\mathbf{s}_1$ and $\mathbf{s}_2$ are Dyck strings, each of the string prefixes must have at least as many 1s as 0s. Similarly, each suffix must have at least as many 0s as 1s. It follows from this observation that one can easily recover the multiset $\mathcal{M}_p(\mathbf{s}_1) \cup \mathcal{M}_p(\mathbf{s}_2) = \{1, 1, 01,\ 1^2, 01^2,\ 01^2, 0^2 1^2,\ 01^3, 0^2 1^3, 0^2 1^3,\ 0^3 1^3, 0^3 1^3\}$.

Claim 11 ensures that given $\mathcal{M}_p(\mathbf{s}_1) \cup \mathcal{M}_p(\mathbf{s}_2)$, one can determine $\mathbf{s}_1 + \mathbf{s}_2 = 211110$. Since $\mathcal{S}_2(6)$ is a binary B_2 code, the sum $\mathbf{s}_1 + \mathbf{s}_2$ uniquely determines the strings $\mathbf{s}_1$ and $\mathbf{s}_2$. □

The next claim establishes the formal result that if the code $C(N, h)$ satisfies these two properties, then it is an h-MC code.

Claim 12. *Suppose that $C(N, h)$ is a B_h code where for any $\mathbf{s} \in C(N, h)$, Equation (3.4) holds. Then, $C(N, h)$ is an h-MC code.*

Proof. Similar to Claim 11, we prove the statement for the case where $h = 2$, since the extension for general h is straightforward. In light of Claim 11, we need to show that the property in (3.4) allows us to uniquely recover $\mathcal{M}_p(\mathbf{s}_1) \cup \mathcal{M}_p(\mathbf{s}_2)$ from $\mathcal{M}(\mathbf{s}_1) \cup \mathcal{M}(\mathbf{s}_2)$. To see that this is indeed possible, observe that from (3.4) both prefixes of length i in $\mathcal{M}(\mathbf{s}_1) \cup \mathcal{M}(\mathbf{s}_2)$ have at least $\lceil \frac{i}{2} \rceil$ 1s whereas both suffixes of length i in $\mathcal{M}(\mathbf{s}_1) \cup \mathcal{M}(\mathbf{s}_2)$ have at most $\lfloor \frac{i}{2} \rfloor$ 1s. □

To construct codes $C(N, h)$ of large cardinality, we perform a simple "balancing procedure" on each codestring $\mathbf{s} \in \mathcal{S}_h(n)$ and then append $\mathcal{O}(\sqrt{n})$ bits of redundancy to the beginning and end of $\mathbf{s}$ so that the resulting string

has length $N = n + \mathcal{O}(\sqrt{n})$. Note that under this setup, it follows that for any ϵ, we have

$$\frac{1}{N}\log|C(N,h)| = \frac{1}{n+\kappa\sqrt{n}}\log|\mathcal{S}_h(n)| = \frac{1}{h} - \epsilon,$$

where κ is a constant, and $\epsilon > 0$ can be made arbitrarily small for n sufficiently large.

To maximize the rate of the coding scheme and combine the two constraints that h-**MC** strings need to satisfy, we use two ideas. First, we use B_h strings parsed into blocks that allow us to tightly control the weights of the code-strings. Second, rather than working directly with the weights of strings as described in (3.4), we use the running digital sums (RDSs). For a string $\mathbf{s} \in \mathbb{F}_2^n$, the RDS up to coordinate i is defined as $R(\mathbf{s})_i = 2\mathrm{wt}(s_1 s_2 \dots s_i) - i$. If the subscript i is omitted, then $R(\mathbf{s}) = 2\mathrm{wt}(\mathbf{s}) - |\mathbf{s}|$, where $|\mathbf{s}|$ denotes the length of $\mathbf{s}$. Using the running digital sum, the constraint in Equation (3.4) can be rewritten as $\mathrm{wt}(\mathbf{s}) = \lceil \frac{N}{2} \rceil$ and $R(\mathbf{s})_i \geqslant 0$, $i \in [N]$.

The balancing procedure operates as follows: Let $\mathbf{s} \in \mathcal{S}_h(n)$, and for simplicity, assume that $\sqrt{n}$ is an even integer. We begin by parsing $\mathbf{s}$ into blocks $\mathbf{s}_i$ of length $\sqrt{n}$, $i = 1, \dots, \sqrt{n}$, so that $\mathbf{s} = \mathbf{s}_1\mathbf{s}_2 \dots \mathbf{s}_{\sqrt{n}} \in \mathbb{F}_2^n$. Using $\mathbf{s}$ we construct an auxiliary string $\mathbf{u} = \mathbf{u}_1\mathbf{u}_2 \dots \mathbf{u}_{\sqrt{n}}$ that is "approximately" balanced following an idea similar to Knuth's balancing, which operates on blocks rather than individual symbols (please refer to Figures 3.1 and 3.2 for an illustration). We start by initializing $\mathbf{u}_1 = \mathbf{s}_1$. For a binary string $\mathbf{u}$, we use $\overline{\mathbf{u}}$ to denote the binary complement of $\mathbf{u}$. For $j \in \{2, 3, \dots, \sqrt{n}\}$, we define $\mathbf{u}_j$ according to:

$$\mathbf{u}_j = \begin{cases} \mathbf{s}_j, & \text{if } R(\mathbf{u}_1 \dots \mathbf{u}_{j-1}) < 0, \text{ and } R(\mathbf{s}_j) \geqslant 0, \\ \overline{\mathbf{s}}_j, & \text{if } R(\mathbf{u}_1 \dots \mathbf{u}_{j-1}) < 0, \text{ and } R(\mathbf{s}_j) < 0, \\ \mathbf{s}_j, & \text{if } R(\mathbf{u}_1 \dots \mathbf{u}_{j-1}) \geqslant 0, \text{ and } R(\mathbf{s}_j) < 0, \\ \overline{\mathbf{s}}_j, & \text{if } R(\mathbf{u}_1 \dots \mathbf{u}_{j-1}) \geqslant 0, \text{ and } R(\mathbf{s}_j) \geqslant 0. \end{cases} \tag{3.5}$$

The next claim immediately follows from (3.5).

Claim 13. *For any $j \in [\sqrt{n}]$, $|R(\mathbf{u}_1 \dots \mathbf{u}_j)| \leqslant \sqrt{n}$. Hence, the RDS of complete collections of subblocks is bounded in absolute value by $\sqrt{n}$.*

We also have the following result.

Lemma 7. *For any $i \in [n]$, $|R(\mathbf{u})_i| \leqslant \frac{3}{2}\sqrt{n}$. Hence, the RDS of any prefix of $\mathbf{u}$ does not exceed $\frac{3}{2}\sqrt{n}$ in absolute value.*

Proof. Suppose, on the contrary, that $|R(\mathbf{u})_i| > \frac{3\sqrt{n}}{2}$. For simplicity, we will only consider the case $R(\mathbf{u})_i > \frac{3\sqrt{n}}{2}$, as the other case can be handled similarly. Next, assume that $j \in [n]$ is the smallest index for which $R(\mathbf{u})_j > \frac{3\sqrt{n}}{2}$ and that we may assume that $R(\mathbf{u})_j = \frac{3\sqrt{n}}{2} + 1$. Now, let $j = k_1\sqrt{n} + k_2$, where $0 \leqslant k_2 < \sqrt{n}$. According to Claim 13, since $R(\mathbf{u})_j = \frac{3\sqrt{n}}{2} + 1$ and $k_2 < \sqrt{n}$, we have

$$\frac{\sqrt{n}}{2} < R(\mathbf{u}_1 \dots \mathbf{u}_{k_1-1}) \leqslant \sqrt{n}. \tag{3.6}$$

Based on (3.5), and since $R(\mathbf{u}_1 \dots \mathbf{u}_{k_1-1}) > \frac{\sqrt{n}}{2}$, it follows that $R(\mathbf{u}_{k_1}) \leqslant 0$ so that

$$-\sqrt{n} < R(\mathbf{u}_{k_1})_\ell \leqslant \frac{\sqrt{n}}{2} \tag{3.7}$$

for any $\ell \in [\sqrt{n}]$. Combining (3.6) and (3.7), we have that $R(\mathbf{u})_{k_1\sqrt{n}+k_2} \leqslant \frac{3\sqrt{n}}{2}$, which is a contradiction. □

We now describe our encoder. Let $\mathbf{u} \in \{0,1\}^n$ be the string which is the result of the procedure described in (3.5), and suppose that $\mathbf{r} \in \{0,1\}^{\sqrt{n}}$ is such that for any $j \in [\sqrt{n}]$:

$$\mathbf{r}_j = \begin{cases} 1, & \text{if } \mathbf{u}_j \neq \mathbf{s}_j, \\ 0, & \text{if } \mathbf{u}_j = \mathbf{s}_j. \end{cases} \tag{3.8}$$

Using $\mathbf{r}$, we now form a string $\mathbf{s} \in C(N,h)$, where $N = n + \frac{17}{2}\sqrt{n}$, and assume for simplicity that N is an even integer. The following claim is used in our subsequent analysis.

Claim 14. *Let $\mathbf{v} = \mathbf{1}^{5/2\sqrt{n}}\mathbf{r}\mathbf{u} \in \{0,1\}^{n+7/2\sqrt{n}}$. Then, for any $i \in [n + \frac{7}{2}\sqrt{n}]$,*

$$|R(\mathbf{v})_i| \leqslant 5\sqrt{n}.$$

Furthermore, for any $i \in [n + 7/2\sqrt{n}]$,

$$R(\mathbf{v})_i > 0.$$

We now append redundant bits to the string $\mathbf{v}$ described in Claim 14 in order to get a string $\mathbf{s} \in \{0,1\}^N$ which is a Dyck string.[1] This results in the following claim.

Claim 15. *Let $N = n + \frac{17}{2}\sqrt{n}$ be an even integer and let $\mathbf{v} = \mathbf{1}^{5/2\sqrt{n}}\mathbf{ru} \in \{0,1\}^{n+7/2\sqrt{n}}$ be as defined in Claim 14. Suppose that $w = wt(\mathbf{v})$. Then, the string $\mathbf{s} = \mathbf{v}\mathbf{1}^{\frac{N}{2}-w}\mathbf{0}^{\frac{N}{2}-(|\mathbf{v}|-w)}$ is a Dyck string.*

Now, assume that $C(N,h) \subseteq \mathbb{F}_2^N$ is constructed according to the procedure outlined in Claim 15 and once again assume that $N = n + \frac{17}{2}\sqrt{n}$ is an even integer. The next theorem is the main result of this section and it establishes the correctness of our construction through the description of a simple decoding algorithm.

Theorem 9. *Suppose that $\mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_h \in C(N,h)$, where $C(N,h)$ is constructed according to the balancing procedure operating on B_h strings. Then, given $\mathcal{M}(\mathbf{s}_1) \cup \mathcal{M}(\mathbf{s}_2) \cup \cdots \cup \mathcal{M}(\mathbf{s}_h)$, we can uniquely determine $\{\mathbf{s}_1, \ldots, \mathbf{s}_h\}$. Furthermore, for any $\epsilon > 0$, there exists $n_\epsilon > 0$ such that for all $N \geqslant n_\epsilon$, $\frac{1}{N}\log|C(N,h)| \geqslant \frac{1}{h} - \epsilon$.*

Proof. For simplicity, we prove the result for $h = 2$, as the extension for general values h is straightforward. According to Claims 12 and 15, we can recover $\mathcal{M}_p(\mathbf{s}_1) \cup \mathcal{M}_p(\mathbf{s}_2)$ from $\mathcal{M}(\mathbf{s}_1) \cup \mathcal{M}(\mathbf{s}_2)$ since $\mathbf{s}_1, \mathbf{s}_2$ are Dyck strings. From $\mathcal{M}_p(\mathbf{s}_1) \cup \mathcal{M}_p(\mathbf{s}_2)$, we can recover $\mathbf{s}_1 + \mathbf{s}_2$ according to Claim 11. Given $\mathbf{s}_1 = \mathbf{1}^{5/2\sqrt{n}}\mathbf{r}_1\mathbf{u}_1\mathbf{1}^{\frac{N}{2}-w_1}\mathbf{0}^{\frac{N}{2}-(|\mathbf{v}_1|-w_1)}$ and $\mathbf{s}_2 = \mathbf{1}^{5/2\sqrt{n}}\mathbf{r}_2\mathbf{u}_2\mathbf{1}^{\frac{N}{2}-w_2}\mathbf{0}^{\frac{N}{2}-(|\mathbf{v}_2|-w_2)}$, from the first $n + \frac{7}{2}\sqrt{n}$ coordinates of $\mathbf{s}_1 + \mathbf{s}_2$ we can recover

$$(\mathbf{r}_1 + \mathbf{r}_2, \mathbf{u}_1 + \mathbf{u}_2) \bmod 2.$$

Next, for shorthand, write $\mathbf{u} = \mathbf{u}_1 + \mathbf{u}_2 \bmod 2 = \mathbf{u}_1\mathbf{u}_2 \ldots \mathbf{u}_{\sqrt{n}}$ and $\mathbf{r} = \mathbf{r}_1 + \mathbf{r}_2 \bmod 2 = r_1 \ldots r_{\sqrt{n}}$. Let $\tilde{\mathbf{u}} = \tilde{\mathbf{u}}_1 \ldots \tilde{\mathbf{u}}_{\sqrt{n}}$. Then, for $j \in [\sqrt{n}]$,

$$\tilde{\mathbf{u}}_j = \begin{cases} \mathbf{u}_j, & \text{if } r_j = 0, \\ \overline{\mathbf{u}}_j & \text{if } r_j = 1. \end{cases}$$

[1] Splitting the string into blocks of length m and then performing the "approximate" balancing task over these blocks would incur a redundancy of $\frac{n}{m} + cm$, where c is a constant. The redundancy is minimized when the summands are of the same order, $\sqrt{n}$. This justifies the use of our partition choice.

It is straightforward to verify from (3.5) that $\tilde{\mathbf{u}} = \mathbf{s}_1 + \mathbf{s}_2 \bmod 2$. Since $\mathbf{s}_1, \mathbf{s}_2 \in \mathcal{S}_2(n)$ are B_2 strings over $\mathbb{F}_2^n$, we can recover $\mathbf{s}_1$ and $\mathbf{s}_2$ from $\tilde{\mathbf{u}}$, which concludes the proof. □

3.5 Upper Bounds on h-MC Codes

Next, we derive an upper bound on the maximum rate of an h-MC code. To this end, recall that $C_p \subseteq \{0,1\}^n$ is an h-prefix code if for any two subsets of sizes $\bar{h} \leqslant h$, say $\mathcal{S}_1, \mathcal{S}_2 \subseteq C_p$, $\mathcal{M}_p(\mathcal{S}_1) \neq \mathcal{M}_p(\mathcal{S}_2)$. Let $C_h^{(MC)}(n)$ be the size of the largest h-MC code of codelength n and suppose that $C_h^{(p)}(n)$ is the size of the largest h-prefix code of codelength n. Formally, we use $R_h^{(MC)}$ to denote the maximum asymptotic rate of an h-MC code,

$$R_h^{(MC)} = \lim_{n\to\infty} \sup \frac{1}{n} \log |C_h^{(MC)}(n)|.$$

We show next that when h is even, $R_h^{(MC)} \leqslant 1 - \frac{1}{2}\left(\frac{1}{1+\frac{1}{h}}\right)$. Once again, for simplicity of exposition, we focus on the case $h = 2$ before considering the general result.

The next lemma states that in order to derive an upper bound on the quantity $R_h^{(MC)}$, we can limit our attention to prefix codes. The result follows since the set of all suffixes is a function of the set of all prefixes provided the total number of ones in each codeword is the same and known beforehand.

Lemma 8. *For any $\epsilon > 0$, there exists an $n_\epsilon > 0$ such that for all $n \geqslant n_\epsilon$, one has*

$$\frac{1}{n} \log |C_h^{(MC)}(n)| \leqslant \frac{1}{n} \log |C_h^{(p)}(n)| + \epsilon.$$

Proof. To simplify the discussion, we focus on the case where $h = 2$; the extension to $h > 2$ is straightforward. For $w \in [n]$, let $C_2^{(w)}(n) \subseteq C_h^{(MC)}(n)$ denote the set of codewords of weight w in $C_2^{(MC)}(n)$. By the pigeon-hole principle, there exists a $w^* \in [n]$ where $|C_2^{(w^*)}(n)| \geqslant \frac{1}{n}|C_2^{(MC)}(n)|$. Given two codewords in $C_2^{(w^*)}(n)$, say $\mathcal{S} = \{\mathbf{s}_1, \mathbf{s}_2\}$, we can easily determine $\mathcal{M}_p(\mathcal{S})$. Assuming that only the prefix composition set is known, the set $\mathcal{M}_s(\mathcal{S})$ can be derived as follows. To determine the compositions of suffixes of length i, for $i \in [n]$, we subtract from w^* the number of ones in each prefix of length $n-i$. For instance, suppose the compositions of prefixes of length $n-1$ of $\mathbf{s}_1, \mathbf{s}_2$

are $\Big\{\{1^{w^*}, 0^{n-w^*}\}, \{1^{w^*-1}, 0^{n-w^*+1}\}\Big\}$. Then, the length-1 suffixes of $\mathbf{s}_1, \mathbf{s}_2$ are $\Big\{\{1\}, \{0\}\Big\}$. This implies that $n|C_2^{(p)}(n)| \geqslant n|C_2^{(w^*)}(n)| \geqslant |C_2^{(MC)}(n)|$, which establishes the desired result. □

Let us first turn our attention to $h = 2$. For any $\mathbf{s} \in C_2^{(p)}(n)$, we write $\mathbf{s}$ as $\mathbf{s} = \mathbf{ab} \in C_2^{(p)}(n)$, where $\mathbf{a} \in \{0,1\}^{\alpha n}$ equals the first αn symbols of $\mathbf{s}$ and $\mathbf{b}$ equals the last $(1-\alpha)n$ symbols of $\mathbf{s}$. We represent the codewords in the codebook using a bipartite graph $G = (V_P, V_S, E)$ with

$$V_P = \Big\{\mathbf{a} \in \{0,1\}^{\alpha n} : \exists \mathbf{s} \in C_2^{(p)}(n) \text{ s.t. } \boldsymbol{s} = \mathbf{ab}\Big\}, \tag{3.9}$$

and

$$V_S = \Big\{\mathbf{b} \in \{0,1\}^{(1-\alpha)n} : \exists \mathbf{s} \in C_2^{(p)}(n) \text{ s.t. } \boldsymbol{s} = \mathbf{ab}\Big\}. \tag{3.10}$$

In this setting, an edge $(v_1, v_2) \in E$, with $v_1 \in V_P$ and $v_2 \in V_S$, connects an admissible prefix (vertex in V_P) to an admissible suffix (vertex in V_S) so that every edge corresponds to a codeword in $C_2^{(p)}$ and vice versa.

Let $w \in \{0, 1, \ldots, \alpha n\} = [[\alpha n + 1]]$. We also find it useful to work with another bipartite graph $G^{(w)} = (V_P^{(w)}, V_S^{(w)}, E^{(w)})$ whose edges are a subset of the edges in E. The partition of the vertices $V^{(w)} = (V_P^{(w)}, V_S^{(w)})$ is such that $v_1 \in V_P^{(w)}$ if and only if the prefix $\boldsymbol{a} \in \{0,1\}^{\alpha n}$ represented by the vertex v_1 in G has weight w, and in addition, $v_2 \in V_S^{(w)}$ if and only if there exists a $v_1 \in V_P^{(w)}$ such that $(v_1, v_2) \in E$. The set $E^{(w)} \subseteq E$ is such that $(v_1, v_2) \in E^{(w)}$ if $v_1 \in V_P^{(w)}$ and $v_2 \in V_S^{(w)}$.

The next result will be used in the proof of Theorem 10.

Lemma 9. *The graph $G^{(w)}$ cannot contain a cycle of length four.*

Proof. Suppose, on the contrary that $G^{(w)}$ contains a 4-cycle, say

$$(\mathbf{a}_1\mathbf{b}_1, \mathbf{a}_2\mathbf{b}_2, \mathbf{a}_1\mathbf{b}_2, \mathbf{a}_2\mathbf{b}_1).$$

Then,

$$\mathcal{M}_p(\mathbf{a}_1\mathbf{b}_1) \cup \mathcal{M}_p(\mathbf{a}_2\mathbf{b}_2) = \mathcal{M}_p(\mathbf{a}_2\mathbf{b}_1) \cup \mathcal{M}_p(\mathbf{a}_1\mathbf{b}_2).$$

To verify the above claim, note that all prefixes of length αn have to be the same since $\mathcal{M}_p(\boldsymbol{a}_1) \cup \mathcal{M}_p(\boldsymbol{a}_2) = \mathcal{M}_p(\boldsymbol{a}_2) \cup \mathcal{M}_p(\boldsymbol{a}_1)$. Furthermore, since

$\mathrm{wt}(\mathbf{a}_1) = \mathrm{wt}(\mathbf{a}_2)$ it is straightforward to verify that all prefixes of length longer than αn are the same among $\mathcal{M}_p(\mathbf{a}_1\mathbf{b}_1) \cup \mathcal{M}_p(\mathbf{a}_2\mathbf{b}_2), \mathcal{M}_p(\mathbf{a}_2\mathbf{b}_1) \cup \mathcal{M}_p(\mathbf{a}_1\mathbf{b}_2)$. But this contradicts the fact that the prefixes and suffixes involved correspond to a 2-prefix code. This establishes the claimed result. □

We are now ready to prove our upper bound on h-prefix codes for $h = 2$.

Theorem 10. *For any $\epsilon > 0$, there exists an $n_\epsilon > 0$ such that for all $n \geqslant n_\epsilon$ one has $\frac{1}{n} \log |C_2^{(p)}(n)| \leqslant \frac{2}{3} + \epsilon$.*

Proof. In order to bound the number of codewords in $C_2^{(p)}(n)$, we will upper bound the number of edges in the graph $G = (V_P, V_S, E)$. To this end, we consider the maximum number of edges in the graph $G^{(w)} = (V_P^{(w)}, V_S^{(w)}, E^{(w)})$. It follows from the pigeonhole principle that there exists a $w^* \in [[\alpha n + 1]]$ such that

$$\left|E^{(w^{(*)})}\right| \geqslant \frac{|E|}{\alpha n + 1}.$$

Thus, $\frac{1}{n} \log \left|E^{(w^{(*)})}\right|$ can be approximated by $\frac{1}{n} \log \left|C_2^{(p)}(n)\right|$ for n sufficiently large.

According to the previous lemma, $G^{(w^{(*)})}$ cannot contain a 4-cycle. It is well known that the number of edges in an $m_1 \times m_2$ bipartite graph without cycles of length 4 is at most [57]

$$m_1 m_2^{\frac{1}{2}} + m_1 + m_2. \tag{3.11}$$

Letting $\alpha n = \frac{n}{3}$ in (3.11) so that $m_1 = 2^{n/3}$ and $m_2 = 2^{2n/3}$ gives

$$\frac{1}{n} \log \left|E^{(w^{(*)})}\right| \leqslant \frac{2}{3} + \mathcal{O}\left(\frac{1}{n}\right).$$

This implies the desired result. □

The next corollary follows from the previous theorem and Lemma 8.

Corollary 3. *A 2-prefix code must have a rate bounded as $R_2^{(MC)} \leqslant \frac{2}{3}$.*

Next, we consider the extension to the case where $h > 2$ based on the same approach. Let $C_h^{(p)}(n)$ denote an h-prefix code of length n. As before, we represent our codewords using a graph $G^{(h)} = (V_P^{(h)}, V_S^{(h)}, E^{(h)})$

as defined in (3.9) and (3.10), except that $(\mathbf{a}, \mathbf{b}) \in E^{(h)}$ if and only if $(\mathbf{a}, \mathbf{b}) \in C_h^{(p)}(n)$. As before, we will also work with the bipartite graph $G^{(w,h)} = (V_P^{(w,h)}, V_S^{(w,h)}, E^{(w,h)}) \subseteq G^{(h)}$, which is restricted to only use prefixes of weight w. Our next lemma is a natural generalization of Lemma 9.

Lemma 10. *The graph $G^{(w,h)}$ cannot contain a $2h$-cycle.*

Proof. Suppose, on the contrary, that the statement in the lemma does not hold and that $(\mathbf{a}_1, \mathbf{b}_1)$, $(\mathbf{b}_1, \mathbf{a}_2)$, $(\mathbf{a}_2, \mathbf{b}_2)$, $\ldots$, $(\mathbf{a}_h, \mathbf{b}_h)$, $(\mathbf{b}_h, \mathbf{a}_1)$. Then, we have:

$$\mathbf{a}_1\mathbf{b}_1, \mathbf{a}_2\mathbf{b}_2, \mathbf{a}_3\mathbf{b}_3, \ldots, \mathbf{a}_h\mathbf{b}_h \in C_h^{(p)}(n), \tag{3.12}$$

but also that

$$\mathbf{b}_1\mathbf{a}_2, \mathbf{b}_2\mathbf{a}_3, \mathbf{b}_4\mathbf{a}_5, \ldots, \mathbf{b}_h\mathbf{a}_1 \in C_h^{(p)}(n). \tag{3.13}$$

Since all the prefixes in $G^{(w,h)}$ have weight w, it is straightforward to verify that the set of codewords from (3.12) and the set in (3.13) have the same prefix composition multisets. □

The next result follows from the same arguments used in Theorem 10 and Corollary 3.

Theorem 11. *For odd h, $R_h^{(MC)} \leqslant \frac{h+1}{2h}$. For even h, $R_h^{(MC)} \leqslant 1 - \frac{1}{2}\left(\frac{1}{1+\frac{1}{h}}\right)$.*

Proof. The result follows using the same arguments as those described in Theorem 10 and by noting that the maximum number of edges in a $m_1 \times m_2$ bipartite graph that does not contain a cycle of length $2h$ is at most $(m_1 m_2)^{h+1}\, h + m_1 + m_2$ when h is odd [57]. For the case when h is even, the maximum number of edges is $m_1^{\frac{k+2}{2k}}\, m_2^{\frac{1}{2}} + m_1 + m_2$ [57]. □

As a final note, we observe that the work in [54, 55] also considered the case of nonbinary B_h codes for $h = 2$. The main result is that for a large enough alphabet, the maximum asymptotic rate of nonbinary B_2 codes is at most $\frac{1}{2}$.

3.6 Error Models and Error-Correction

The MS/MS readout technique is error prone. Often, not all the masses of prefixes and suffixes are measured and/or reported. Furthermore, polymer fragmentation causes the loss of some atoms and creates errors in the actual mass values; as a result, some fragments can gain or lose in mass value based on how the fragment was created.

Mass errors can lead to issues in the process of mixture reconstruction and hence, in what follows, we first describe common error patterns in MS/MS readouts and describe simple techniques that can be used to correct them. In all the described error-modes, as throughout the whole text, we tacitly assume that one can actually determine the length of the prefixes/suffixes based on their masses. This is possible if the masses of 0s and 1s differ significantly (for example, if the masses of the 1 or 0 molecules differ by at least n) or if other design criteria are met.

Converting practical mass errors into abstract error models is rather challenging. This is why we first describe how to model such errors, and in particular, errors that cannot be automatically detected and corrected. The focus of the first part is missing prefixes or suffixes. We start with the description of the effect of such errors on the process of reconstructing *a single string*, and then proceed to describe how missing string errors affect the reconstruction of *a mixture of multiple strings*. In both settings, we break up the discussion into two cases: One, in which we explain how to use the natural redundancy ensured by the presence of both prefixes and suffixes of the string to identify and correct errors; and another one, where we explain how to add controlled redundancy to mitigate the effect of MS/MS errors. In the latter case, we describe several simple error-control schemes that either add redundancy to the B_h strings themselves, and/or use unequal error-protection for various substrings in the Dyck-B_h codestrings as well. Furthermore, based on the approach that converts prefix/suffix masses into sums of codestrings, we introduce a straightforward idea for encoding that uses integrals and derivatives of strings.

Consider a prefix-suffix composition multiset $\mathcal{M}(\mathbf{s})$ of a Dyck string $\mathbf{s}$ of length n. A single missing composition can be identified and corrected without coding redundancy: Since either $\mathcal{M}_p(\mathbf{s})$ or $\mathcal{M}_s(\mathbf{s})$ is available for reconstruction, one of these two multisets will contain no errors and can

consequently be used for error-free reconstruction. The following example illustrates that it is not always possible to reconstruct $\mathcal{M}(\mathbf{s})$ of a string $\mathbf{s}$ from an erroneous prefix-suffix composition multiset $\tilde{\mathcal{M}}(\mathbf{s})$ that contains two or more erasures if no redundancy is used.

Example 7. *Let $\tilde{\mathcal{M}}_p(\mathbf{s})$ and $\tilde{\mathcal{M}}_s(\mathbf{s})$ denote the prefix and suffix composition multisets of $\mathbf{s}$ with missing fragment errors, respectively. Furthermore, let $\tilde{\mathcal{M}}(\mathbf{s}) = \tilde{\mathcal{M}}_p(\mathbf{s}) \cup \tilde{\mathcal{M}}_s(\mathbf{s})$.*

- *Consider the string $\mathbf{s} = 111000$. Given*

$$\tilde{\mathcal{M}} = \{1, 1^3, 01^3, 0^2 1^3, 0^3 1^3, 0^3 1^3, 0^3 1^2, 0^3 1, 0^3, 0^2, 0\},$$

  one can immediately see that a prefix composition of length two is missing. Since the weight of the string is three and the suffix composition of length $6-2=4$ is $0^3 1$, it is clear that the missing composition is 1^2.

- *Consider the same string, and the erroneous prefix-suffix composition multiset*

$$\tilde{\mathcal{M}} = \{1, 1^3, 01^3, 0^2 1^3, 0^3 1^3, 0^3 1^3, 0^3 1^2, 0^3, 0^2, 0\}.$$

  Clearly, the composition of a prefix of length two and a suffix of length four are missing. However, the constraints imposed by the prefix compositions 1 and 1^3 imply that the composition of the prefix of length two is 1^2. Similarly, the constraints imposed by suffix compositions $0^3 1^2$ and 0^3 imply that the composition of the suffix of length four is $0^3 1$.

- *In the third example, let the string be $\mathbf{s} = 110100$, and the erroneous prefix-suffix composition multiset*

$$\tilde{\mathcal{M}} = \{1, 1^2, 01^2, 0^2 1^3, 0^3 1^3, 0^3 1^3, 0^3 1^2, 0^3 1, 0^2, 0\}.$$

  From $\tilde{\mathcal{M}}_p$ one can reconstruct the partial prefix $110\varepsilon\varepsilon 0$ (where 'ε' denotes that the corresponding bit cannot be determined from the procedure described in Claim 11); similarly, from $\tilde{\mathcal{M}}_s$ one can reconstruct the partial suffix string $11\varepsilon\varepsilon 00$. By combining the partially reconstructed strings with erasures, we can easily recover the bits in all positions

except for position four, and then using a process similar to the one described in the previous examples reconstruct $\mathbf{s} = 110100$ *and* $\mathcal{M}(\mathbf{s})$.

- *Next, let*

$$\tilde{\mathcal{M}} = \{1, 1^2, 01^3, 0^2 1^3, 0^3 1^3, 0^3 1^3, 0^3 1^2, 0^3 1, 0^2, 0\}.$$

Note that both $\mathcal{M}(111000)$ *and* $\mathcal{M}(110100)$ *are consistent with the erroneous composition multiset and hence reconstruction is impossible.*

□

Based on Example 7, a number of observations are in place (any comment pertaining to prefixes also applies to suffixes and vice versa, due to symmetry).

- If $\tilde{\mathcal{M}}_p$ is such that prefix compositions of lengths $i, i+1, i+2, \ldots, i+j-2$ are erased, while compositions of prefixes of length $i-1$, $i+j-1$ and $i+j$ remain intact, then the bits $i, i+1, i+2, \ldots, i+j, i+j-1$ of the prefix string cannot be inferred using the technique described in Claim 11. Such a prefix composition list is said to have a *contiguous erasure burst* of length j, starting at index i.

- If the weight of the prefix composition of length $i-1$ is w_{i-1} and that of length $i+j-1$ is w_{i+j-1}, then among the bits at positions $i, i+1, i+2, \ldots, i+j-2$, exactly $w_{i+j-1} - w_i$ are 1s, while the remaining bit values are 0s.

- If $\mathcal{M}$ is missing t compositions, then either $\mathcal{M}_p$ or $\mathcal{M}_s$ is missing at most $\lfloor \frac{t}{2} \rfloor$ compositions.

Thus, in the presence of t prefix-suffix erasures, one is always able to reconstruct the string with at most t missing bits. The erasures that occur in the prefix (suffix) strings are correlated and every erasure occurs as part of a contiguous burst of length $\geqslant 2$. By comparing the string reconstructed using prefixes with that using suffixes, certain types of erasures can be corrected as further illustrated in Example 8. In summary, if at most t_p prefix compositions are missing, and at most t_s suffix compositions are missing, then one needs to correct not more than $2 \min \{t_p, t_s\}$ erasures in the prefix (suffix)

string, each occurring in a contiguous burst of length at least two. As will be shown Section 3.6.3, the length-two bursts can be completely eliminated from analysis and subsequent coding approaches by resorting to the use of integrals of strings, which amounts to running sums (over the reals) of the elements of the string [39].

Example 8. *Let us revisit Example 7. Given the following erroneous multiset of the string* $\mathbf{s} = 110100$,

$$\tilde{\mathcal{M}} = \{1, 1^2, 01^2, 0^2 1^3, 0^3 1^3, 0^3 1^3, 0^3 1^2, 0^3 1, 0^2, 0\},$$

from $\tilde{\mathcal{M}}_p$ *we reconstructed* $110\varepsilon\varepsilon 0$*, and from* $\tilde{\mathcal{M}}_s$ *we reconstructed* $11\varepsilon\varepsilon 00$*. Using these two strings with erasures we were able to reconstruction the original string.*

Next, assume that we are instead given the multiset $\tilde{\mathcal{M}} = \{1, 1^2, 0^2 1^3, 0^3 1^3, 0^3 1^3, 0^3 1^2, 0^2, 0\}$*. From* $\tilde{\mathcal{M}}'_p$ *we can reconstruct* $11\varepsilon\varepsilon\varepsilon 0$*, and from* $\tilde{\mathcal{M}}_s$ *we can reconstruct* $1\varepsilon\varepsilon\varepsilon 00$*. By combining the two reconstructions we can only recover* $11\varepsilon\varepsilon 00$*. The multiset* $\tilde{\mathcal{M}}$ *is consistent with both* $\mathcal{M}(111000)$ *and* $\mathcal{M}(110100)$*.* □

A simple observation for the t missing prefix/suffix model is as follows. A missing prefix composition of length i can be recovered from a suffix composition of length $n-i$. Thus, the number of error patterns that can be corrected without additional coding redundancy, independently of whether the string is a Dyck string, is at least

$$\sum_{i=0}^{t} \binom{n}{i} \binom{n-i}{t-i} \geqslant \binom{n}{\lfloor \frac{t}{2} \rfloor} \binom{n - \lfloor \frac{t}{2} \rfloor}{\lceil \frac{t}{2} \rceil}.$$

This follows from the fact that we can choose $0 \leqslant i \leqslant t$ missing masses in the prefix set, fix those i masses in the suffix set as "observed" and then select additional $t - i$ missing masses from the remaining $n - i$ suffixes.

Based on Example 8, and by noting that prefixes are red from the left, while suffixes are read from the right, define

$\mathcal{I}_p = \{(i_p, j_p)$ s.t. the prefix string has a contiguous erasure burst of length j_p starting at index $i_p\}$,

and

$\mathcal{I}_s = \{(i_s, j_s)$ s.t. the suffix string has a contiguous erasure burst of length

j_s starting at index $i_s\}$.

The prefix string is said to have a *contiguous erasure overlap* of length ℓ with the suffix string if there exists $(i_p, j_p) \in \mathcal{I}_p$ and $(i_s, j_s) \in \mathcal{I}_s$ such that $j_s, j_p \geqslant \ell$ *and* either $i_p \leqslant i_s, (i_p + j_p - i_s) = \ell$ or $i_s < i_p, (i_s + j_s - i_p) = \ell$. We say that the prefix string and the suffix string erasures *do not overlap* if for all lengths $\ell \geqslant 1$ the prefix string does not have a contiguous erasure overlap with that of the suffix string.

Claim 16. *A Dyck string* $\mathbf{s}$ *can be reconstructed from the prefix-suffix composition multiset* $\tilde{\mathcal{M}}(\mathbf{s})$ *without using error-correction redundancy if the missing prefixes and suffixes are such that*

- *The prefix string and the suffix string do not overlap; or,*

- *The prefix string has exactly one contiguous erasure overlap with the suffix string, and the overlap is of length one.*

We next turn our attention to describing the effect of missing prefixes and suffixes on multistring reconstruction from the union of the underlying prefix-suffix composition multiset, and detection/correction strategies that may be used without resorting to controlled error-control redundancy.

As for the case of a single string, one missing prefix-suffix composition in $\mathcal{M}(\mathbf{s}_1) \cup \mathcal{M}(\mathbf{s}_2) \cdots \cup \mathcal{M}(\mathbf{s}_h)$ corresponding to the Dyck strings $\mathbf{s}_1, \mathbf{s}_2, \ldots \mathbf{s}_h$ can be easily corrected as either the union of prefix composition multiset or the union of suffix composition multiset is error-free. But when more than one error is present, it is not always possible to reconstruct $\mathcal{M}(\mathbf{s}_1) \cup \mathcal{M}(\mathbf{s}_2) \cdots \cup \mathcal{M}(\mathbf{s}_h)$ from the union of the erroneous prefix-suffix composition multiset $\tilde{\mathcal{M}}(\mathbf{s}_1) \cup \tilde{\mathcal{M}}(\mathbf{s}_2) \cdots \cup \tilde{\mathcal{M}}(\mathbf{s}_h)$ as illustrated below. Here, we find the following definitions useful. The partially reconstructed prefix of the sum of strings constructed from the union of erroneous prefix composition multisets using Claim 11 is referred to as the *partial prefix-sum string* and its analogue pertaining to suffixes is referred to as the *partial suffix-sum string*.

Example 9.

- *Let us revisit Example 4 under the assumption that we are given the*

erroneous multiset

$$\tilde{\mathcal{M}}(\mathbf{t}_1) \cup \tilde{\mathcal{M}}(\mathbf{t}_2) = \{1, 1, 01, 1^2, 01^2, 0^2 1^2, 01^3, 0^2 1^3, 0^2 1^3, 0^3 1^3, 0^3 1^3, \\ 0^3 1^3, 0^3 1^3, 0^3 1^2, 0^3 1^2, 0^2 1^2, 0^3 1, 0^2 1, 0^2 1, 01, 0^2, 0, 0\}.$$

Clearly, the composition of a prefix of length three is missing. Thus, using the union of the suffix composition multiset,

$$\tilde{\mathcal{M}}_s(\mathbf{t}_1) \cup \tilde{\mathcal{M}}_s(\mathbf{t}_2) = \mathcal{M}_s(\mathbf{t}_1) \cup \mathcal{M}_s(\mathbf{t}_2) = \{0^3 1^3, 0^3 1^3, 0^3 1^2, 0^3 1^2, 0^2 1^2, 0^3 1, \\ 0^2 1, 0^2 1, 01, 0^2, 0, 0\},$$

one can easily determine $\mathbf{t}_1 + \mathbf{t}_2 = 211110$ *using the procedure outlined in Example 4. If one were to use* $\tilde{\mathcal{M}}_p(\mathbf{t}_1) \cup \tilde{\mathcal{M}}_p(\mathbf{t}_2)$ *instead, while setting aside the fact that a composition of length three is missing, the same procedure applied to the third symbol of* $\mathbf{t}_1 + \mathbf{t}_2$ *would results in* $wt(01^2) - wt(01) - wt(1^2) = -1$*, which is clearly indicative of an error. A negative value in the partial prefix-sum string or suffix-sum string is a clear indication of one or more missing compositions or composition errors in general.*

- *In the next example, we assume that we are given the following erroneous multiset instead:*

$$\tilde{\mathcal{M}}(\mathbf{t}_1) \cup \tilde{\mathcal{M}}(\mathbf{t}_2) = \{1, 1, 01, 1^2, 01^2, 0^2 1^2, 01^3, 0^2 1^3, 0^2 1^3, 0^3 1^3, 0^3 1^3, \\ 0^3 1^3, 0^3 1^3, 0^3 1^2, 0^2 1^2, 0^3 1, 0^2 1, 0^2 1, 01, 0^2, 0, 0\}.$$

It is once again easy to see that a prefix of length three and a suffix of length five are missing. Using the erroneous union of the prefix composition multisets, $\tilde{\mathcal{M}}_p(\mathbf{t}_1) \cup \tilde{\mathcal{M}}_p(\mathbf{t}_2)$*, one can recover the partial prefix-sum* $21\varepsilon\varepsilon 10$*, and similarly, using the union of the suffix composition multisets,* $\tilde{\mathcal{M}}_s(\mathbf{t}_1) \cup \tilde{\mathcal{M}}_s(\mathbf{t}_2)$*, one can recover the partial suffix-sum* $\varepsilon\varepsilon 1110$*. By combining the two partial sums, one can recover* $\mathbf{t}_1 + \mathbf{t}_2 = 211110$*. Note that the third bits in the two strings equal* 0 *and* 1 *or* 1 *and* 0*, implying that the correct prefix compositions of length three are either* $\{01^2, 01^2\}$ *or* $\{1^3, 0^2 1\}$*. Since* $\{1^3, 0^2 1\} \cap \tilde{\mathcal{M}}_p(\mathbf{t}_1) \cup \tilde{\mathcal{M}}_p(\mathbf{t}_2) = \phi$*, one can conclude that the missing prefix composition is* 01^2*. A similar line of reasoning may be used to recover the missing suffix composition* $0^3 1^2$*.*

- *In the third scenario, we are given the following erroneous multiset*

$$\tilde{\mathcal{M}}(\mathbf{t}_1) \cup \tilde{\mathcal{M}}(\mathbf{t}_2) = \{1, 1, 01, 1^2, 01^2, 0^2 1^2, 0^2 1^3, 0^2 1^3, 0^3 1^3, 0^3 1^3, \\ 0^3 1^3, 0^3 1^3, 0^3 1^2, 0^2 1^2, 0^3 1, 0^2 1, 0^2 1, 01, 0^2, 0, 0\}.$$

  Here, one prefix of length three and one of length four are missing, as well as a suffix of length four and another of length five. Using the erroneous union of the prefix composition multisets, $\tilde{\mathcal{M}}_p(\mathbf{t}_1) \cup \tilde{\mathcal{M}}_p(\mathbf{t}_2)$*, one can recover the partial prefix-sum* $21\varepsilon\varepsilon\varepsilon 0$*, and similarly using the union of the suffix composition multisets,* $\tilde{\mathcal{M}}_s(\mathbf{t}_1) \cup \tilde{\mathcal{M}}_s(\mathbf{t}_2)$*, one can recover the partial suffix-sum* $\varepsilon\varepsilon\varepsilon 110$*. By combining the two partial strings, we obtain* $\mathbf{t}_1 + \mathbf{t}_2 = 21\varepsilon 110$*. Since* $wt(\mathbf{t}_1 + \mathbf{t}_2) = wt(\mathbf{t}_1) + wt(\mathbf{t}_2) = 3 + 3 = 6$*, it must be that* $\mathbf{t}_1 + \mathbf{t}_2 = 211110$*. Hence, as for the case of a single string, one can recover the sum of the mixture strings even when both the partial prefix and suffix contain errors.*

- *In the last example to consider, assume that we are given the erroneous multiset*

$$\tilde{\mathcal{M}}(\mathbf{t}_1) \cup \tilde{\mathcal{M}}(\mathbf{t}_2) = \{1, 1, 01, 1^2, 01^2, 0^2 1^2, 01^3, 0^2 1^3, 0^2 1^3, 0^3 1^3, 0^3 1^3, \\ 0^3 1^3, 0^3 1^3, 0^3 1^2, 0^3 1^2, 0^2 1^2, 0^3 1, 0^2 1, 01, 0^2, 0, 0\}.$$

  In this case, the prefix and suffix of length three are missing. Using the erroneous union of the prefix composition multisets, $\tilde{\mathcal{M}}_p(\mathbf{t}_1) \cup \tilde{\mathcal{M}}_p(\mathbf{t}_2)$*, one can recover the partial prefix-sum* $21\varepsilon\varepsilon 10$*, and, similarly, using the union of the suffix composition multisets,* $\tilde{\mathcal{M}}_s(\mathbf{t}_1) \cup \mathcal{M}'_s(\mathbf{t}_2)$*, one can recover the partial suffix-sum* $21\varepsilon\varepsilon 10$*. However, for this case one cannot recover the missing compositions or the sum of the two input strings: Both* $\mathcal{M}(111000) \cup \mathcal{M}(101010)$ *and* $\mathcal{M}(110100) \cup \mathcal{M}(101010)$ *are consistent with the given input erroneous composition multiset.*

□

We hence have the following claim.

Claim 17. *The sum over the reals of* h *Dyck strings* $\mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_h$ *of length* n *can be reconstructed from the union of their prefix-suffix composition multiset with erasures* $\tilde{\mathcal{M}}(\mathbf{s}_1) \cup \tilde{\mathcal{M}}(\mathbf{s}_2) \cup \cdots \cup \tilde{\mathcal{M}}(\mathbf{s}_h)$ *if*

1) The prefix sum string and the suffix sum string do not overlap; or,
2) The prefix sum string has exactly one contiguous erasure overlap with the suffix sum string, and the overlap is of length one.

As before, we remark that the multiset of all h prefix compositions of length $1 \leqslant i \leqslant n$ can be recovered provided the complete multiset of h suffix compositions of length $n-i$ is available. Hence, the number of error patterns that can be corrected without additional redundancy is at least the number of error patterns such that the missing prefix compositions of length i can be recovered from the complete set of suffix compositions of length $n-i$ and vice-versa. To that end,

1) Assume that erasures occur in compositions of exactly $1 \leqslant j \leqslant t$ different lengths. There are $\binom{n}{j}$ different ways to choose these lengths.

2) For every $i \in [n]$, either the erasures are contained in the multiset of prefix compositions of length i or the multiset of suffix compositions of length $n-i$. Thus, there are 2^j different prefix-suffix composition error patterns for the j different lengths.

3) An erroneous prefix-suffix composition multiset comprising strings of length i contains between 1 and h errors. Thus, the number of error patterns restricted to j composition lengths equals the number of positive solutions of the equation $t = t_1 + t_2 + \cdots + t_j$, such that $t_1, t_2, \ldots, t_j \leqslant h$, and is well known to be $\binom{t-1}{j-1}$ provided that $t \leqslant h$.

Thus, assuming that the number of missing oligos is $t \leqslant h$ and by adding up all possible contributions for different choices of j, one can find a simple lower bound on the number of error patterns that be corrected without additional redundancy

$$\sum_{j=0}^{t} \binom{n}{j} \binom{t-1}{j-1} 2^j = \sum_{j=0}^{t} \binom{n}{j} \binom{t-1}{t-j} 2^j.$$

3.6.1 One-Step Error-Correction

In what follows, we present a simple scheme that can correct up to t missing prefix-suffix composition errors. Recall that the B_h codebook $\mathcal{S}_h(n)$, described in the previous sections can be constructed using the columns of a parity-check matrix of a code with minimum Hamming distance $d \geqslant 2h+1$. The idea behind our error-correction technique is to ensure that the real-

valued sum of every h-string subset of the code is an error-tolerant codestring. An approach to this problem was first proposed in [58] for the purpose of designing signature codes for the noisy MAC problem (i.e., codes capable of correcting errors in the syndrome of a received word) and it consists of encoding the columns of a parity-check matrix $\tilde{H}^{k\times n}$, capable of correcting h substitution errors, using a linear binary code that can correct $\lfloor \frac{t}{2} \rfloor$ substitution errors. Note that the parameter t can be chosen independently of the parameter h as long as $\lfloor \frac{t}{2} \rfloor \leqslant k \leqslant n$. For encoding purposes, the authors suggest using two binary BCH codes, so that $\tilde{H}$ is the parity check matrix of a BCH code of designed distance $\geqslant 2h+1$, while the parity-check matrix used to introduce error-control redundancy to the columns of $\tilde{H}$ is also chosen according to a BCH code with dimension k, length n and redundancy not exceeding $\lfloor \frac{t}{2} \rfloor \log(n+1)$, and capable of correcting at least $\lfloor \frac{t}{2} \rfloor$ substitution errors. Here and elsewhere in this section all logarithms are taken to the base two unless stated otherwise. Clearly, the only difference is that in our setting, we only encounter erasures in the coded strings (the augmented columns), and hence can handle t erasures.

Note that this construction, as pointed out by the authors, does not fully exploit the fact that addition is performed over the reals and not over the field $\mathbb{F}_2$. As in the previous construction, $\tilde{H}$ is the parity-check matrix of a binary h substitution error-correction code. But instead of using another binary code to protect the syndromes, [58] suggests finding the smallest prime $p > h$, and using a linear code over $\mathbb{F}_p$ (e.g., a Reed-Solomon code of length $p-1$) for the syndrome error-control redundancy. The dimension of the latter code equals n, and it is required that the code be able to correct t substitution errors over the field $\mathbb{F}_p$. Since the redundancy is nonbinary, each symbol of the parity-check string is converted into a string of length $\log(p+1)$, representing the binary expansion of the symbol over $\mathbb{F}_p$. The binary expansions are stacked on top of each other according to the given parity-check string. The interesting observation is that, from the sum of the binary strings over the reals, one can clearly obtain the binary expansion of the symbols in the sum, and then generate the residues modulo p of the elements of the string to obtain the redundancy information needed for decoding. The obtained code is linear.

Henceforth, we use the value N to denote the length of the uniquely reconstructable strings h-***MC*** with added error-control redundancy. It is not to be confused with the parameter N from Section 3.4 as the notation is reused to

avoid clutter. Also, as before, we let $\mathcal{S}_h(n)$ be a binary B_h code constructed using the parity-check matrix of a binary code with minimum Hamming distance $\geqslant 2h+1$, and to add the syndrome redundancy, we use a BCH code of appropriate parameters. The main observation leading to the discussion of our scheme is that due to our encoding method which uses the complementation/bit flipping procedure, we require an unequal error-protection scheme. Recall that the substring $\mathbf{r}$ used in the construction described earlier is the indicator vector for substring (of length $\sqrt{n}$ bits) flips. Errors in the $\mathbf{r}$ substring may clearly cause a burst of "complementation errors" due to the fact that $\mathbf{r}$ indicates if a string or its complement should be used. There are two approaches one can follow, by either encoding the string to handle a larger number of erasures independent on their location (the one-step procedure) or by adding specialized redundancy to the $\mathbf{r}$ string (the two-step procedure).

The one-step encoding method is illustrated in Figure 3.3 and proceeds as follows:

- Each string $\mathbf{s} \in \mathcal{S}_h(n)$ is encoded using a BCH code into an intermediary string $\mathbf{s}'$ of length m, capable of correcting $t(\sqrt{m}+1)$ erasures. The redundancy required is at most $\lceil \frac{t}{2} \rceil (\sqrt{m}+1)\log(m+1)$.

- The intermediary string $\mathbf{s}'$ of length m is subsequently encoded via the balancing procedure described in Section 3.4. The encoded balanced string has length N and belongs to a h-***MC*** code capable of correcting up to t composition erasures; here, $N = m + \frac{17}{2}\sqrt{m}$ which is at most

$$n + \lceil \frac{t}{2} \rceil (\sqrt{m}+1)\log(m+1) + \frac{17}{2}\sqrt{m}.$$

  The parameter N can be further bounded by above by

$$n + \lceil \frac{t}{2} \rceil (\sqrt{n}+1)\log n + \frac{17}{2}\sqrt{n} + \epsilon_n\sqrt{n}\left(\lceil \frac{t}{2} \rceil \log n + \frac{17}{2}\right) + \lceil \frac{t}{2} \rceil \delta_n,$$

  where $\epsilon_n = \frac{\lceil \frac{t}{2} \rceil (\sqrt{m}+1)\log(m+1)}{2n}$ and $\delta_n = \frac{\lceil \frac{t}{2} \rceil (\sqrt{m}+1)\log(m+1)}{n}$.

As either the partial prefix-sum or the partial suffix-sum string has $\leqslant t(\sqrt{m}+1)$, the binary sum of the input strings can be recovered correctly. The decoding procedure for strings involved in the sum is identical to the one described in Section 3.4, and hence omitted.

3.6.2 Two-Step Error-Correction

As observed in the one-step scheme, errors in the substring $\mathbf{r}$ cause blocks of errors in the global string: Each erasure in $\mathbf{r}$ results in $\sqrt{m}$ additional erasures, where m is the length of the approximately balanced strings. In order to overcome this issue in a more tailor-made manner, one can use unequal error-correction schemes that ensure that the binary sum of the $\mathbf{r}$ substring components across the input strings can be recovered independently from the rest of the string. The correctly reconstructed binary $\mathbf{r}$ sum can then be used for subsequent decoding of the complete collection of input strings. This two-step approach is illustrated in Figure 3.3.

As before, let $\mathcal{S}_h(n)$ be a binary B_h code constructed using the parity-check matrix of a code with minimum Hamming distance $\geqslant 2h+1$ (say, a BCH code). Furthermore, let N denote the overall length of the h-***MC*** codestrings with added redundancy for mass error-correction. The encoding method is illustrated in Figure 3.3) and proceeds as follows:

- Each string $\mathbf{s} \in \mathcal{S}_h(n)$ is encoded into an intermediary string $\mathbf{s}'$ of length m_1 capable of correcting t erasures, using a BCH code. The redundancy required is at most $t\log(m_1+1)$, and

$$m_1 \leqslant n + \lceil \frac{t}{2} \rceil \log(m_1+1).$$

- Each intermediary string $\mathbf{s}'$ of length m_1 is encoded into a Dyck string using the procedure described in Section 3.4, to arrive at a second intermediary string $\mathbf{s}$" of length m_2, where

$$m_2 = m_1 + \frac{17}{2}\sqrt{m_1}.$$

- The substring $\mathbf{r}$ of the intermediary string $\mathbf{s}$" is encoded into a codestring $\mathbf{r}\mathbf{r}'$ of total length m_3, capable of correcting t erasures. Let $m_4 = m_3 - \sqrt{m_1}$ denote the length of $\mathbf{r}'$. It is easy to see that $m_4 \leqslant \lceil \frac{t}{2} \rceil \log(m_3+1)$.

- Since the string has to be balanced, $\mathbf{r}' = r'_1 r'_2 \ldots r'_{m_4}$ is converted into $\mathbf{z} = r'_1 \bar{r'}_1 r'_2 \bar{r'}_2 \ldots r'_{m_4-1} \bar{r'}_{m_4}$, where $\bar{r'}_i = 1 - r'_i$.

- The balanced redundancy $\mathbf{z}$ is appended to the $\mathbf{r}$ substring of the in-

termediary string $\mathbf{s}$". Also, a bit 1 is added to the prefix of $\mathbf{1}$s and a bit 0 is appended to the suffix of $\mathbf{0}$s to preserve the Dyck property of the string.

The length of the coded string equals

$$N = m_1 + \frac{17}{2}\sqrt{m_1} + 2(m_3 - \sqrt{m_1}) + 2,$$

and upper-bounded in terms of the length n as

$$n + \lceil \frac{t}{2} \rceil (\log n + \mu_n) + \frac{17}{2}\sqrt{n}(1 + \nu_n) + \lceil \frac{t}{2} \rceil (\log n + \mu_n) + 2\theta_n + 2,$$

where $\mu_n = \frac{\lceil \frac{t}{2} \rceil \log(m_1+1)+1}{n}$, $\nu_n = \frac{\lceil \frac{t}{2} \rceil \log(m_1+1)}{2n}$, and $\theta_n = \frac{t \log(m_3+1)+1}{\sqrt{n+\lceil \frac{t}{2} \rceil \log(m_1+1)}}$.

3.6.3 Composition Error-Correction Using String Integrals

One observation that is apparent from the previous examples is that erasures/errors caused in one mass propagate to one more error when used to reconstruct the real-valued sum of the strings. One simple means to mitigate this problem is to use running sums of symbols, in which case the errors cancel. A more precise explanation for how to perform encoding with this approach in mind is as follows.

Without loss of generality, suppose that $t_p < t_s$. In this case, it is always possible for the errors in the suffix string to be such that we receive no additional information by considering both the prefix and suffix string, and so the problem at hand becomes to recover $\mathbf{s}$ from a set of at most $n - t_p$ prefix compositions.

Claim 18. *Suppose that $\mathcal{C}(n, d) \subseteq \mathbb{F}_2^n$ is a code with minimum Hamming distance $d = \min\{t_p, t_s\} + 1$. Let $\mathbf{s} \in \mathbb{F}_2^n$ and fix $\mathrm{wt}(\mathbf{s}) = w_0$. Let $\tilde{\mathcal{M}}_p(\mathbf{s})$ be the result of removing t_p compositions from $\mathcal{M}_p(\mathbf{s})$, and t_s compositions from $\mathcal{M}_s(\mathbf{s})$. Then, we can recover $\mathbf{s} = s_1 s_2 \dots s_n \in \{0,1\}^n$ from $\tilde{\mathcal{M}}_p(\mathbf{s})$ provided that*

$$s_1 \ (s_1 + s_2) \ (s_1 + s_2 + s_3) \ \dots \ \sum_{j=1}^{n} s_j \in \mathcal{C}(n, d).$$

Proof. Without loss of generality, assume that $t_p = \min\{t_p, t_s\}$. The result follows since for $i \in [n]$ the value of the i-th component in the string

$s_1\ (s_1+s_2)\ (s_1+s_2+s_3)\ \ldots\ \sum_{j=1}^{n} s_j \in \mathcal{C}(n,d)$ can be recovered by summing the number of 1s (modulo 2) in the i-th prefix composition. The claim then follows since we know the lengths of the compositions that are missing from the set $\tilde{\mathcal{M}}_p(\mathbf{s})$ and can hence recover the string $s_1\ (s_1+s_2)\ (s_1+s_2+s_3)\ \ldots\ \sum_{j=1}^{n} s_j$, where $\boldsymbol{s} \in \mathbb{F}_2^n$, from which $\mathbf{s}$ can be then determined uniquely. Note that for the case that $t_s = \min\{t_p, t_s\}$, since the weight of $\mathbf{s}$ is known, a missing composition of a prefix of length i can be recovered from the known composition of a suffix of length $n-i$. Thus, $t_p + t_s$ missing compositions in $\tilde{\mathcal{M}}_p(\mathbf{s})$ can be recovered from $\tilde{\mathcal{M}}_s(\mathbf{s})$ and w_0. This concludes the proof. □

Using the result of Claim 18, we can encode our mixture-strings using the following encoding technique:

- Given a string $\mathbf{s} \in \{0,1\}^n$, construct $I(\mathbf{s}) = s_1(s_1+s_2)(s_1+s_2+s_3)\ldots(s_1+s_2+\cdots+s_n) = I(\mathbf{s})_1 I(\mathbf{s})_2 \ldots I(\mathbf{s})_n$.
- Encode $I(\mathbf{s})$ using a BCH code such that the resulting string $I(\mathbf{s})R'(\mathbf{s})$, where $R'(\mathbf{s})$ denotes the redundancy bits, is of length m and can correct $\lfloor \frac{t}{2} \rfloor$ erasures. To ensure that the redundancy is itself properly balanced, we write $R(\mathbf{r})_1 = R'(\mathbf{r})_1 + I(\mathbf{r})_1$; $R(\mathbf{r})_{2i} = \bar{R(\mathbf{r})}_{2i} = 1 - R(\mathbf{r})_{2i}$, and $R(\mathbf{r})_{2i+1} = R'(\mathbf{r})_i + R'(\mathbf{r})_{i+1} + R(\mathbf{r})_{2i}$, for all $i \in [m-n-1]$, where m, n are as described in the encoding scheme of Section 3.6.3.
- Encode $\mathbf{s}$ as $\mathbf{s}R(\mathbf{s})$.

To apply the above procedure, we need to be able to partition the prefix and suffix compositions of the $\mathbf{s}R(\mathbf{s})$. This is easily achieved when $\mathbf{s}R(\mathbf{s})$ is a substring of a Dyck string such that the composition of the prefix preceding the $\mathbf{s}R(\mathbf{s})$-substring in the Dyck string is known. In particular, since the substring $\mathbf{s}R(\mathbf{s})$ occurs after the runlength of 1s in the construction of Section 3.4, the prefix compositions of the constructed string $\mathbf{s}R(\mathbf{s})$ can be recovered by subtracting the weight of the leading runlength of 1s from the corresponding compositions. Consequently, $\mathbf{s}R(\mathbf{s})$ satisfies the conditions of Claim 18 and the code is linear. Thus, the binary sum of multiple strings constructed using this technique also satisfies the conditions supporting Claim 18.

In conclusion, the t-erasure-correcting two-step procedure can be replaced by the $\lfloor \frac{t}{2} \rfloor$ error-correcting technique outlined above. This results in an

overall reduction of the redundancy by a factor of two, but adds to the encoding complexity.

3.6.4 Correcting Prefix-Suffix Mass Substitution Errors

We now briefly turn our attention to describing how substitution errors in prefix-suffix compositions influence the reconstruction process. As already pointed out, in practice we often observe a larger mass being reported as a smaller mass due to losses of atoms during the fragmentation process. Such errors can be modeled as *asymmetric errors* in which a bit equal to 1 in a prefix or suffix compositions is replaced by a 0. For convenience, we refer to such errors as *mass reducing substitution* errors. In the previous exposition, we illustrated the fact that the union of the prefix-suffix composition multisets and the sum of the input Dyck strings can be recovered even in the presence of a single erasure without additional redundancy. However, the same is not always true for mass reducing substitution errors, and hence the problem of correcting such errors is significantly more challenging. We illustrate this issue in Example 10.

Example 10.

1. *Consider the following codebook of Dyck strings* {110100 , 101010 , 110010, 111000}, *and select* $\mathbf{t}_1 = 111000$, $\mathbf{t}_2 = 110100$. *The error-free union of the prefix-suffix multisets of the strings equals*

$$\mathcal{M}(\mathbf{t}_1) \cup \mathcal{M}(\mathbf{t}_2) = \{1, 1, 1^2, 1^2, 1^3, 01^2, 01^3, 01^3, 0^2 1^3, 0^2 1^3, 0^3 1^3, 0^3 1^3, 0^3 1^3, 0^3 1^3, 0^3 1^2, 0^3 1^2, 0^3 1^1, 0^3 1^1, 0^3, 0^2 1, 0^2, 0^2, 0, 0\}.$$

Using the defining property of Dyck strings, from $\mathcal{M}(\mathbf{t}_1) \cup \mathcal{M}(\mathbf{t}_2)$ *one can reconstruct the multisets*

$$\mathcal{M}_p(\mathbf{t}_1) \cup \mathcal{M}_p(\mathbf{t}_2) = \{1, 1, 1^2, 1^2, 1^3, 01^2, 01^3, 01^3, 0^2 1^3, 0^2 1^3, 0^3 1^3, 0^3 1^3\}$$

and

$$\mathcal{M}_s(\mathbf{t}_1) \cup \mathcal{M}_s(\mathbf{t}_2) = \{0^3 1^3, 0^3 1^3, 0^3 1^2, 0^3 1^2, 0^3 1^1, 0^3 1^1, 0^3, 0^2 1, 0^2, 0^2, 0, 0\}.$$

Now, assume that one composition in the union, 1^2*, is erroneously read*

as 0^2*. Thus, in the presence of such an error, we obtain a different partition of compositions into prefix and suffix sets,*

$$\begin{aligned}\mathcal{M}_p(\mathbf{t}_1) \cup \mathcal{M}_p(\mathbf{t}_2) &= \{1, 1, 1^2, 1^3, 01^2, 01^3, 01^3, 0^2 1^3, 0^2 1^3, 0^3 1^3, 0^3 1^3\}, \\ \mathcal{M}_s(\mathbf{t}_1) \cup \mathcal{M}_s(\mathbf{t}_2) &= \{0^3 1^3, 0^3 1^3, 0^3 1^2, 0^3 1^2, 0^3 1^1, 0^3 1^1, 0^3, 0^2 1, 0^2, 0^2, \\ &\qquad 0^2, 0, 0\}.\end{aligned}$$

If it were known a priori that at most one mass reducing substitution error occurred, then an immediate conclusion would be that the composition of a prefix of length two was erroneously read as a suffix. Noting that the suffix compositions of the Dyck strings of length $n - 2 = 4$ *are* $0^3 1^1, 0^3 1^1$*, it must be that the correct prefix compositions of length* 2 *are* 1^2 *and* 1^2*. Thus, from the corrected multiset*

$$\mathcal{M}_p(\mathbf{t}_1) \cup \mathcal{M}_p(\mathbf{t}_2) = \{1, 1, 1^2, 1^2, 1^3, 01^2, 01^3, 01^3, 0^2 1^3, 0^2 1^3, 0^3 1^3, 0^3 1^3\},$$

one can easily compute $\mathbf{t}_1 + \mathbf{t}_2 = 221100$*.*

2. *Next, consider the strings* $\mathbf{t}_2 = 110100$ *and* $\mathbf{t}_3 = 110010$*. Their error-free union of prefix-suffix multisets is given by*

$$\begin{aligned}\mathcal{M}(\mathbf{t}_2) \cup \mathcal{M}(\mathbf{t}_3) = \{&1, 1, 1^2, 1^2, 01^2, 01^2, 01^3, 0^2 1^2, 0^2 1^3, 0^2 1^3, \\ &0^3 1^3, 0^3 1^3, 0^3 1^3, 0^3 1^3, 0^3 1^2, 0^3 1^2, 0^3 1^1, \\ &0^3 1^1, 0^2 1, 0^2 1, 0^2, 01, 0, 0\}.\end{aligned}$$

Using once again the defining properties of Dyck strings, from $\mathcal{M}(\mathbf{t}_1) \cup \mathcal{M}(\mathbf{t}_2)$ *we can reconstruct the multisets*

$$\mathcal{M}_p(\mathbf{t}_2) \cup \mathcal{M}_p(\mathbf{t}_3) = \{1, 1, 1^2, 1^2, 01^2, 01^2, 01^3, 0^2 1^2, 0^2 1^3, 0^2 1^3, 0^3 1^3, 0^3 1^3\}$$

and

$$\begin{aligned}\mathcal{M}_s(\mathbf{t}_2) \cup \mathcal{M}_s(\mathbf{t}_3) = \{&0^3 1^3, 0^3 1^3, 0^3 1^2, 0^3 1^2, 0^3 1^1, 0^3 1^1, 0^2 1, \\ &0^2 1, 0^2, 01, 0, 0\}.\end{aligned}$$

In addition, let us also examine the strings $\mathbf{t}_1 = 111000$ *and* $\mathbf{t}_3 =$

110010. *Their error-free union of prefix-suffix multiset is given by*

$$\mathcal{M}(\mathbf{t}_1) \cup \mathcal{M}(\mathbf{t}_3) = \{1, 1, 1^2, 1^2, 1^3, 01^2, 01^3, 0^2 1^2, 0^2 1^3, 0^2 1^3, 0^3 1^3, 0^3 1^3, \\ 0^3 1^3, 0^3 1^3, 0^3 1^2, 0^3 1^2, 0^3 1^1, 0^3 1^1, 0^3, 0^2 1, 0^2, 01, 0, 0\}.$$

In this case,

$$\mathcal{M}_p(\mathbf{t}_2) \cup \mathcal{M}_p(\mathbf{t}_3) = \{1, 1, 1^2, 1^2, 1^3, 01^2, 01^3, 0^2 1^2, 0^2 1^3, 0^2 1^3, 0^3 1^3, 0^3 1^3\},$$

$$\mathcal{M}_s(\mathbf{t}_1) \cup \mathcal{M}_s(\mathbf{t}_3) = \{0^3 1^3, 0^3 1^3, 0^3 1^2, 0^3 1^2, 0^3 1^1, 0^3 1^1, 0^3, 0^2 1, 0^2, 01, 0, 0\}.$$

Given the erroneous union of prefix-suffix composition multiset of two strings

$$\{1, 1, 1^2, 1^2, 0^3, 01^2, 01^3, 0^2 1^2, 0^2 1^3, 0^2 1^3, 0^3 1^3, 0^3 1^3, \\ 0^3 1^3, 0^3 1^3, 0^3 1^2, 0^3 1^2, 0^3 1^1, 0^3 1^1, 0^2 1, 0^2 1, 0^2, 01, 0, 0\},$$

the following scenarios are possible:
a) In $\mathcal{M}(\mathbf{t}_2) \cup \mathcal{M}(\mathbf{t}_3)$*, the prefix composition* 01^2 *was changed to* 0^3*, or*
b) In $\mathcal{M}(\mathbf{t}_1) \cup \mathcal{M}(\mathbf{t}_3)$*, the prefix composition* 1^3 *was changed to* $0^2 1$*.*
Thus, in the presence of even a single mass reducing composition error, without additional coding redundancy, neither the union of the prefix-suffix composition multiset nor the sum of the strings can be uniquely reconstructed.

3. *In the final example, let us revisit the strings* $\mathbf{t}_2 = 110100$ *and* $\mathbf{t}_3 = 110010$*. Given the erroneous composition multiset*

$$\tilde{\mathcal{M}}(\mathbf{t}_2) \cup \tilde{\mathcal{M}}(\mathbf{t}_3) = \{1, 1, 1^2, 01, 01^2, 01^2, 01^3, 0^2 1^2, 0^2 1^3, 0^2 1^3, 0^3 1^3, 0^3 1^3, \\ 0^3 1^3, 0^3 1^3, 0^3 1^2, 0^3 1^2, 0^3 1^1, 0^3 1^1, 0^2 1, 0^2 1, 0^2, 01, 0, 0\},$$

one can recover

$$\tilde{\mathcal{M}}_p(\mathbf{t}_2) \cup \tilde{\mathcal{M}}_p(\mathbf{t}_3) = \{1, 1, 1^2, 01, 01^2, 01^2, 01^3, 0^2 1^2, 0^2 1^3, 0^2 1^3, 0^3 1^3, 0^3 1^3\}$$

and

$$\tilde{\mathcal{M}}_s(\mathbf{t}_2) \cup \tilde{\mathcal{M}}_s(\mathbf{t}_3) = \{0^3 1^3, 0^3 1^3, 0^3 1^2, 0^3 1^2, 0^3 1^1, 0^3 1^1, \\ 0^2 1, 0^2 1, 0^2, 01, 0, 0\}.$$

Noting that the prefix compositions $1^2, 01$ *are not compatible with the suffix compositions* $0^3 1^1, 0^3 1^1$*, we conclude that one of these four compositions must be in error. Hence, either the prefix sum string* 211110 *or the suffix sum string* 220110 *is correct and we can examine both settings to determine the possible solutions.*

□

In summary, the examples above illustrate that a single mass reducing error can always be detected but not necessarily corrected. For the special case where a composition of a prefix or suffix of length i is replaced by the composition of a prefix or suffix of the same length, a simple modification to the coding technique described for correcting t mass erasures can be used to correct t mass substitution errors. To this end, we make the following two observations.

Although based on the pigeon-hole principle it is true that either the union of the prefix composition multisets or the union of the suffix composition multisets is such that it contains at most $\lfloor \frac{t}{2} \rfloor$ mass reducing errors, it is not possible to identify which of the two multisets contains fewer errors. As a result, the two-step encoding procedure has to involve a h-***MC*** code capable of correcting t substitution errors. This results in a four times higher redundancy compared to that used for correcting missing mass errors.

3.7 Open Problems

Many combinatorial and coding-theoretic problems related to string reconstruction from prefix-suffix compositions remain open. A sampling is listed below.

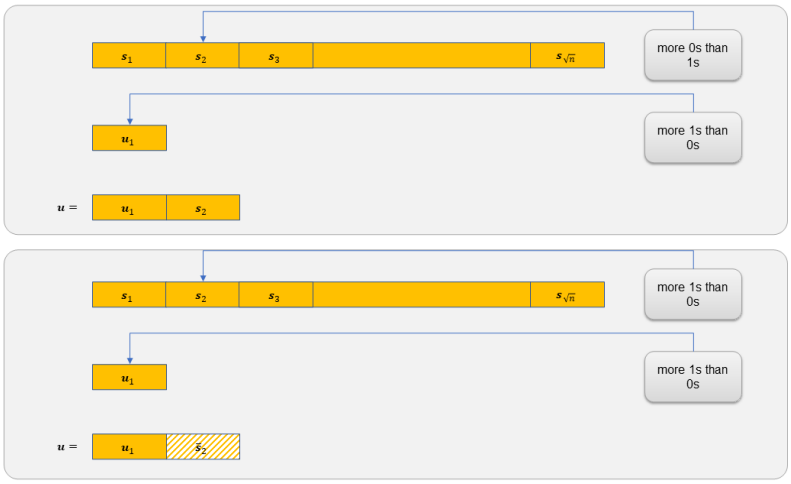
- Our techniques for converting a binary string of length n into strings that are both Dyck and belong to a B_h codebook have suboptimal

redundancy. We seek methods that can reduce our overhead and at the same time, offer low encoding and decoding complexity.

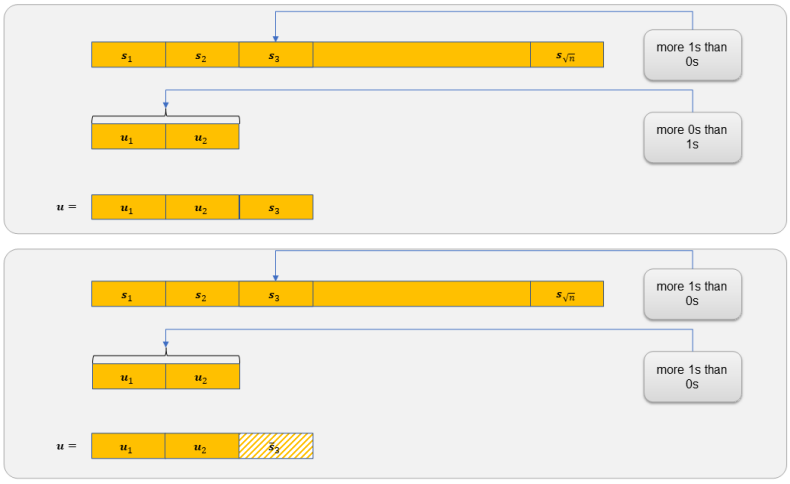
- In practice, one often encounters nonbinary alphabets, as polymers can be synthesized to have highly different masses and chemical properties. The question remains to generalize our approach for nonbinary alphabets. Furthermore, it is of interest to investigate such coding techniques for strings that have some form of balanced symbol contents or masses confined to a certain interval.

- It remains an open question to characterize all the missing mass errors that can be corrected by simply utilizing the Dyck, B_h properties of strings and the presence of both prefix and suffix masses.

- At this point, we have no efficient means for correcting mass reducing (or, mass increasing) substitution errors in our mixtures. A solution to this problem can have interesting and important implications in the field of polymer-based data storage.



(a) In the first step, we set $\mathbf{u}_1 = \mathbf{s}_1$. Without loss of generality, let us assume that $\mathbf{u}_1$ contains more 0s than 1s. Then, the next block $\mathbf{u}_2$ is set to $\mathbf{s}_2$ if $\mathbf{s}_2$ has more 1s than 0s. Otherwise, $\mathbf{u}_2$ is set to the complement of $\mathbf{s}_2$, $\bar{\mathbf{s}}_2$. To reconstruct a block $\mathbf{u}_2$, we either use the complement of $\mathbf{s}_2$ or the string itself, as determined by the added flag bit r_2.



(b) In the next step, the approximate balancing procedure continues as follows: Let us assume that $\mathbf{u}_1\mathbf{u}_2$ has more 1s than 0s. The next block $\mathbf{u}_3$ equals $\mathbf{s}_3$ if $\mathbf{s}_3$ has more 0s than 1s. Otherwise, $\mathbf{u}_3$ equals the complement of $\mathbf{s}_3$, $\bar{\mathbf{s}}_2$. The procedure is repeated for all subsequent steps.

Figure 3.1: Illustration of the approximate balancing procedure. The string $\mathbf{s} = \mathbf{s}_1\mathbf{s}_2 \dots \mathbf{s}_{\sqrt{n}}$ is decomposed into a concatenation of $\sqrt{n}$ blocks, each of length $\sqrt{n}$. The string $\mathbf{u} = \mathbf{u}_1\mathbf{u}_2 \dots \mathbf{u}_{\sqrt{n}}$ is constructed sequentially from the blocks $\mathbf{s}_1, \mathbf{s}_2, \dots, \mathbf{s}_{\sqrt{n}}$ by ensuring that the discrepancy between the number of 1s and 0s in the reconstructed string $\mathbf{u}_1\mathbf{u}_2 \dots \mathbf{u}_i$ is reduced or kept the same at every step i. Subfigure (a) depicts how the block $\mathbf{u}_2$ is chosen, while subfigure (b) depicts how the block $\mathbf{u}_3$ is chosen.

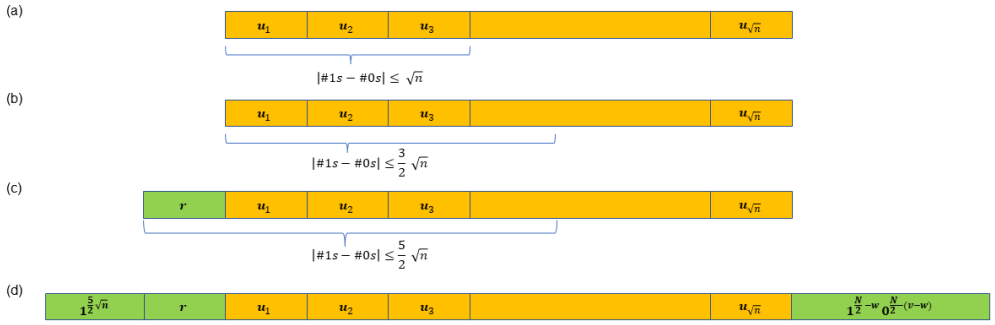


Figure 3.2: Further examples regarding the code construction. Subfigure (a) illustrates the finding of Claim 13: The discrepancy between the number of 1s and 0s in the string $\mathbf{u}_1 \dots \mathbf{u}_i$, $i \in [\sqrt{n}]$ is at most $\sqrt{n}$. Subfigure (b) illustrates the finding of Lemma 7: The discrepancy between the number of 1s and 0s in the string $u_1 u_2 \dots u_j$, $j \in [n]$ is at most $\frac{3}{2}\sqrt{n}$. Subfigure (c) illustrates an immediate extension of Lemma 7: The discrepancy between the number of 1s and 0s in the string $\mathbf{r}u_1 u_2 \dots u_j$, $j \in [n]$ is at most $\frac{5}{2}\sqrt{n}$. Recall that $\mathbf{v} = \mathbf{1}^{5/2\sqrt{n}}\mathbf{ru}$ is as defined in Claim 14, and that $w = \mathrm{wt}(\mathbf{v})$. Subfigure (d) depicts how adding a properly chosen prefix and suffix to the string ensures the Dyck property.

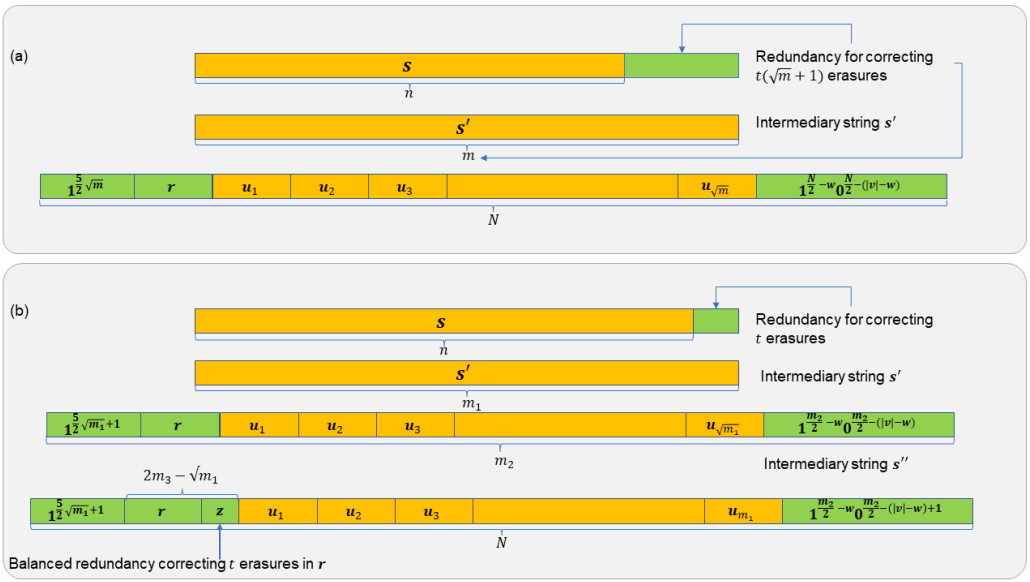


Figure 3.3: Illustration of the error-correction procedure. Subfigure (a) describes the one-step error-correction procedure. The string is encoded to handle the worst-case erasure scenario. Subsequently, the approximate balancing construction from Section 3.4 is performed to generate a Dyck codestring. Note that $\mathbf{v} = \mathbf{1}^{5/2\sqrt{m}}\mathbf{ru}$ is as defined in Claim 14, and it pertains to the intermediary string $\mathbf{s}'$. Here, $w = \mathrm{wt}(\mathbf{v})$. Subfigure (b) describes the two-step error-correction procedure. The string is encoded to be able to correct t erasures. The approximate balancing procedure from Section 3.4 is performed to generate a Dyck string. In the next step, redundancy is added to the substring $\mathbf{r}$ that can be used to correct t erasures. To preserve the Dyck property, "balancing redundancy" $\mathbf{z}$ is appended to $\mathbf{r}$, and additional bits are added to the prefix and suffix to obtain the desired codestring. Note that here $\mathbf{v} = \mathbf{1}^{5/2\sqrt{m_1}}\mathbf{ru}$ is as defined as in Claim 14, as applied to the intermediary string $\mathbf{s}''$ depicted in Subfigure (b). Once again, we have $w = \mathrm{wt}(\mathbf{v})$.

CHAPTER 4

COVID-19 TESTING

4.1 Introduction

In less than ten months since the first case reported in the Hubei province of China, Covid-19 had rapidly spread across all continents except Antarctica [8]. The disease has caused more deaths than Ebola, SARS, and the seasonal flu combined (reaching $5,000,000$ mortalities in November 2021), disrupted the global economy to an extent not seen since the Great Depression and altered the lives of hundreds of millions of people across the globe [9].

Many analyses associated with the Covid-19 pandemic have established that widespread population testing is key to effectively containing outbreaks of this and other infectious diseases. In May 2020, the United States was able to test around $150,000$ people per day, while countries that had managed to keep the outbreak under control then, such as Germany and South Korea, had performed millions of tests during the same stage of the spread of the disease. Although there is no general consensus on the exact number of individuals that need to be tested, most experts agree that the reported numbers were highly inadequate [59]. Some universities, such as Yale University and the University of Illinois, had a biweekly test schedule in place for all individuals accessing school property [60]. This is believed to be a sufficiently large-scale testing protocol that allowed the institution to safely operate.

To address the need for sustainable high-frequency population testing, a number of countries and states proposed and implemented *group testing schemes* in which genetic samples from different individuals are pooled together in a manner that incorporates thresholded real time reverse transcription polymerase chain reaction (RT-PCR) fluorescence signals into the

testing scheme.[1]

Group testing (GT) is a combinatorial screening method introduced by Dorfman [62] for identifying small groups of soldiers infected by syphilis. His scheme, known as single-pooling, consists of mixing blood samples from five soldiers at a time, and running one test for each pool. For positive test outcomes, the soldiers involved in the test are examined individually in a second round to determine who has the disease. For negative outcomes, all subjects involved are declared healthy and removed from future testing schedules. Given a relatively small number of infected individuals in a population, this scheme provides a significant reduction in the number of tests required when compared to individual testing [63]. The scheme proved ineffective for its original task as blood sample pooling dilutes the resulting sample to a point below the sensitivity of the tests used.

A number of recent reports suggest using Dorfman's or other mostly off-the-shelf GT schemes for Covid-19 testing [10–19]. Most of the proposed schemes do not incorporate relevant biological priors or exploit the highly specific measurement and noise properties of the RT-PCR method in their testing schemes. We argue this is a significant detriment, as in order to properly execute the effort and avoid dangerous failures, testing schemes should to be guided both by mathematical considerations as well as social, clinical, and genomic side information.[2] This suggests designing Covid-19 group testing schemes that carefully address the following issues:

1. **Selection of adequate primers.** As stated in the CDC SARS-CoV-2 testing guidelines [66], only two primers are recommended for use in the USA for RT-PCR reactions, selected from the N open reading frame (ORF) of the SARS-CoV-2 genome. It is often hard to predict which regions will have small mutations and it is currently not known how fast the N regions and other primer regions chosen by various countries mutate and how these mutations affect the PCR protocol. To determine

[1] A recent ordinance issued by the governor of Nebraska [61] recommends using group testing for widespread screening for Covid-19, while group testing methods are employed in part in Israel.

[2] As the disease affects people from different age groups, ethnicities and regions in a highly disparate manner [64]; it has also been reported that mortality rates across different populations can deviate by as much as two orders of magnitude [65]. The World Health Organization (WHO) has also repeatedly issued testing guidelines that suggest "suspect influenza should be tested with consideration for geographical, gender and age representativeness" in order to contain the spread of the disease in real-time.

the influence of mutations one first needs to determine which regions will remain mostly unaffected by mutations, determine the melting temperatures of the primers [67] and their binding affinity to the mutated reference regions. For this purpose, the recent work [68] may be used to guide the primer selection process, while actually recorded mutated N primer regions may be used to estimate the failure rates of the individual PCR tests or model the errors in group PCR protocols due to mutations. These issues are examined in [69].

2. **Selection of (near-)optimal sample mixing strategies with priors.** If not properly designed, GT schemes may lead to errors. Since some individuals, such as health workers, may harbor multiple strains of the virus, and since clinical priors are often readily available (e.g., symptom charts, chest X-ray images) the sampling and mixing approaches should be carefully designed to include the right number and combinations of subjects in order to minimize test errors. This is a complicated issue that will be examined elsewhere.

3. **Use of quantitative test outcomes.** RT-PCR experiments provide significantly more information about the viral load or number of infected individuals within the group rather than just a simple binary answer, "no infected samples" or "at least one infected sample". Except for a handful of works proposing to use quantitative RT-PCR through *compressed sensing* (CS) [70–72], most reported Covid-19 GT schemes assume binary test outcomes (among them the scheme used in Israel and described in [73]). Furthermore, practically tested schemes operate in a nonadaptive setting, which is suboptimal and not justified for large-scale testing strategies which use a limited number of PCR machines. Another important issue is that heavy hitters (individuals with very high viral loads) can "mask" individuals with smaller loads which makes the use of quantitative information difficult [74, 75]. This masking phenomenon, as well as saturation effects present in RT-PCR outputs, can make CS approaches highly susceptible to errors. The focus of this work is to develop schemes that can address these issues in a simple and efficient manner. Consequently, our main results pertain to scalable, adaptive and semiquantitative testing methods that can efficiently correct errors that are specific to RT-PCR systems. We also

describe techniques that can handle heavy hitters.

4. **Incorporation of social/contact network information.** Due to the highly heterogeneous response of individuals to the infection, diverse infection rates across different geographic and communal regions, the best testing practices have to be guided by infection risk assessments and scores. Such "network-guided" schemes are currently not well studied in the GT and CS literature, except for some recent work that uses community information to model correlations and reduce the number of tests required [76]. Instead, [69] proposes to identify highly infected communities and their neighboring communities rather than all individuals in each infected community. In this case, GT is used jointly with other commonly employed mitigation strategies such as quarantine.

We argue that the availability of (semi)quantitative test outcomes and the use of adaptive strategies can greatly increase the efficiency and scalability of Covid-19 testing schemes. In that direction, we generalize the Semiquantitative Group Testing (SQGT) strategy [77–79] to an adaptive setting and devise simple probabilistic adaptive GT methods[3] and worst-case adaptive GT schemes that take the specific measurement data noise and quantization properties into account. The SQGT scheme assumes that one cannot tell the exact viral load or number of infected individuals in a pool but only an interval in which the load or number of defectives lies. The setting is a generalization of GT that includes more than two outcomes, or a quantized version of the adder channel/CS approach [53, 82]. It also represents a generalization of the setting [83] in which only saturation effects are taken into account within the adder model. It is worth pointing out that this is also the first approach that uses actual RT-PCR features and also postulates rigorous models that allow for relating viral loads to actual fluorescence values and analyze the testing schemes rigorously. Other methods that will be reviewed

[3]The quantifier "probabilistic" may refer to either the individuals being ill according to some probability distribution (usually iid Bernoulli(p), or generalized binomial [80] or Poisson [81], where the number of infected individuals has a right-truncated Poisson distribution with parameter $\lambda(n) = o(n)$; Dorfman's scheme falls into the first category) or, the test matrices having entries dictated by a probability distribution (again, usually iid Bernoulli(q)). Some papers refer to the former setting as "group testing with probabilistic priors" and the latter setting as "group testing with probabilistic tests." Which paradigm we refer to will be clear from the context.

in later sections either completely ignore the RT-PCR measurements or do not properly justify or analyze their proposed models.

For an illustration of the differences between various testing schemes (group testing, additive tests, additive tests with saturation, and general SGQT), the reader is referred to Figure 4.1.

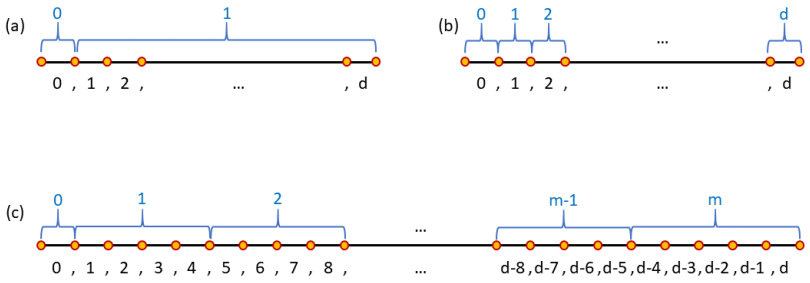


Figure 4.1: Subfigure (a) illustrates the *classical GT* framework. Here, 0 corresponds to a test outcome that is indicative of no infected individual being present, while 1 corresponds to an outcome indicative of at least one infected individual. Subfigure (b) represents an *additive test output* model, in which the underlying assumption is that one can tell the exact number of infected individuals in a test. An instance of the *general SQGT* is illustrated in Subfigure (c). In this case, the test outcome $\lceil \ell/\tau \rceil = i$ for $i > 2$ indicates that $(i-2)\tau \leqslant \ell \leqslant (i-1)\tau$ defectives are present. When the number of defectives detected is $> (m-1)\tau$, the test reports m. When $\tau = 1$, (c) represents an adder model with saturation.

For **the worst case setting**, in which we assume a known number of d defectives, but make no assumptions about how they are distributed, and where one can tell if zero defectives were present, or the number of defectives is nonzero and lies in one out of m consecutive intervals of length τ, the number of tests per defective roughly equals

$$\frac{\log(n/d) + \log\log(m+1)}{\log(m+1)}.$$

The savings in the number of tests as compared to the classical GT setting provided by the increased resolution of the levels is $\log(m+1)$-fold, which even for 7 levels amounts to three-fold savings. Clearly, one has to be able to properly calibrate the RT-PCR readouts and determine adequate thresholds in order to take full advantage of the scheme. This issue will be addressed

in Section 4.2.

For **the probabilistic setting**, where each item is assumed to be independently defective with some probability, [69] provides results that include simple-to-implement algorithms for adaptive testing that involve two thresholds and two test stages, and are also capable of handling heavy hitters (i.e., individuals with high viral loads that may mask other individual's presence, provided these individuals are not too common).

The remainder of this chapter is organized as follows. Section 4.2 provides an overview of the PCR, RT-PCR and the Real Time (Quantitative) PCR techniques. The section also addresses key issues that impede the amplification efficiency of the methods currently used for Covid-19 testing and introduces several practical noise models. Section 4.3 describes various GT approaches and assesses their utility for Covid-19 testing. Section 4.5 describe the results. Section 4.4 describes a probabilistic version of a SQGT model, simplified to account for two rounds of testing and two test thresholds only. This section also introduces test schemes that aim to identify highly virulent individuals, termed *heavy hitters*. Section 4.5 introduces a new worst-case adaptive SQGT technique that is near-optimal and describes a noise model termed the *birth-death chain model*. Section 4.6 concludes the chapter and discusses future work.

4.2 Background

We start our exposition by describing the real-time reverse transcription (RT-PCR) testing mechanism. DNA has a double helix structure and both strands in the helix are composed of periodic sugar and phosphate groups to which one of four different bases is attached, namely A, T, C, and G. A sugar, phosphate, and base are jointly referred to as a nucleotide. As the sugar is asymmetric in terms of the placement of its carbon atoms with respect to the position of binding to the phosphates, the two strands of the DNA have two different directions: One runs from the 3' to 5' carbon end, while the other runs from the 5' to 3' carbon end. The two strands are held together through the stacking of bases and the hydrogen bonds that exist between them. The pairing rule is dictated by Watson-Crick (WC) complementarity asserting that only (or with overwhelming probability) the bases A and T

and G and C bind to each other, respectively.

4.2.1 Reverse Transcription PCR

The Reverse- Transcription PCR (RT-PCR) technique is used to identify/amplify RNA strands. Since RNA is single-stranded and hence an unstable molecule, RT-PCR first converts the target RNA into its complementary double-stranded DNA (cDNA, as illustrated in Figure 4.2) and then performs amplification using the standard PCR technique. Note that RNA has three of the same building blocks as DNA, namely A, C, and G, but instead of T (encountered in DNA), RNA contains U (Uracil).

Conversion of RNA into cDNA is accomplished through the use of the reverse transcription (RT) enzyme that stitches together "free" nucleotides A, T, C, and G together, in the presence of primers that are complementary to a specific part of the target RNA sample (see Figure[4] 4.2). Since RNA is inherently single-stranded, the primers have an affinity to attach to the complementary RNA regions, recruit the RT enzyme and thereby initiate synthesis. The process proceeds through two steps: In the first step, the first-strand cDNA is created using the single-stranded RNA as a template. In the second step, the second-strand cDNA is formed by using the first-strand cDNA as template. Consequently, the product cDNA represents an accurate replica of the original RNA content, converted to the DNA alphabet.

The test results are usually compared to a control as a means to assess the quality of the experiment. RT-PCR is used to detect RNA viruses, i.e., viruses whose genomic content is stored in RNA rather than DNA. SARS-Cov-2, the virus causing Covid-19, is an RNA virus, as is for example the HIV virus that causes AIDS. For viral detection, the first step of testing involves isolating genomic RNA by breaking the viral membrane, but this and other processes that lead to actual sample isolation will not be discussed in this short review.

The Covid-19 detection and amplification process relies on standard RT-PCR methods and RT-PCR and its quantitative version described next.

[4]Reverse Transcriptase image is from Wikipedia.

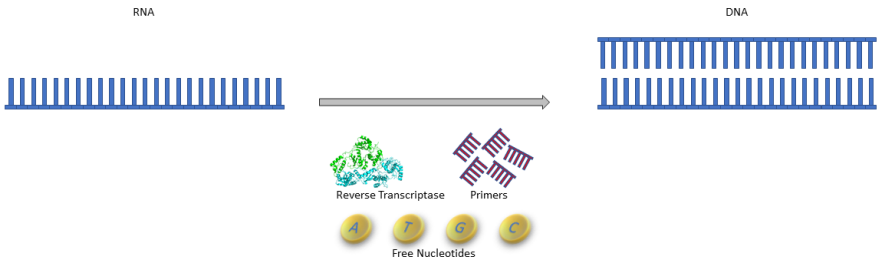


Figure 4.2: Reverse transcription for converting viral RNA into cDNA.

4.2.2 The Polymerase Chain Reaction (PCR)

The polymerase chain reaction (PCR) is used to amplify specific segments of the DNA strands in order to enable a downstream analysis of the segments or to detect the presence of specific DNA content. The operating principles of the PCR process are illustrated in Figures 4.3 and 4.4. A thermal cycler uses the target DNA, specific primers (short DNA segments that initiate the replication process by allowing the polymerase to bind to the DNA), the taq polymerase (which actually performs DNA replication after the primers get attached), and free A (adenine),T (thymine), C (cytosine) and G (guanine) nucleotides needed to amplify the segment of interest through *repeated cycles* that involve the following steps: *DNA denaturation, annealing, and extension.*

1. DNA Denaturation: The DNA sample to be amplified or detected is first heated to 96 °C. At this temperature, hydrogen bonds between the bases across the two strands break, producing two complementary single-stranded DNA fragments.

2. Annealing (Hybridization): The sample is subsequently cooled to 55 °C. This allows the primers to bind to their WC complementary segments on the two single-stranded DNA targets. The primer that binds to the forward strand is referred to as the *forward primer* while the one that binds to the reverse strand is referred to as *reverse primer*.

3. Extension: The sample is heated to 72 °C to enable the taq polymerase to extend the primers to form two complete copies of the original double-stranded DNA molecule.

Under ideal conditions, at the end of the Extension step of a cycle, the amount of target DNA doubles. This setting is illustrated in Figure 4.3. However,

due to several factors [67] including the efficiency of denaturation, primer annealing affinity, polymerase binding strength, and others, the DNA content may not double during each cycle. For example, denaturation requires heat-

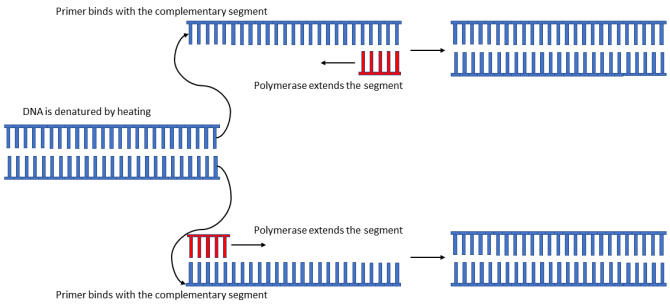


Figure 4.3: Polymerase chain reaction: In any given cycle, the DNA strands in the sample are first denatured into single strands. The two single strands are then extended to form complete DNA double-helixes. The primers (short DNA fragments) are attached to the single-stranded DNA such that extension is facilitated in the 3'-5' direction. At the end of each ideally executed cycle, the number of DNA strands in the sample doubles.

ing the sample to a higher temperature which by itself may cause oxidative and other damage to the DNA being amplified. The efficiency of denaturation is measured in terms of the concentration of viable single-stranded DNA present after heating.

During the primer annealing stage, single-stranded DNA strings previously denatured can anneal back, therefore prohibiting access to the primer segments. The primer annealing efficiency is captured by the proportion of single-stranded DNA with bound primers.

When the polymerase binds to the DNA-primer complex it forms a potentially unstable tertiary complex in which the polymerase can disassociate in a stochastic manner. The polymerase binding efficiency is captured by the fraction of tertiary products in the assay. The tertiary complexes formed during the early stage of a cycle are more likely to result in complete double-stranded DNA compared to those formed in a later stage of the cycle, due to cycle timing issues. This effect is captured through what is known as the extension efficiency of PCR.

These effects jointly contribute towards the reduction of the average effi-

ciency of DNA amplification, which goes down from the expected doubling factor to some value < 2, usually written as $(1 + \eta)$, where η is referred to as the cycle efficiency. The doubling of the target material at every cycle corresponds to $\eta = 1$. At the end of i cycles, a sample with concentration x DNA strands is amplified to a sample with concentration $x(1 + \eta)^i$. More precisely, the cycle efficiency depends on the cycle number. Consequently, a more accurate amplification model should use the factor η_j for cycle j, so that the amplified concentration after the i^{th} cycle reads as $x \cdot \Pi_{j=1}^{i}(1 + \eta_j)$. It is also known that η_j decreases with j, which may be attributed to the fact that the primers used for amplification are more and more integrated into the DNA products and that the efficiency of the polymerase decreases. At the same time, for a small number of cycles (usually $i \leqslant 10$) the DNA products are hard to detect. As a result, it is a common practice to run $30 - 40$ cycles of PCR, depending on the expected original concentration of the double-stranded DNA to be amplified.

Note that the polymerase can also be active at temperatures below 72 °C, thereby initiating the extension process. However, the polymerase is non-specific at lower temperatures and leads to amplification of nonspecific DNA strands. The high concentration of the stronger and more stable GC bonds in the DNA strands hinders effective denaturation at 96 °C. Regions with high GC bond concentration also form secondary products that prevent primer bonding [84]. These phenomena all jointly contribute to "noise" in the amplification PCR process which is not associated with the cycle efficiency. Additional sources of noise such as CCD thermal noise and shot noise can lead to a further decrease in the reliability of data points at low signal levels [85].

Also, primers may fail to attach to the DNA if the corresponding DNA primer regions contain mutations (indels or point mutations). Since the error is caused by the actual DNA sample strand, and not the PCR process, this phenomenon should not be considered as part of the PCR noise model. The results of a simulation that studies the effect of mutations along the primer region on PCR amplification are described in [69], using a collection of real genomic datasets retrieved from the GISAID database [86].

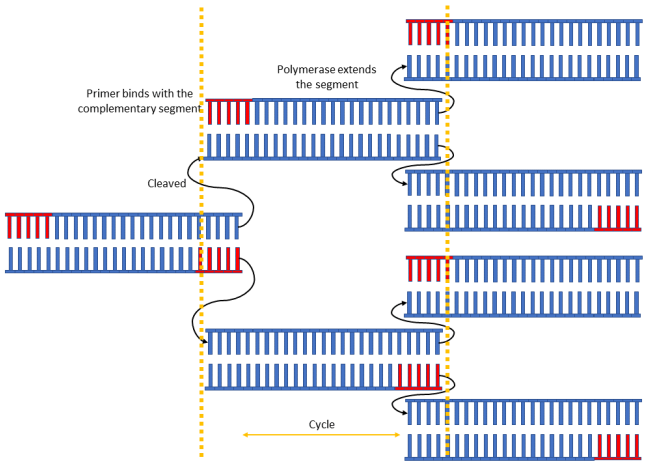


Figure 4.4: Under ideal conditions, every cycle of the PCR process should double the DNA content. Due to various factors, described in the main text, not every cycle may result in twice as many strands and an averaged efficiency factor $\eta < 1$ is used to describe the growth rate of the PCR product.

4.2.3 Quantitative (Real-Time) RT-PCR

Quantitative Real Time PCR (qPCR) is a technique used for precise analysis of viral and bacterial samples. As implied by its name, qPCR allows for the amplification process to be monitored in real-time. This is achieved by introducing fluorescent labels into the DNA products and recording the change in fluorescence with an increasing number of cycles (which also allows for estimating the number of cycles needed to detect an appropriate product). The result of a qPCR experiment is usually given in terms of an *amplification curve* (an example of such a graph is shown in Figure 4.7, where real measurements are approximated by piecewise polynomial fragments of degree $\leqslant 10$). The amplification curve plots the normalized (relative) fluorescence ΔR_n against the cycle number. The fluorescence increases with the increase in the target genetic material with every cycle until the fluorescence saturates. The cycle number for which the fluorescence crosses the detection threshold (which can be defined in several ways) is referred to as the *cycle threshold*, and denoted by C_t. Note that C_t is inversely proportional to the concentration of the target material in the sample: A low C_t value indicates a higher concentration of the sample we wish to detect, while a high C_t value indicates a low concentration of the same or spurious amplification results. The slopes of the curves most often show very small variations with the con-

centration of the subject but may potentially be used as further indicators of the sample load.

Real-time or qPCR is usually performed using one of the following two approaches:

- *Dye-based qPCR.* The dye-based method uses dyes that *only* fluoresce when bound to double-stranded DNA. Thus, at the end of each extension stage, the fluorescence increases (see Figure 4.5). The chemistry of the dyes used helps in distinguishing desired and undesired products. However, the dye-based method is often nonspecific, thereby inaccurately quantifying genetic material that is not of interest. As a result, this approach requires highly selective primers and other additional controls to provide accurate amplification curve results.

- *Probe-based qPCR.* In this technique, primers specific to the target DNA include two molecules, a fluorescent reporter dye and a quencher on its two ends. When the quencher is in close proximity to the fluorescent dye, the former molecule inhibits (quenches) the fluorescence of the latter. This is usually the case when the primer is not bound to the target (see Figure 4.6). However, when the primer is hybridized to the target and the polymerase extends the primer segment, the quencher and reporter separate out and the dye is cleaved and displaced. In its free form, it fluoresces which leads to detectable signals.

4.2.4 Amplification Curves and the Viral Load

From Figures 7 and 4.8, and as already discussed in the previous section, it is clear that one can *estimate upper/lower bounds* on the viral load of an individual by observing the C_t value and the slope and saturation point of the amplification curve. It is important to point out that the viral load of individuals may vary up to five orders of magnitude, as shown in the recent study [87]. Viral loads in infected individuals tend to follow a "typical" inverted-V dynamics shown in Figure 4.9. There, it can be seen that an individual tested $3-5$ days after the infection may have a viral load that is large enough to mask any other individual tested by the same test under the GT framework. This is a sensitive issue for SQGT schemes as the C_t curves

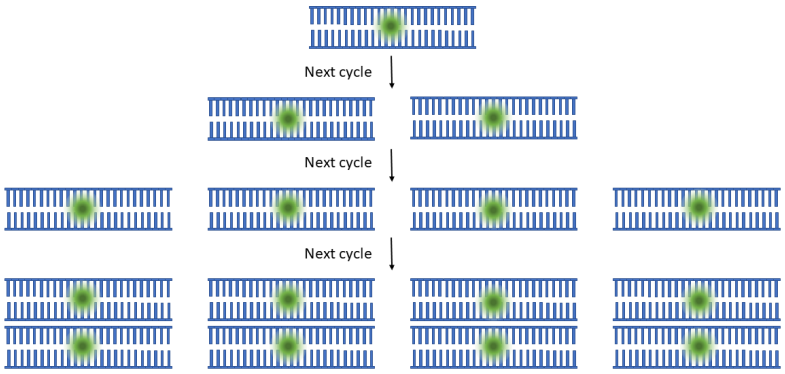


Figure 4.5: Dye-based qPCR: The dye attaches to the double-stranded DNA formed at the end of the extension stage and fluoresces. Thus, the fluorescence measured increases with the number of cycles.

may have multiple interpretations: As an example, the same C_t value may correspond to $10 - 100$ individuals tested $5 - 6$ days after infection or one individual tested 3 days after infection. There are multiple possible ways to mitigate this problem: First, given that high viral loads very often positively correlate with observable disease symptoms [88],[5] asking individuals about symptoms before scheduling the tests (as is, for example, done at UIUC [89]) allows one to determine if the individual should be group-tested or not. Another approach is to perform adaptive testing where samples with large viral loads are subjected to additional screenings, as is done in one of our proposed methods. Specialized testing strategies for pooled measurements with high viral loads can also be devised using heavy-hitter detection methods [90].

As an abstraction, and only for our worst-case analysis we, assume that each individual is represented by a viral load equal to the expected value over the tested population. In this case, the test outcome can be translated into an interval in which the number of infected individuals lies. Hence, the assumption in this case is that one can convert C_t values into a rough estimate of the number of infected people in the test. For probabilistic testing, we do not have to rely on such assumptions as the testing scheme itself can be easily adapted to handle heavy hitters.

[5]According to this study, among the set of infected patients, those who exhibited "severe" symptoms had *significantly lower* $C_t = C_t(\text{sample}) - C_t(\text{reference})$ values than those who exhibited "mild" symptoms.

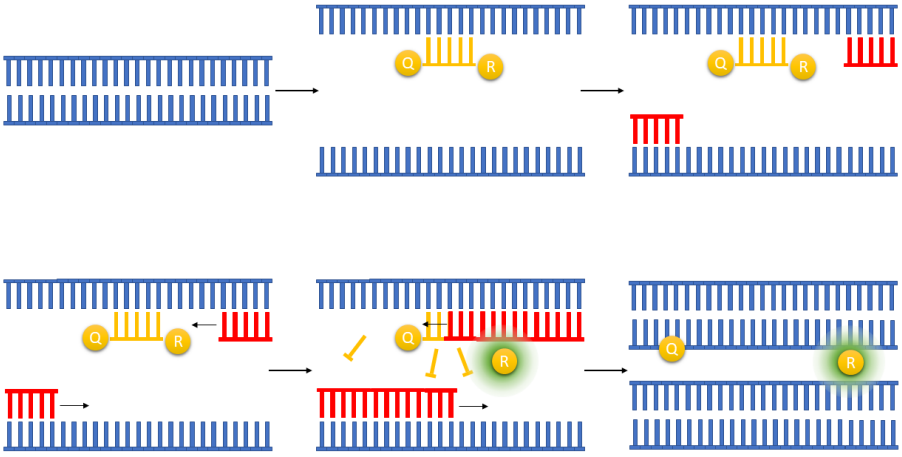


Figure 4.6: Probe-based qPCR: When DNA is denatured, a primer specific to the target DNA is attached to a single strand. The primers are then attached and extended by the polymerase. During the extension, the probes are cleaved and the reporter dye is no longer in the proximity of the quencher molecule, which enables it to fluoresce.

4.3 Basics of GT

In what follows, we provide concise overviews of all relevant GT schemes used or proposed for potential use for Covid-19 testing: (1) Classical nonadaptive and adaptive GT; (2) Nonadaptive SQGT; (3) Threshold GT; (4) Compressive sensing (CS); (5) Graph-Constrained GT. For all these methods, we describe their potential advantages and drawbacks and then proceed to introduce a new method, which we refer to as adaptive SQGT. Adaptive SQGT with a "curve fitting"-based noise model appears to provide the theoretical state of the art GT results for qPCR test models and is the focus of our subsequent discussions.

4.3.1 Nonadaptive and Adaptive GT

The assumptions are as follows: In a group of n individuals, there are d infected people. When a subset of people are tested, the result is positive (e.g., equal to 1) if at least one person in the tested group is infected, else the test result is negative (e.g., equal to 0). Such a testing scheme is referred to as *binary*, as the outcomes take one of two values (see Figure 4.1 (a)). GT aims

to find the set of all infected people with the fewest number of binary tests possible and may use nonadaptive and adaptive tests. In the former case, all tests are performed simultaneously and the outcome of one test cannot be used to inform the selection of the individuals for other tests. In the adaptive setting, one can use multiple stages of testing and different combinations of individuals to best inform the sequentially made test choices.

When $d \ll n$, it is well known that $\Omega(d \log(n/d))$ number of tests are required to find all infected individuals. Furthermore, it was shown in [91] that for NAGT, at least $\Omega(d^2 \log(n)/\log(d))$ tests are required in the worst-case model. For the same parameter regime, there exist explicit nonadaptive schemes that require $\mathcal{O}(d^2 \log n))$ tests to find the infected group [92]. A four-stage adaptive scheme that uses an optimal number of tests that meets the lower bound was recently described in [93]. Of special interest is the classical binary search result of [94] which established an elegant adaptive scheme that differs from the information-theoretic limit only by an additive $\mathcal{O}(d)$ term.

Despite the many proposed applications of this model to Covid-19 testing, it is obvious from the previous discussion that the GT measurement outcomes do not fully use the actually available qPCR results. One could argue that the fluorescence exceeding the detection threshold may correspond to the test outcome 1, but clearly, significantly more information is available as the detection threshold depends on the concentration of the viral cDNA and hence the number of infected individuals. This motivates using a more quantitative GT approach, already introduced under the name of SQGT.

4.3.2 Nonadaptive SQGT

In SQGT, one is given a collection of thresholds $0 = \tau_1 < \tau_2 < \cdots < \tau_r$, and the outcome of each test is an interval $(\tau_i, \tau_{i+1}]$, where $0 \leqslant i \leqslant r-1$. The outcome of an experiment cannot specify the actual number of infected individuals but rather provides a lower and upper bound on that number, τ_{i-1} and τ_i, respectively. If $\tau_i = \tau_{i-1} + 1$ for all values of i and $r = d$, the scheme is referred to as additive GT, or the adder model [53, 82]. The two models are depicted in Figure 4.1 (b) and (c). The additive test model described in [53] requires $2 \cdot (n/\log\, n)$ tests to determine all possible infected

individuals, for $0 \leqslant d \leqslant n$.

Another special SGT case of interest assumes that the test results are additive up to some threshold τ and after that, they saturate [83] (see Figure 4.1). This model is of special interest for Covid-19 testing as it takes the warm-up/saturation information into account and, in addition, under a proper noise model, captures the fact that amplification graphs have different C_t values determined by the concentration of the viral load (an hence the approximate number of infected individuals). Furthermore, one can argue that the RT-PCR fluorescence intensity information is inherently semiquantitative [77] as the fluorescence levels and C_t values can be placed into bounded bins determined by the number of cycles. This observation is explained in more detail in the next section, along with new theoretical results pertaining to adaptive SQGT schemes with appropriate noise models (refer to Figure 4.10).

4.3.3 Threshold GT

An extension of the GT problem was introduced by Damaschke in [95]: In this setting, if the number of defectives in any pool is at most the lower threshold $\ell > 0$, then the test outcome is negative. If the number of defectives in the pool is at least the upper threshold $u > \ell$, then the test outcome is positive. However, if the number of defectives in the pool is between u and ℓ, the test outcome is arbitrary (i.e., either 0 or 1). Thus, the algorithms for Threshold GT are designed to handle worst-case adversarial model errors. Note that when $\ell = 0$, and $u = 1$, Threshold GT reduces to GT. It is known that for nonadaptive threshold GT, $\mathcal{O}(d^{g+2}(\log d)\log(n/d))$ tests (where $g = u-\ell-1$) suffice to identify the d infected individuals [96].

The Threshold GT model is partly suitable for modeling the qPCR process, as the lower threshold can obviously assume the role of the fluorescence-based detection threshold, $\ell = C_t$; unfortunately, due to the saturation phenomena, a specialized choice for the upper threshold u does not allow one to accurately assess the number of infected individuals in the pool. The "in-between" threshold results also make the simplistic assumption that despite the observed fluorescence value being closer to the upper threshold, one can still call the outcome negative (and similarly for the small fluorescence levels and the lower threshold).

4.3.4 Compressive Sensing

In compressive sensing, the defectives are represented by nonnegative real-valued entries. Thus, quantitative GT represents a special instance of compressive sensing. Compressive sensing assumes that one is given an unknown vector $\mathbf{x} \in \mathbb{R}^n$, in which only $d \ll n$ entries are nonzero. The vector $\mathbf{x}$ is observed through linear measurements formed using a measurement matrix $M \in \mathbb{R}^{m \times n}$, leading to an observed vector $\mathbf{y} = M\mathbf{x} + \mathbf{n}$, where $\mathbf{n}$ is the measurement noise (usually taken to be Gaussian $\mathcal{N}(0, \sigma^2)$). For noiseless support recovery, $m = \mathcal{O}(d \cdot \log n)$ measurements are sufficient. For support recovery in the noisy setting, it was shown in [97] that the minimum signal-to-noise-ratio required is $\Omega(\log n)$, and that the number of tests must scale as $\Omega(d \log \frac{n}{d})$ for exact recovery. Compressive sensing reconstruction is possible through linear programming methods or low-complexity greedy approaches [98–100].

The recently proposed Tapestry method [101] combines group testing with compressive sensing and uses combinatorial designs (i.e., Kirkman systems) to construct the measurement matrix. However, the approach does not account for several practical features inherent to quantitative PCR. Although Tapestry proposes a model that involves multiplicative noise and converts it into additive noise through the use of a logarithmic function, it is still *inherently linear:* Tapestry is based on a CS framework, which is additive and applies *to viral loads*. But as seen from the previous discussion, PCR measurements report intersections of fluorescence level curves and a given threshold, and these values are nontrivial nonlinear functions of the viral load. Additionally, although the compressive sensing measurements used in the work are assumed to correspond to C_t values, no thresholding is used to model the actual practical process of generating the same.[6] Also, this and all other methods do not account for the stochasticity of the PCR measurements and the fact that different lab protocols may lead to different C_t values when presented with the same sample mixture. The CS methods in [101] rely on Gaussian assumptions for the cycle inefficiency exponent and do not take into account that the efficiency decays with the number of cycles and with the number of potential mutations in the primer regions (see also our anal-

[6]For many related questions arising in the context of group testing microarrays and quantized compressive sensing, the interested reader is referred to [102–104].

ysis in Section 4.2). As many other quantitative methods, it also appears vulnerable to heavy hitters.

CS-based and many other proposed Covid-19 testing methods also do not take into account the fact that the number of RT-PCR machines/staff members is limited in virtually all test settings.[7] The unavailability of arbitrary number of PCR machines inherently suggests using adaptive testing strategies. Adaptive quantitative testing schemes for Covid-19 were reported in [105]. There, the same problem setup as in [101] is used to postulate an additive viral load model in the absence of noise. The new contribution of the work is a proposal for a two-stage testing scheme that bears a small resemblance to our methods insofar as we also propose two-stage adaptive pooling schemes. However, these techniques and the model used are different from ours since [105] employs a combination of maximum likelihood and maximum-*a-posteriori* estimators to determine the infected individuals in the second stage, while we employ zero-error GT and SQGT techniques to find *all* infected individuals. Additionally, while [105] reports the number of tests and conditional false positive and conditional false negative rates for the simulation experiments, we supplement our new tailor-made modeling and testing schemes with an in-depth theoretical analysis and performance guarantees.

Nevertheless, there seem to be multiple advantages of CS methods for Covid-19 testing: One should be able, in principle, to recover not only the infected individuals but their viral loads as well (it still remains to be seen as such approaches are feasible as reported experiments use controlled concentrations of viral loads [101]). In particular, integer and nonnegative CS testing, along with quantized CS approaches can impose model restrictions on such testing schemes [103, 106] to render them more suitable for the problem at hand.

4.3.5 Graph-constrained GT

Let $\mathcal{G} = \{V, E\}$ be a graph with vertex set V, $|V| = n$, and edge set E, representing a connected network of n people out of whom d are infected. In graph-constrained GT, vertices participating in the same test are restricted

[7] PCR tests are performed on samples typically organized within 96 wells, each of which can be used for one (group) test.

to form a path in the graph [107]. This model is relevant as it can be adapted to require that only individuals that did not have contacts with each other are tested together (one only has to apply the problem to the dual of the contact graph used in Covid-19 testing). This allows us to identify individuals that fell ill in an "independent" fashion rather than through contact with each other. If $T(n)$ denotes the mixing time of the random walk on the graph is used to denote the ratio between the maximum degree and the minimum degree of the graph, then no more than $\mathcal{O}(d^2 \cdot T^2(n) \log(n/d))$ tests are required to find the set of infected vertices. For example, a complete graph ($T(n) = 1$) requires no more than $\mathcal{O}(d^2 \cdot \log(n/d))$ tests since it corresponds to the classical GT regime. Unfortunately, graph-constrained GT requires a significantly higher number of tests than classical GT methods as the tests are inherently restricted. As a result, despite the fact that this scheme is a natural choice for problems such as network tomography where these constraints need to be satisfied, it is not a proper choice for Covid-19 testing. Another "community-constrained" GT scheme (although without an underlying interaction graph) was recently proposed in [76] and is discussed in the next subsection.

4.3.6 Community-aware GT

Several lines of work have focused on what is now known under the name of *community-aware GT*. In [76, 108], the authors leverage correlations arising due to the presence of community structures to reduce the number of tests and increase the reliability of testing. More precisely, they assume that a community of n members has $d \ll n$ infected individuals and that the population is partitioned into F families. In the combinatorial infection model, it is assumed that d_f families have at least one infected individual and that all the members of the remaining families are infection-free. An infected family indexed by j is assumed to have $d^{(j)}$ infected members so that $d = \sum_{i=1}^{F} d^{(j)}$. The testing scheme can be succinctly described as follows: A representative individual from each family is selected uniformly at random. The representative community members are tested using either an adaptive or a nonadaptive GT algorithm. Family members whose representatives tested positive are tested individually. Members from the remaining families are tested together

using either an adaptive or a nonadaptive GT scheme. The first approach proposed in [76] did not account for inter-community interactions, but that issue was subsequently addressed in [109].

In a related line of work, the authors of [110] establish how correlations in samples that arise due to the ordering of the tested individuals in a queue save in terms of pooled testing costs. In particular, the authors assume that in the first stage of Dorfman's testing scheme, the samples that are pooled and tested together are correlated. They model the correlation through the use of a random arrival process [110]. The expected number of tests required to identify all infected individuals for their modified Dorfman scheme is compared against the expected number of tests required by the classical Dorfman testing scheme in which samples that are tested together are picked at random. The authors show that under certain conditions, the expected number of tests required by the modified Dorfman testing scheme does not exceed the expected number of tests required by the original scheme. Under additional conditions, the authors derive a closed form expression that captures the savings available for correlated samples. Furthermore, [110] considers an underlying social contact graph, and proposes an hierarchical agglomerative algorithm to identify individuals to be pooled together in the first stage of the modified Dorfman testing scheme. This line of work is closely related to the problem of identifying bursts of defectives, first introduced in [111] and analyzed for the case of a single burst.

Before proceeding with the original contributions, we remark that all the above GT techniques and scheduling models have *probabilistic counterparts* in which each individual is assumed to be infected with the same probability p [62] or members of different communities are infected with different probabilities, p_i, $i = 1, \ldots, F$ [80]. The latter setting is especially important when prior information about the individuals is known (for example, their risk groups, potential symptoms etc.). For an excellent in-depth review of these and some other GT schemes, the interested reader is referred to [63].

4.4 AC-DC: The Probabilistic Setting

Two adaptive SQGT schemes are introduced, one which is suitable for probabilistic testing and another one that is worst-case and nearly-optimal from

the information-theoretic perspective. In the former case, considered in this section, a simple two-stage testing scheme is designed and analyzed with the goal of enabling practical implementations of adaptive SQGT. The results are described for two thresholds only, but a generalization is straightforward. This scheme also allows for incorporating heavy hitters into the testing scheme, which is of great practical relevance. In the worst-case, which is considered in the section to follow, the schemes extend the ideas behind Hwang's generalized splitting [94] in two directions that lead to algorithms using what we call *parallel* and *deep search*, respectively. In both settings, the outcomes of the first round of testing inform the choice of the composition of the test in the rounds to follow. The methods are collectively referred to as the AC-DC schemes, in reference to the use of the information provided by the amplification curve (AC) during the process of diagnostics of Covid-19 (DC). A relevant observation is that the worst-case adaptive schemes allow for using *nonuniform amounts of genetic material from different individuals,* which may be interpreted as using *nonbinary test matrices.*

4.4.1 Practical Adaptive AC-DC Schemes

In [69] a simple probabilistic two-stage AC-DC scheme that significantly improves upon the original single-pooling scheme of Dorfman and builds upon the SQGT framework is described. The underlying idea is to follow the same overall strategy as in the single-pooling scheme, but exploit the SQ information obtained in the first stage to perform better-informed testing in the second stage (i.e., dispense with individual testing of all individuals that feature in infected pools as part of the second stage).

Consider a scenario where we have access to semiquantitative tests that return one of three values: If no individual featured in the test is positive, the test returns 0. If between 1 and τ individuals are positive, for some threshold $\tau \geqslant 1$, the test returns 1. Finally, if more than τ individuals test positive, the test returns 2. This scheme can be interpreted as follows: Suppose that C_t is the observed cycle thresholds (defined in Section 4.2.3 for a particular test). If $C_t > c_1$ for some large threshold c_1, we say that the outcome is 0 as the potential viral or viral-like contamination load is too small to claim the presence of an infected individual. If $c_2 \leqslant C_t \leqslant c_1$, we say that the output

is 1 and based on the average viral load, convert this into the maximum possible number of infected individuals τ. If $C_t < c_2$, we say that the output is 2 and that more than τ individuals in the pool are affected.

For the new single-pooling AC-DC scheme, we assume that the population contains n individuals, each of which is independently positive with some probability p (which can be estimated based on regional infection rate reports; for example, at UIUC in September/October 2020 [89], $p \simeq 0.05$), and proceed as follows:

1. **Stage 1:** Divide the n individuals into n/s disjoint pools $\mathcal{S}_1, \ldots, \mathcal{S}_{n/s}$, each of size s.

2. **Stage 2:**

   If a pool $\mathcal{S}_i$ tests 0, then immediately set the status of all individuals $\in \mathcal{S}_i$ as "negative".

   If a pool $\mathcal{S}_i$ tests 1, then apply a nearly-optimal zero-error nonadaptive group testing scheme to detect the t infected individuals in $\mathcal{S}_i$.

   If a pool $\mathcal{S}_i$ tests 2, then test all individuals $\in \mathcal{S}_i$ separately.

The zero-error nonadaptive GT schemes we use in the second stage can be designed with $m(s, \tau) = c \cdot \tau^2 \log(s/\tau)$ tests. Thus, the expected number of tests per individual of the testing scheme, T/n, as a function of the probability of infection p, the first-stage pool size s, and the threshold τ can be computed as:

$$\mathbb{E}[T/n] = \frac{1}{s} + p_1 \cdot \frac{c \cdot \tau^2 \log(s/\tau)}{s} + p_2, \tag{4.1}$$

where $p_1 = \Pr[1 \leqslant B(s,p) \leqslant \tau]$ and $p_2 = \Pr[B(s,p) > \tau + 1]$ denote the probability that a given pool tests 1 and 2, respectively. Here, $B(s,p)$ stands for a binomial random variable with s trials and success probability p. Additional details and analysis are presented in [69]. Furthermore, [69] notes a simple scheme to improve upon the results presented herein: By employing double pooling (or multiple pooling) strategy, the expected number of tests can be reduced.

4.4.2 Probabilistic SQGT with Variable Viral Load

It is also simple to analyze how the SQGT scheme from the previous section performs when infected individuals may have either low or high viral loads, i.e., it is straightforward to account for heavy hitters. To this end, we consider a simplified model where each individual is independently infected and presents a low viral load at the time of testing with probability p_{i1}, or is infected and presents a high viral load at the time of testing with probability p_{i2}. In particular, each individual is infected (regardless of her/his viral load) with total infection probability $p = p_{i1} + p_{i2} < 1$.

As already explained, individuals with high viral load are problematic because, based on the SQ output of RT-PCR, pools featuring *one* such individual may be mistaken for pools with *several* infected individuals with low-to-average viral load.[8] This phenomenon naturally leads us to consider the following modified version of the testing method studied in Section 4.4.1: A test applied to a pool of individuals has outcome 0 if there are no infected individuals in the pool, outcome 1 if there exists *exactly* one infected individual with *low* viral load, and 2 if either there exists more than one infected individual with low viral load, or at least one infected individual with *high* viral load. Therefore, as expected, individuals with high viral load obfuscate the test outcomes. The expected number of tests per individual as a function of p_{i1} and p_{i2} is given by

$$\frac{1}{s} + s \cdot p_{i1} \cdot (1-p)^{s-1} \cdot \lceil \log s \rceil + 1 - s \cdot p_{i1} \cdot (1-p)^{s-1} - (1-p)^s, \qquad (4.2)$$

where $p = p_{i1} + p_{i2}$. As in the previous case, double pooling and multiple pooling strategies can be incorporated to obtain improved results.

4.5 AC-DC Schemes: Worst-Case Model Analysis

We introduce an adaptive SQGT scheme that is worst-case and nearly-optimal from the information-theoretic perspective. In the worst case, the schemes extend the ideas behind Hwang's generalized splitting [94] in two directions that lead to algorithms using what we call *parallel* and *deep search*,

[8] This is not problematic for *binary* group testing, where the test outcomes do not distinguish between one or several infected individuals in the pool.

respectively. In both settings, the outcomes of the first round of testing inform the choice of the composition of the test in the rounds to follow. The methods are collectively referred to as the AC-DC schemes, in reference to the use of the information provided by the amplification curve (AC) during the process of diagnostics of Covid-19 (DC). A relevant observation is that the worst case adaptive schemes allow for using *nonuniform amounts of genetic material from different individuals,* which may be interpreted as using *nonbinary test matrices.* We assume that we are given a set of n samples with at most d infected individuals. Our goal is to minimize the number of tests needed to identify *all infected individuals* and we do not impose any restrictions on the "simplicity" of our scheme. As a result, we consider a generalization of the model described in the previous section which allows for more than three test outcomes.

For simplicity, as well as for practical reasons,[9] we focus on equidistant thresholds but allow for warm-up/saturation effects. We refer to this model as the saturation GT scheme.

Let $\tau, m \in \mathbb{Z}^+$ represent the distance between the thresholds and the number of thresholds, respectively.

Denote the outcomes of the test by a nonnegative integer $t \leqslant m$. Then,

$$t = \begin{cases} 0, & \text{if every sample in the test is negative,} \\ 1, & \text{if the number of infected individuals is between 1 and } \tau, \\ 2, & \text{if the number of infected individuals is between} \\ & \tau + 1 \text{ and } 2\tau, \\ \vdots & \vdots \\ m-1, & \text{if the number of infected individuals is between} \\ & (m-2)\tau + 1 \text{ and } (m-1)\tau, \text{ or} \\ m, & \text{if the number of infected individuals is at least} \\ & (m-1)\tau + 1. \end{cases} \tag{4.3}$$

We seek to identify d infected individuals from a population of size n given that each test returns a value in (4.3). We refer to this problem as the (n, d)

[9] As we quantize the C_t values or the phase transition thresholds according to equally spaced cycle numbers

adaptive SQGT problem or the (n, d)-ASQGT problem for short.

Another way of looking at (4.3) is that if the collection of samples tested contains d' infected individuals, then the output of the test is $\lceil \frac{d'}{\tau} \rceil$ when $d' \leqslant m\tau$ and m otherwise. Note that for every test there are $m+1$ possible outcomes and the output of a test tells us roughly (within at most τ) how many total infected samples are part of the tested pool of samples.

Remark 12. *As discussed in Example 13, the ideas discussed here are applicable to the case where the widths of the thresholds are nonuniform.*

Let $2^\beta = m+1$. Motivated by practical applications, we will be interested in the case where $\beta = \mathcal{O}(1)$. Our main results are two algorithms, which we refer to as ***parallel search*** and ***deep search***. Parallel search is applicable for the setting $d > \beta$. In Lemmas 12 and 13, we show that using parallel search it is possible to efficiently identify from a set of pools (each of size $s = 2^\alpha$ and large enough to contain at least β infected individuals) a set of β defectives using at most α tests. Note that as a first-step simplification, one may think of n being approximately equal to $d \cdot 2^\alpha$; the notation involving α is chosen to enable a comparison between our SQGT search scheme and the well-known splitting approach by Hwang [112]. Deep search, discussed in Lemma 14 and applicable for the setting $d < \beta$, shows that it is possible to identify all d infected individuals using roughly $\frac{d \cdot \alpha}{\beta - \log(\beta)}$ tests. Our main result is Algorithm 1, which for $d = \Omega(n)$ shows that one can identify d infected individuals using at most $\frac{d}{\beta} \cdot (\alpha + 3 + \log \beta)$ tests. These results show that adaptive SQGT roughly provides β-fold savings in the number of tests when compared to classical adaptive GT. Furthermore, they differ from the information-theoretic lower bound (as it applies to ASQGT) of Lemma 11 by $\mathcal{O}(\frac{d}{\beta})$ tests. It remains an open problem to identify whether it is possible to solve the (n, d)-ASQGT problem using fewer tests.

We start with the following obvious claim, which allows us to restrict our attention to the case where $\tau = 1$ and simplifies the problem at hand.

Claim 19. *Let $\mathcal{G}$ be the set of test subjects and suppose that there are at most d infected individuals within this group. Let $\mathcal{P}^{(1)}$ be a pool formed by taking one sample from each individual in $\mathcal{G}$ and let $\mathcal{P}^{(w)}$ be a pool formed by taking w samples from each individual in $\mathcal{G}$. Let $t^{(1)}$ be the output of testing $\mathcal{P}^{(1)}$ under the setup $(m, \tau) = (m, 1)$ and let $t^{(w)}$ be the output of testing $\mathcal{P}^{(w)}$ under the setup $(m, \tau) = (m, w)$, according to (4.3). Then, $t^{(1)} = t^{(w)}$.*

Next, we present a lower bound (i.e., information-theoretic or counting lower bound) on the number of tests necessary to solve the (n, d)-ASQGT problem. The result follows from a simple counting argument and is consistent with the result from Claim 19, as it does not depend on the width τ of the threshold.

Lemma 11. *Let $n = (2^\alpha + 1) \cdot d + 2^\alpha \cdot \delta + \Delta$, where α, δ, Δ are integers, $\delta < d$, and $\Delta < 2^\alpha$. Then, the number of tests $L(n, d, m)$ needed to identify the infected individuals is bounded as:*

$$L(n, d, m) \geqslant \frac{d}{\beta} \cdot (\alpha + 1).$$

Proof. The number of ways to select at most d infectives in a group of n individuals is $\sum_{i=0}^{d} \binom{n}{i}$. Thus, we have

$$\begin{aligned}
L(n, d, m) \geqslant \log_{m+1} \left(\sum_{i=0}^{d} \binom{n}{i} \right) &\geqslant \log_{m+1} \binom{n - d + d}{d} \\
&\geqslant \log_{m+1} \left(\frac{n - d + d}{d} \right)^d \\
&\geqslant d \cdot \log_{m+1} \left(2^\alpha \big(1 + \frac{\delta}{d} + \frac{\Delta}{2^\alpha d} + \frac{1}{2^\alpha} \big) \right) \\
&\geqslant \frac{d \cdot \alpha}{\beta} + \frac{d}{\beta} \cdot \log_2 \left(1 + \frac{2^\alpha \cdot \delta + \Delta + d}{2^\alpha d} \right) \\
&\geqslant \frac{d}{\beta} \cdot (\alpha + 1) \,.
\end{aligned}$$

□

The next example illustrates a simple approach for addressing the ASQGT problem, and motivates the analysis that follows.

From here on, we write $[[x]] = \{0, 1, \ldots, x - 1\}$ and $[x] = \{1, \ldots, x\}$.

Example 11. *Suppose that we are given a collection of n individuals with exactly d infected subjects. We start by randomly partitioning the set of n individuals into d groups each of size $s = \frac{n}{d} = 2^\alpha$, where we assumed for simplicity that $d|n$. The expected number of infected individuals in each group is* 1.

Denote the d groups or pools by $\mathcal{G}_0, \mathcal{G}_1, \ldots, \mathcal{G}_{d-1}$; all groups have the same size and from this point on, for simplicity, assume that each group contains

exactly one *infected subject. For* $i \in [[d]]$ *we proceed as follows. We partition* $\mathcal{G}_i$ *into* 2^β *groups of equal size and denote the subgroups as* $\mathcal{G}_i^{(0)}, \mathcal{G}_i^{(1)}, \dots, \mathcal{G}_i^{(2^\beta - 1)}$. *Under this setup, there exists exactly one index* $j^\star$ *such that the number of infected individuals in* $\mathcal{G}_i^{(j^\star)}$ *equals to one, and every other group* $\mathcal{G}_i^{(j)}$, $j \in [[2^\beta]] \setminus j^\star$ *is free of infected individuals.*

Next, we form a new set of pools, which we denote by $\mathcal{P}_i$, $i \in [[d]]$, *comprising* k *replicas of the samples in* $\mathcal{G}_i^{(k)}$, *for all* $k = 0, \dots, 2^\beta - 1$. *Let* t_i *denote the output of the semi-quantitative test described in (4.3) after the pool* $\mathcal{P}_i$ *is tested. Then, it is straightforward to observe that the outcome* t_i *is* $j^\star$, *and hence we can identify the group which contains the one single infected individual using only one* nonbinary outcome *test. We repeat this procedure for each group* $\mathcal{G}_i$, $i \in [[d]]$, *partitioned into subgroups. It can be hence seen that it is possible to identify* d *infected individuals using only* $d\frac{\alpha}{\beta}$ *tests assuming each of the* d *groups of size* 2^α *each contain* exactly *one infected subject.* □

To make this argument rigorous, we need to account for the fact that not every group will have exactly one infected individual. In this case, upon creating the subpools we have to recursively test them until we identify a prescribed number of infected individuals. In fact, the approach from the previous example is a special case of what we refer to as *deep search*, described in Lemma 14. The resulting algorithm is summarized in Algorithm 1, and it requires roughly an additional factor of $\mathcal{O}(\frac{d}{\beta})$ tests compared to the information-theoretic lower bound.

4.5.1 Parallel Search

We start by introducing some useful notation. Suppose that $\mathcal{G}'$ is a subgroup of individuals to be tested and that the outcome of a test governed by (4.3) is t. In this case, we say that $\mathcal{G}'$ is a t-infected group. When referring to an ordered collection of groups $(\mathcal{G}_0, \mathcal{G}_1, \dots, \mathcal{G}_{g-1})$, we say that the collection is a $(t_0, t_1, \dots, t_{g-1})$-infected group if $t_0 \geqslant t_1 \geqslant \cdots \geqslant t_{g-1}$ and $\mathcal{G}_i$ is a t_i-infected group, for $i \in [[g]]$. We also say that $(\mathcal{G}_0, \dots, \mathcal{G}_{g-1})$ is a β-***minimal group*** if $\sum_{j=0}^{g-2} t_j < \beta$, but $\sum_{j=0}^{g-1} t_j \geqslant \beta$.

The following lemma constitutes the key component of one of our approaches to solving the (n, d)-ASQGT problem. We refer to the procedure described in the proofs of the next two results as parallel search.

In the first lemma below, we make the simplifying assumption that a group is β-minimal and $g = \beta$. Afterward, in Lemma 13 we consider the case when $g < \beta$.

Lemma 12. *Let α and β be positive integers. Suppose that $(\mathcal{G}_0, \mathcal{G}_1, \ldots, \mathcal{G}_{\beta-1})$ is a β-minimal group, where $g = \beta$, and that each group has size at most 2^α. Then, we can identify β infected individuals in the group using at most α tests.*

Proof. We prove the result by induction on α, where 2^α as before is the size of each subgroup. Recall that under this setup $t_0 = t_1 = \cdots = t_{g-1} = 1$ and $g = \beta$. Refer to Figure 4.11.

First, consider the case $\alpha = 1$, for which we have β 1-infected groups of individuals and each group has size 2. For shorthand, denote the β infected groups as $\mathcal{G}_0, \mathcal{G}_1, \ldots, \mathcal{G}_{\beta-1}$. From these β groups, we form a "super-pool" of samples which contains a total of $2^0 + 2^1 + 2^2 + \cdots + 2^{\beta-1} = 2^\beta - 1$ samples. More precisely, for $i \in [[\beta]]$, the super-pool contains 2^i samples from one individual $\in \mathcal{G}_i$. Since $t_0 = t_1 = \cdots = t_{\beta-1} = 1$ and $\tau = 1$, according to (4.3) the output returned after testing this super-pool of samples is a number t between 0 and $2^\beta - 1$. Let $b_0, b_1, \ldots, b_{\beta-1}$ be the binary representation of the number t. It is straightforward to verify that $b_i = 1$ then the individual selected from $\mathcal{G}_i$ is infected. Otherwise, if $b_i = 0$, then the above described individual is not infected, which implies the other individual (the one not tested) in group $\mathcal{G}_i$ is infected. Thus, we conclude the statement in the lemma holds when $\alpha = 1$.

For the inductive step, assume that the statement holds when the group size is at most $2^{\alpha'}$ and consider the setup where the group size is $2^\alpha = 2^{\alpha'+1}$. We follow the same approach as described for $\alpha = 1$ for creating super-pools. Under this setup, we have β 1-infected groups $\mathcal{G}_0, \mathcal{G}_1, \ldots, \mathcal{G}_{\beta-1}$, each of size $2^{\alpha'+1}$. For $i \in [[\beta]]$, let $\mathcal{Q}_i \subseteq \mathcal{G}_i$ be a subset of $\mathcal{G}_i$ of size $2^{\alpha'}$. Next we construct a super-pool that contains 2^i samples from each individual in $\mathcal{Q}_i$, $i \in [[\beta - 1]]$. Let t denote the output of testing this super-pool according to (4.3), where $b_0, b_1, \ldots, b_{\beta-1}$ is the binary representation of t. As before, if $b_i = 1$, then $\mathcal{Q}_i$ has a single infected individual. Otherwise, if $b_i = 0$, then there is an infected individual in the set $\mathcal{G}_i \setminus \mathcal{Q}_i$ which also has size $2^{\alpha'}$. For $i \in [[\beta]]$, let $\mathcal{G}'_i = \mathcal{Q}_i$ if $b_i = 1$ and otherwise, if $b_i = 0$, set $\mathcal{G}'_i = \mathcal{G}_i \setminus \mathcal{Q}_i$. Then, $(\mathcal{G}'_0, \mathcal{G}'_1, \ldots, \mathcal{G}'_{\beta-1})$ is a $(1, 1, \ldots, 1)$-infected group and we can apply

the inductive hypothesis to $(\mathcal{G}'_0, \mathcal{G}'_1, \ldots, \mathcal{G}'_{\beta-1})$. $\square$

For the case $g < \beta$, we use a similar partitioning idea to identify at most β subgroups which satisfy the conditions in the lemma. The difference between the approaches is that for $g < \beta$ the number of samples added into the pool is dictated by a mixed-radix representation (in which the numerical base varies from position to position) rather than a binary representation. For simplicity, we assume from now on that β is an even integer although the results hold for odd integers as well.

Lemma 13. *Let α, β, g be positive integers such that $g < \beta$. Suppose that $(\mathcal{G}_0, \mathcal{G}_1, \ldots, \mathcal{G}_{g-1})$ is a β-minimal group and that each group has size at most 2^α. Then, we can identify β infected individuals using at most α tests.*

Proof. We begin with the following claim which we find useful in our subsequent discussion.

Claim 20. *Suppose we are given a sequence $(t_0, \ldots, t_{g-1}) \in [[\beta + 1]]^g$, where $g < \beta$, and the values $t_0 \geqslant t_1 \geqslant \cdots \geqslant t_{g-1}$ are such that $\sum_{j=0}^{g-1} t_j \geqslant \beta$, but $\sum_{j=0}^{g-2} t_j < \beta$. Furthermore, let $(n_0, \ldots, n_{g-1}) \in [[t_0 + 1]] \times [[t_1 + 1]] \times \cdots \times [[t_{g-1} + 1]]$. Then, the number of different choices for $(n_0, \ldots, n_{g-1})$ is at most 2^β.*

Recall the main idea behind the proof of Lemma 12, where we tacitly assumed that $g = \beta$. There, we used the binary representation of the integer t, where t denotes the test outcome of the super-pool, to determine which of the tested subgroups involved infected individuals. In order to make this argument work, we formed the super-pool by adding 2^i samples from group $\mathcal{Q}_i \subseteq \mathcal{G}_i$ for $i \in [[\beta]]$, where $|\mathcal{Q}_i| = \frac{|\mathcal{G}_i|}{2}$. Next, the idea is to add N_i samples from each group, where N_i is chosen by considering a mixed-radix representation of the number t.

We say that $(b_0, b_1, \ldots, b_{g-1})$ is the $(t_0, t_1, \ldots, t_{g-1})$-mixed radix representation for t if the following is true. Let $N_0 = 1$. For $i \in [g-1]$, let $N_i = (t_{i-1} + 1) \cdot N_{i-1}$. Note that when $t_0 = t_1 = t_2 = \cdots = t_{g-1} = 1$, $N_i = 2^i$. The mixed radix representation of t is of the form $t = \sum_{i=0}^{g-1} b_i \cdot N_i$, where $b_i \leqslant t_i$. Note that under this setup since $b_i \leqslant t_i$, the sequence $(b_0, b_1, \ldots, b_{g-1}) \in [[t_0+1]] \times [[t_1+1]] \times \cdots \times [[t_{g-1}+1]]$ provides a unique representation and is invertible provided that $(t_0, t_1, \ldots, t_{g-1})$ is given. In other words, given the number t we can uniquely determine the i-th digit in the

$(t_0, t_1, \ldots, t_{g-1})$ mixed radix representation for t, which is b_i. Furthermore, as a result of Claim 20, we know that $t \leqslant 2^\beta - 1 = m$.

We are now ready to proceed with the proof. Suppose that $(\mathcal{G}_0, \mathcal{G}_1, \ldots, \mathcal{G}_{g-1})$ is a β-minimal group. We will prove the result by induction and we will show the inductive step (since the base case follows from similar ideas).

For the inductive step, assume the statement holds when the group size is at most $2^{\alpha'}$ and consider the setup where the group size is $2^\alpha = 2^{\alpha'+1}$. Note that we have $(t_0, t_1, \ldots, t_{g-1})$-infected groups $\mathcal{G}_0, \mathcal{G}_1, \ldots, \mathcal{G}_{g-1}$ each of size $2^{\alpha'+1}$. We form our super-pool as follows. As before, for each $i \in [[g]]$, we select a subset $\mathcal{Q}_i \subset \mathcal{G}_i$ of size $|\mathcal{Q}_i| = 2^{\alpha'}$. For each individual in $\mathcal{Q}_i$ we add N_i samples into the superpool, where N_i is as defined in the previous paragraphs.

Let t be the output of testing the resulting super-pool according to (4.3) and let b_i denote the i-th symbol of the $(t_0, t_1, \ldots, t_{g-1})$-mixed radix representation of t.

Note that based on t, we can determine the number of infected individuals in each of the subgroups $\mathcal{Q}_0$, $\mathcal{G}_0 \setminus \mathcal{Q}_0$, $\ldots$, $\mathcal{Q}_{g-1}$, $\mathcal{G}_{g-1} \setminus \mathcal{Q}_{g-1}$. In particular, since we know t_i, and given the output b_i which can be recovered after testing the super-pool, we know that for all $i \in [[g]]$, the number of infected subjects in $\mathcal{Q}_i$ is b_i and the number of infected subjects in $\mathcal{G}_i \setminus \mathcal{Q}_i$ is $t_i - b_i$. Given this information, we can generate $\mathcal{G}'_0, \mathcal{G}'_1, \ldots, \mathcal{G}'_{g'-1}$, where for $i \in [[g]]$, $\mathcal{G}'_i \subseteq \mathcal{Q}_i$ or $\mathcal{G}'_i \subseteq \mathcal{G}_i \setminus \mathcal{Q}_i$, such that the collection is a β-minimal group. Thus, we can apply the inductive hypothesis to $\mathbf{G}$. This establishes that we can identify β infected individuals using at most α tests and completes the proof. $\square$

4.5.2 Deep Search

Next, we consider the case $d < \beta$, and show that there exists an ASQGT scheme which requires roughly $\frac{d}{\beta - \log(\beta)} \cdot (\alpha + \log(\beta)) + d$ tests. Recall that the main idea behind the parallel search procedure was to simultaneously run a binary search on g subpools each of size 2^α. In this manner, using α tests we can identify β infected. For $d < \beta$, there are not sufficiently many infected individuals to use this method, and so for this setup, rather than perform a binary search in parallel, we test roughly $2^{\beta - \log(\beta)}$ (significantly smaller) subpools at the same time. We refer to this procedure as deep search.

Before proving a relevant lemma, we begin by describing a variant of well-known Newton identities. For completeness, we include a proof.

Claim 21. *Let $\mathcal{S} = \{j_1, \ldots, j_d\} \in \mathbb{Z}$ be a multiset of nonnegative integers each of which has value at most $p-1$, where p is an odd prime. Define $p_\ell(\mathcal{S}) = \sum_{k=1}^{d} j_k^\ell \bmod p$, the ℓ^{th} power sum of $\mathcal{S}$ over the finite field $\mathbb{F}_p$. Then, one can recover $\mathcal{S}$ given $\big(p_0(\mathcal{S}), p_1(\mathcal{S}), \ldots, p_d(\mathcal{S})\big)$.*

Proof. We represent $\mathcal{S}$ using $\mathcal{S}^{(+)}$, containing the positive elements in $\mathcal{S}$ and $z \in \mathbb{Z}$, which denotes the number of zeros in $\mathcal{S}$. Given $\mathcal{S}^{(+)}$ and z, the set $\mathcal{S}$ is uniquely determined.

First, note that Newton's identities can be used to recover the set $\mathcal{S}^{(+)} = \big\{i_1, i_2, \ldots, i_{d'}\big\}$. To see this, let $\sigma(i_1, i_2, \ldots, i_{d'}) = \prod_{k=1}^{d'}(1 - i_k x) = \sum_{k=0}^{d'} \sigma_k x^k \in \mathbb{F}_p[x]$ and assume that the operations are over the polynomial ring $\mathbb{F}_p[x]$, where the elements in $\mathcal{S}$ are assumed to lie in $\mathbb{F}_p$. Then, we have

$$\begin{aligned}
\sum_{\ell=1}^{d} p_\ell(\mathcal{S}) \cdot x^\ell \,(\mathrm{mod} x^{d+1}) &= \sum_{\ell=1}^{d} p_\ell(\mathcal{S}^{(+)}) \cdot x^\ell \,(\mathrm{mod} x^{d+1}) \\
&= \sum_{\ell=1}^{d} \sum_{k=1}^{d'} i_k^\ell \cdot x^\ell \,(\mathrm{mod} x^{d+1}) \\
&= \sum_{k=1}^{d'} \sum_{\ell=1}^{d} i_k^\ell \cdot x^\ell \,(\mathrm{mod} x^{d+1}) \\
&= \sum_{k=1}^{d'} \left(\frac{1 - i_k^{d+1} \cdot x^{d+1}}{1 - i_k \cdot x} - 1 \right) (\mathrm{mod} x^{d+1}) \\
&= \sum_{k=1}^{d'} \frac{i_k \cdot x}{1 - i_k \cdot x} \,(\mathrm{mod} x^{d+1}),
\end{aligned}$$

which implies

$$\sum_{\ell=1}^{d} p_\ell \cdot (\mathcal{S}^{(+)}) \cdot x^\ell \cdot \sigma(i_1, \ldots, i_{d'}) \,(\mathrm{mod} x^{d+1}) = -x \cdot \sigma'(i_1, \ldots, i_{d'}).$$

The above equality in turn implies $\sum_{k=0}^{\ell-1} \sigma_k \cdot p_{\ell-k}(\mathcal{S}) = -\ell \cdot \sigma_\ell$. Thus, given $p_\ell(\mathcal{S})$, $\ell \in [d]$, one can recover $\sigma(\mathcal{S}^{(+)})$ as well as the multiset $\mathcal{S}^{(+)}$. The multiset $\mathcal{S}$ can be subsequently recovered by noting that the number of zeros in $\mathcal{S}$ equals $p_0(\mathcal{S}) - |\mathcal{S}^{(+)}|$. □

Lemma 14. *Let p be an odd prime such that $p \geqslant 2^L - 1$ and $(p-1) \cdot d < 2^\beta$. Suppose that $\mathcal{G}$ is a d-infected set of size 2^α, and $d \leqslant p-1$. Then we can identify the d infected individuals using at most $d \cdot \frac{\alpha}{L}$ tests.*

Proof. For simplicity we assume that $L|\alpha$, and, similar to Lemmas 12 and 13, use induction in α. For the case $\alpha = L$, we run d tests, and for each test we design a different test group. For $\ell \in [d]$, test group ℓ contains $j^\ell \in \mathbb{F}_p$ samples from each individual indexed by $j \in [[2^L]]$. Suppose that $\mathcal{D}$ is a multi-set of elements from $[[2^L]]$ and that $\mathcal{D}$ is such that if group j has k infected individuals, then the elements from group j appear k times in $\mathcal{D}$. Then according to the above setup the output of performing the SQGT on pool ℓ results in the following ℓ-th power sum:

$$p_\ell(j_1, j_2, \ldots, j_d) = \sum_{k=1}^{d} j_k^\ell.$$

Note that $j_k^\ell < p$ (since by design $j^\ell \in \mathbb{F}_p$) and so $p_i(j_1, j_2, \ldots, j_d) \leqslant (p-1)d < 2^\beta$. Thus, for $\ell \in [[d+1]]$, we can recover $p_\ell(j_1, j_2, \ldots, j_d) = p_\ell(j_1, j_2, \ldots, j_d) \bmod p$, since $p_0(j_1, j_2, \ldots, j_d) \bmod p = d$ follows from the fact that $\mathcal{G}$ is a d-infected set. From the set of $d+1$ power sums over the field $\mathbb{F}_p$, we can recover the multi-set $\{j_1, \ldots, j_d\}$ from Claim 21, which completes the proof of the base case.

For the inductive step, assume the statement holds for group sizes at most $2^{\alpha'}$ and consider a group size $2^\alpha = 2^{\alpha'+L}$. As in the proofs of Lemmas 12 and 13, we work with subgroups. The subgroups are formed by partitioning the set of $2^{\alpha'+L}$ individuals into 2^L subgroups $\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_{2^L}$ each of size $2^{\alpha'}$. Applying the same ideas as before, we form d test groups where test group $\ell \in [d]$ contains $j^\ell \in \mathbb{F}_p$ samples from each individual in subgroup $j \in [[2^L]]$.

Let $\mathcal{D} = \{j'_1, j'_2, \ldots, j'_d\}$ be a multiset of integers such that j'_u appears t times in the multiset if and only if group j'_u has t infected individuals. Using the same approach as for the base case, we first recover the power sums $p_i(j'_1, j'_2, \ldots, j'_d)$. Then from Claim 21, we recover the set $\mathcal{D}$ in the same manner as before and we apply the inductive hypothesis to the subgroups in $\mathcal{D}$. This completes the inductive step and the proof. □

Remark 13. *For the case $d = 1$, the deep search procedure coincides with the approach described in Example 11. Deep search may be of limited practical*

value due to the large amounts of sample material required for testing, but is of theoretical relevance due to the fact that it generalizes Hwang's generalized splitting method to the SQGT setting for a small number of infected individuals.

4.5.3 (n, d)-ASQGT Schemes

As discussed in the text following Example 11, our general approach to adaptive SQGT is to first partition the set of n individuals into $\frac{d}{\beta}$ subpools and test each subpool separately using either parallel search or deep search, depending on the number of infected in each subpool. Parallel search produces the best results in the worst case, provided that the number of infected individuals across all the subpools is $\leqslant \beta$, while parallel search gives the best results for the case of a large number of infected individuals.

Let $T_P(n, d)$ denote the number of tests required by our ASQGT scheme, summarized in Algorithm 1, and let $n - d = 2^\alpha \cdot d + 2^\alpha \cdot \delta + \Delta$, where α, δ, Δ are integers such that $\delta < d$ and $\Delta < 2^\alpha$. In order to simplify the notation by avoiding floor and ceiling functions, we assume that $\beta | d$ and $\beta | \delta$.

Theorem 14. $T_P(n, d) \leqslant \frac{d}{\beta} \cdot (\alpha + 3 + \log \beta) + \frac{\delta}{\beta}$.

Proof. Since the first step involves testing $\frac{d}{\beta} + \frac{\delta}{\beta}$ groups, the first step requires $\frac{d}{\beta} + \frac{\delta}{\beta}$ tests. For the next steps, note that each group has size $\leqslant 2^{\alpha+1}\beta$. Hence, we can uncover β infected individuals using at most $\frac{\alpha+1+\log(\beta)}{\beta}$ tests according to Lemmas 12 and 13. In step 3, we use one additional test for every β infected individuals. Since there are d infected, the total number of tests required by Algorithm 1 equals

$$T_P(n, d) \leqslant \left(\frac{d}{\beta} + \frac{\delta}{\beta}\right) + \frac{d}{\beta} \cdot (\alpha + 1 + \log(\beta)) + \frac{d}{\beta} = \frac{d}{\beta} \cdot (\alpha + 3 + \log(\beta)) + \frac{\delta}{\beta}.$$

□

As discussed earlier, the parallel search ASQGT scheme requires $\mathcal{O}(\frac{d}{\beta})$ more tests than the information-theoretic lower bound. When $\beta = 1$, our scheme requires $\mathcal{O}(d)$ additional tests which agrees with the traditional adaptive binary setting studied in [94].

Next, we consider the second approach to the ASQGT problem based on deep search, for the case where $d < \beta$. Let $T_D(n, d)$ denote the number

Algorithm 1 Parallel search ASQGT scheme

1. ***Initialize***: Partition the set of n individuals into $\frac{d+\delta}{\beta}$ groups, denoted by $\mathcal{G}_0, \mathcal{G}_1, \ldots, \mathcal{G}_{\frac{d+\delta}{\beta}-1}$, each of size $\leqslant \beta \cdot 2^{\alpha+1}$.

   Test each subgroup individually. For $i \in [[\frac{d+\delta}{\beta}]]$, suppose that $\mathcal{G}_i$ is a t_i-infected group and let D denote the total number of infected subjects across all groups.

2. ***Parallel Search***: Identify a β-minimal group $(\mathcal{G}_{i_0}, \ldots, \mathcal{G}_{i_{g-1}})$, and apply parallel search on the group to uncover β infected individuals. Remove the β infected individuals from their respective groups.

3. ***Update***: Use one additional test to determine the number of infected subjects in $(\mathcal{G}_{i_0}, \ldots, \mathcal{G}_{i_{g-1}})$ after Step 2. Update $t_{i_0}, \ldots, t_{i_{g-1}}$ and D. If $D > 0$, go to Step 2.

of tests required by our algorithm and, with a slight abuse of parameter definitions, assume that $n - d = 2^\alpha \cdot d$. Furthermore, assume as before that $d|2^\beta$ and $d|n$. The corresponding approach is described in Algorithm 2.

Algorithm 2 Deep search ASQGT scheme

1. ***Initialize***: Partition the set of n individuals into d groups, denoted by $\mathcal{G}_0, \mathcal{G}_1, \ldots, \mathcal{G}_{d-1}$, each of size 2^α. Test each subgroup individually and let D denote the total number of infected subjects across all groups.

2. ***Deep Search***: Identify a t_i-infected group $\mathcal{G}_i$, and apply deep search to uncover t_i infected subjects, for some $i \in [[d]]$.

3. ***Update***: Let $D = D - t_i$. If $D > 0$, go to Step 2.

Theorem 15. *The number of tests for deep search ASQGT satisfies*

$$T_D(n, d) \leqslant d \cdot \frac{\alpha}{\beta - \log \beta - 1} + \beta.$$

Proof. The first step in Algorithm 2 requires $d < \beta$ tests. According to Lemma 14, Step 2 requires at most $t_i \cdot \frac{\alpha}{\beta - \log(\beta) - 1}$ tests. Hence, the total

number of tests is upper bounded as

$$T_D(n,d) \leqslant \beta + \sum_{i=0}^{d-1} t_i \cdot \frac{\alpha}{\beta - \log\beta - 1}$$
$$= \beta + d \cdot \frac{\alpha}{\beta - \log\beta - 1}.$$

□

4.5.4 Error-resilient (n,d)-ASQGT Schemes

We consider next the question of designing ASQGT models that can tolerate a bounded number of birth-death (BD) chain errors. Recall from (4.3) that in the event that there are no errors, the output of testing a pool of individuals, of which d are infected, is an integer t, such that $t = d$ for $d \leqslant m$ and $t = m$, whenever $d > m$. Suppose instead that the erroneous output of testing a pool is t', where $t' \in \{t-1, t+1\}$ with the appropriate boundary conditions. We refer to such an error as a single BD error.

Our main result is described in Theorem 16. We prove that there exists a scheme that requires $\frac{d}{\beta-2} \cdot (\alpha + 3 + \log\beta) + \frac{\delta}{\beta}$ tests that can correct an arbitrary number of test errors. For the case where the number of test errors is a small integer e, $\left(\frac{d}{\beta} + e\right) \cdot (\alpha + 3 + \log\beta) + 2 \cdot \frac{d}{\beta} + \frac{\delta}{\beta} + e$ tests suffice, which implies that only $e\,(\alpha + 3 + \log\beta) + 2\frac{d}{\beta} + e$ additional tests are required to correct e errors in Algorithm 1.

The next claim highlights one of the main ideas behind our approach: Take multiple copies of samples from each of the individuals being tested in such a way to get error-free readouts even when errors occur. Here, as before, we assume that $2^\beta = m + 1$.

Claim 22. *Let $\mathcal{P}$ be a pool of individuals and suppose that $\mathcal{P}^{(\times 3)}$ is a pool which contains three samples from each individual in $\mathcal{P}$. Let t be the output of the test performed on the pool $\mathcal{P}^{(\times 3)}$ given that no errors occur, and suppose t' is a possibly erroneous output of the test performed on the pool $\mathcal{P}^{(\times 3)}$. Given t', one can determine t.*

Proof. Since we have taken 3 samples from each of the individuals in the pool $\mathcal{P}$, it follows that $t\,(\mathrm{mod}3) = 0$. Thus, if an error occurs, the output of the test under the BD model equals $t' \in \{t+1, t-1\}$ which implies that

$t' \,(\mathrm{mod} 3) = \pm 1$. If $t' \bmod 3 = 1$, then $t' = t + 1$ and so we can recover t by simply decrementing t' by one. Similarly, if $t' \bmod 3 = -1$, then $t' = t - 1$ and we can recover t by incrementing t' by one. □

Using the idea from the previous claim, we can determine exactly how many infected individuals are present in each of the tested pools despite the fact that testing errors can occur. We describe the underlying method through an example, for which we need the following terminology.

We say that $\mathcal{P}_i$ ***is a*** t_i***-infected group*** if the output of testing $\mathcal{P}_i^{(\times 3)}$ is in the set $\{3t_i - 1, 3t_i, 3t_i + 1\}$. We also say that $(\mathcal{P}_{i_0}, \ldots, \mathcal{P}_{i_{g-1}})$ ***is a*** β***-minimal group*** if $t_0 \geqslant t_{i_1} \geqslant \cdots \geqslant t_{i_{g-1}}$, $\sum_{j=0}^{g-2} t_j < \beta$, but $\sum_{j=0}^{g-1} t_j \geqslant \beta$.

Example 12. *For simplicity, assume that we have* $m = 3\,(2^\gamma - 1)$ *thresholds, and suppose that* $(\mathcal{P}_0, \mathcal{P}_1, \ldots, \mathcal{P}_{g-1)}$ *is a* γ*-minimal group, where we again make a simplifying assumption, namely* $\gamma = g$*. We proceed in the same manner as described in Lemma 6 and we first form a super-pool, denoted* $\overline{\mathcal{P}}$ *which consists of* 2^i *copies of each sample in* $\mathcal{P}_i$*. Afterward, we generate a larger pool of samples,* $\overline{\mathcal{P}^{(\times 3)}}$ *which contains* 3 *copies of each sample in* $\overline{\mathcal{P}}$*.*

Notice that given the output of the test $\overline{\mathcal{P}^{(3)}}$*, we can uniquely determine the number of infected that are in each of the groups* $\mathcal{P}_0, \mathcal{P}_1, \ldots, \mathcal{P}_{g-1}$*. Suppose that* t' *is the output of testing* $\overline{\mathcal{P}^{(3)}}$ *and suppose* t *is the output of testing* $\overline{\mathcal{P}^{(3)}}$ *assuming no errors occur during testing. From Claim 22, we can recover* t *and from Lemma 6 it is possible to determine how many infected are in* $\mathcal{P}_0$*, how many infected are in* $\mathcal{P}_1$*, etc.*

Another simple way to see how the above scheme overcomes BD noise is to see that it suffices that test outcomes differ from one another by at least three. This can be easily accomplished by fixing *the coefficients of* 2^1 *and* 2^0 *in the binary representation of the test outcomes to zero.*

More precisely, we can artificially introduce two subgroups, so that when $m = 2^\gamma - 1$*, we collect samples from subgroups labeled by* $1 < i < \gamma$*,* 2^i *with the amounts dictated by their labels. If the observed test outcome is*

$t' = \sum_{i=0}^{\gamma-1} \bar{e}_i \cdot 2^i$, *then the true test outcome is decoded as:*

$$e_{\gamma-1}e_{\gamma-2}\dots e_2e_1e_0 = \begin{cases} \bar{e}_{\gamma-1}\bar{e}_{\gamma-2}\dots \quad \bar{e}_2 00, \\ \qquad\qquad\qquad\quad \textit{if } \bar{e}_1\bar{e}_0 = 00 \textit{ or } \bar{e}_1\bar{e}_0 = 01, \\ \bar{e}_{\gamma-1}\bar{e}_{\gamma-2}\dots \quad \bar{e}_2\bar{e}_1\bar{e}_0 + 1 \textit{ (binary addition)}, \\ \qquad\qquad\qquad\quad \textit{if } \bar{e}_1\bar{e}_0 = 11. \end{cases} \tag{4.4}$$

□

The following claim is straightforward.

Claim 23. *Let $\alpha, \beta \geqslant 2, g$ be positive integers where $2^\beta = m + 1$, $g \leqslant \beta - 2$. Suppose that $(\mathcal{G}_0, \mathcal{G}_1, \dots, \mathcal{G}_{g-1})$ is a $(\beta-2)$-minimal group and that each group has size at most 2^α. Then we can identify $\beta - 2$ infected individuals using at most α tests.*

Proof. The proof follows immediately by applying the procedure described in Example 12 and noting that $3 \cdot 2^{\beta-2} < 2^\beta - 1$ when $\beta \geqslant 2$. □

Next, we turn our attention to a scheme designed for a small number of testing errors e. To this end, let $T_N(n, d, e)$ denote the number of tests required for a noisy ASQGT scheme that tolerates up to e BD testing errors (see Algorithm 3). As before, let $n - d = 2^\alpha d + 2^\alpha \delta + \Delta$, where α, δ and Δ are integers such that $\delta < d$ and $\Delta < 2^\alpha$. Once again we assume that $\beta | d$ and $\beta | \delta$.

We prove the correctness of our algorithm in the following theorem.

Theorem 16. *Let $\beta \geqslant 2$. We have $T_N(n, d, e)$ is at most*

$$\min\left(\left(\frac{d}{\beta} + e\right)(\alpha + 3 + \log\beta) + 2\frac{d}{\beta} + \frac{\delta}{\beta} + e,\ \frac{d}{\beta - 2}(\alpha + 3 + \log\beta) + \frac{\delta}{\beta}\right).$$

Proof. The second term under the minimum follows immediately from the parallel search ASQGT scheme given the use of a robust parallel search. Therefore, in the remainder of the proof, we focus our attention on the first term.

The first step of our algorithm requires $\frac{d+\delta}{\beta}$ tests and each time we execute Step 2, we perform $\alpha + 1 + \log\beta$ tests. Since Step 2 is executed at most $\frac{d}{\beta} + e$

Algorithm 3 Noisy search ASQGT scheme

1. ***Initialize***: Partition the set of samples from the n individuals into $\frac{d+\delta}{\beta}$ groups, denoted by $\mathcal{P}_0, \mathcal{P}_1, \ldots, \mathcal{P}_{\frac{d+\delta}{\beta}-1}$, and each of size at most $\beta 2^{\alpha+1}$.

   Test each subgroup $P_i^{(\times 3)}$ individually. For $i \in [[\frac{d+\delta}{\beta}]]$, suppose that $\mathcal{P}_i$ is a t_i-infected group and let D denote the total number of infected subjects in all subgroups.

2. ***Parallel Search***: Identify a β-minimal group $(\mathcal{P}_{i_0}, \ldots, \mathcal{P}_{i_{g-1}})$, and apply parallel search to uncover β potential infected subjects. Divide the set of β potentially infected individuals into two groups of sizes $\lfloor \frac{\beta}{2} \rfloor$ and $\lceil \frac{\beta}{2} \rceil$, denoted by $\mathcal{D}_1^{(\times 3)}, \mathcal{D}_2^{(\times 3)}$.

3. ***Verify***: Test $\mathcal{D}_1^{(\times 3)}, \mathcal{D}_2^{(\times 3)}$ to determine the total number of infected recovered. Update $t_{i_0}, \ldots, t_{i_{g-1}}$ and D.

4. ***Update Large Group Counts***: If only one group is present, $|\mathcal{P}_{i_0}| \geqslant \beta$, and $t_0 \geqslant 1$, then test $\mathcal{P}_{i_0}^{(\times 3)}$ to determine the number of infected in $\mathcal{P}_{i_0}$. Go back to Step 2.

times this implies that the total number of tests required by the first two steps of our procedure is at most

$$\frac{d+\delta}{\beta} + \left(\frac{d}{\beta} + e\right) \cdot (\alpha + 1 + \log \beta) .$$

For Step 3, note that since $\max\left\{|\mathcal{D}_1^{(\times 3)}|, |\mathcal{D}_2^{(\times 3)}|\right\} \leqslant \lceil \frac{\beta}{2} \rceil$ we have $t' \leqslant 3(2^{\lceil \frac{\beta}{2} \rceil} - 1) < 2^\beta - 1 = m$ when $\beta \geqslant 2$, and so we can determine exactly how many infected subjects are in each of the sets $\mathcal{D}_1^{(\times 3)}, \mathcal{D}_2^{(\times 3)}$ in Step 3. Each time Step 3 is executed, we require 2 tests. Since Step 2 is executed at most $\frac{d}{\beta} + e$ times, this step requires at most

$$\frac{2 \cdot d}{\beta} + 2 \cdot e$$

tests. Finally, since Step 4 is executed at most $\frac{d}{\beta} + e$ times, it follows that the total number of tests is at most $\frac{d+\delta}{\beta} + \left(\frac{d}{\beta} + e\right) \cdot (\alpha + 1 + \log \beta) + \frac{2 \cdot d}{\beta} + 2 \cdot e + \frac{d}{\beta} + e$, which proves the claimed result. □

We conclude the above exposition by observing that in a very recent com-

panion paper [113], we described adaptive schemes for SQGT that only use two rounds of testing and may hence have practical advantages over deep search methods. Nevertheless, the results in [113] rely on nonconstructive expander graph existence guarantees and trade other desirable testing properties for a reduced number of testing rounds.

4.5.5 Extensions to Nonuniform Threshold Widths

The next two examples illustrate how the ideas from the previous sections can be extended to the case where the threshold widths increase exponentially. For this case, we only consider small values of m (i.e., $m = 3, 4$).

Example 13. *In the following, we consider a model that mirrors the results from the previous section that discusses probabilistic priors in [69]. Suppose that the test outcomes equal*

$$t = \begin{cases} 0, & \text{if there are no infected subjects in the test,} \\ 1, & \text{if the number of infected samples is } 1, \\ 2, & \text{if the number of infected samples is in } [2,3], \\ 3 & \text{if the number of infected samples is in } [4,7]. \end{cases} \tag{4.5}$$

We consider the following extension of the approach discussed in Example 11. Suppose we have a pool of size 2^α that contains at least one infected subject. We start by testing this pool to determine the total number of infected individuals. There are two cases to consider: (a) The output of the test is 2 or 3, which indicates that there is more than a single infected in the pool or (b) The output of the test is 1.

Suppose that the outcome is (b). In this case, we run a variant of deep search. In particular, we divide the pool into 4 subpools and form a superpool from these 4 subpools which contains 0 samples from the first pool, 1 sample from the second pool, 2 samples from the third pool, and 4 samples from the fourth pool. It is straightforward to verify that in this case we can determine which of the 4 subpools contain the single infected sample by testing the superpool, and we then repeat this procedure using the subpool which contains the single infected.

If the outcome is (a), then we proceed to divide the pool of size 2^α into

two disjoint subpools, each of size $2^{\alpha-1}$. We further select one of the two subpools for testing. If the subpool tested contains a single infected, then we continue by applying the procedure discussed in the previous paragraph on the subpool of size $2^{\alpha-1}$ that contains one infected sample. Otherwise, we repeat the procedure from this paragraph on one of the subpools of size $2^{\alpha-1}$ that contains more than a single infected subject.

Using the procedure described above, it is straightforward to verify that recovering 2 infected individuals requires at most α tests provided we know the number of infected samples in the pool of size 2^{α}. Now suppose $n = d\,2^{\alpha}$. Then we can recover d infected subjects using at most $2 \cdot d + \frac{d \cdot \alpha}{2}$ tests as follows. First, we partition the set of n individuals into d groups each of size 2^{α} and we initially test each of these d groups. Afterward, we search for the infected individuals using the process outlined in this example. □

We note that despite the fact that we have focused on the case where m is a power of two, the next example shows that in some cases our ideas extend to settings where m is not necessarily a power of two. In the next example, we show an adaptive scheme that requires at most roughly $d + d \cdot \big(\log_3(\frac{n}{d}) + 1\big)$. The ideas are similar to the previous example except that here we only allow 3 thresholds.

Example 14. *For this example, we assume that $n = d \cdot 3^{\alpha}$. The output of the test is t, where:*

$$t = \begin{cases} 0, & \text{if no infected samples are present in the pool,} \\ 1, & \text{if the number of infected samples equals } 1, \\ 2, & \text{if the number of infected samples is } > 1. \end{cases} \tag{4.6}$$

The core idea behind the testing strategy is a simple extension of the previous example. Suppose we have a pool of size 3^{α} that contains at least one infected individual. First, we test this pool of size 3^{α} to determine the total number of infected individuals. There are two cases to consider: (a) The output of the test equals 2, which indicates that there is more than one infected sample in the pool or (b) The output of the test equals 1.

Suppose (b) occurred. We perform the same procedure as before except that instead of dividing the pool into 4 subpools, we divide the pool into 3 subpools of equal size. Next, we form a superpool from these 3 sub pools which contains

0 *samples from the first pool,* 1 *sample from the second pool, and* 2 *samples from the third pool. Similarly as before, we can determine which of the three subpools contains the single infected sample, and we then repeat this procedure using the subpool which contains the single infected sample.*

If (a) occurs, then we perform the same procedure as in the previous example. In particular, we divide the pool of size 3^α *into two disjoint subpools each of size at most* $\lceil \frac{3^\alpha}{2} \rceil$ *and perform a single test. If two infected individuals are contained in a single pool, then we repeat the procedure from this paragraph on the pool of size at most* $\lceil \frac{3^\alpha}{2} \rceil$ *that contains at least two infected samples. Otherwise, we perform the procedure from the previous paragraph on the subpool of size at most* $\lceil \frac{3^\alpha}{2} \rceil$ *that contains a single infected sample.*

Using this approach, it is straightforward to verify that recovering an infected requires at most $\alpha + 1$ *tests. Thus we can recover* d *infected individuals using at most* $d + d \cdot (\alpha + 1)$ *tests as follows. First, we partition the set of* n *infected into* d *groups each of size* 3^α*. Afterward, we search for the infected individuals using the process outlined in this example.* □

4.6 Open Problems

We provided an in-depth description of the quantitative RT-PCR protocol suitable for nonexperts, an overview of existing GT testing protocols for Covid-19 and their practical implications. These comparative studies motivated further explorations of quantized GT (or semiquantitative GT (SQGT)) protocols, especially under a new measurement-error model termed the birth-death chain noise model. We furthermore developed state-of-the-art adaptive SQGT schemes with combinatorial priors and provided extensive analytical results, including performance bounds, algorithmic solutions, and noisy testing protocols. Many open problems remain, including:

- *Probabilistic testing schemes for more than* 3 *thresholds*: We consider the setup where each test generated the output $0, 1,$ or 2 depending upon the number of defectives in each group. How much can one reduce the number of tests of our schemes if we incorporate additional semi-quantitative information, in the presence of errors?

- *Worst-case general SQGT testing schemes with a constant number*

of rounds: The schemes described in Section 4.5 have the potential drawback that almost every test depends upon the results of prior tests. It has been shown that in the binary group testing setting, the information-theoretic lower bound can be achieved using only two rounds of nonadaptive testing when the number of infected individuals is at most n^c for any constant $c < 1$ [114]. In a recent line of work, the authors of [113] showed how to implement two-round SQGT schemes for the saturation model studied in Section 4.5. It remains an open problem to generalize the approach for general quantized GT paradigms.

- *Practical SQGT schemes resilient to errors:* Practical two-stage SQGT schemes presented herein can be enhanced with noise-resilience properties in a straightforward manner by repeating each test a prescribed number of times, while keeping the number of testing stages the same. Nevertheless, it would be interesting to find more efficient, and still practical, ways of adding good noise-resilience properties to these schemes.

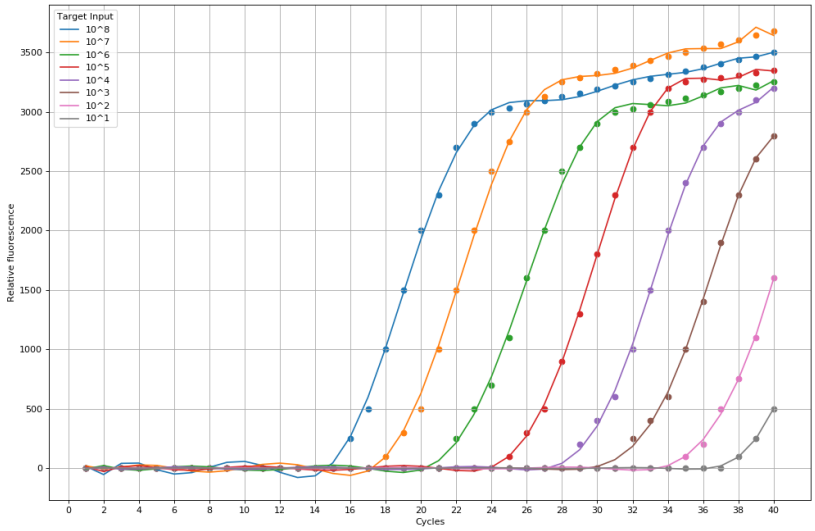


Figure 4.7: A typical amplification graph, plotting the relative fluorescence versus the number of PCR cycles for various input concentrations of the DNA sample. The dots represent actual fluorescent levels, while the curves represent a degree-10 polynomial approximation of the measurements. Since the solid curves are approximations, the fluorescence level for a small number of cycles can be negative, which is clearly not physically possible. Simple yet less precise piecewise linear and quadratic curves will be described when discussing error models for real-time PCR. Also, note that the fluorescence saturates after roughly $35 - 40$ cycles which shows that models that use the final cycle fluorescence cannot distinguish viral loads. Another observation is that due to the stochastic nature or RT-PCR it usually takes around $5 - 10$ cycles to obtain visible fluorescence, independent of the viral load. Both of these features demonstrate the highly nonlinear relationship between the viral load and the fluorescence.

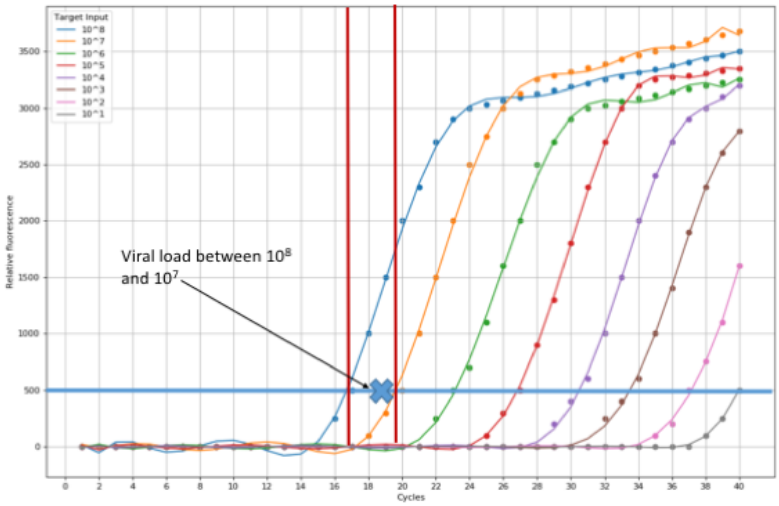


Figure 4.8: Amplification curves and quantization regions for the C_t values. Given a number of amplification curves used for calibration in a specific lab, the quantization regions in this example are chosen based on the intersection of the fluorescence detection level 500 and the calibration amplification curves. A C_t value for a particular experiment is placed in the quantization region bounded by the two "closest" amplification curves used for calibration and their underlying C_t values, or into the corresponding quantization bin. In this particular example, except for the quantization regions corresponding to the early and late cycles, the quantization regions are of nearly uniform length. Note that the larger the C_t value, the lower the viral load. Also, if one were to only use the fluorescence levels observed at the final RT-PCR cycle (i.e., cycle number $\sim$40), the results would be noninformative with respect to the viral load as a strong saturation effect comes into existence.

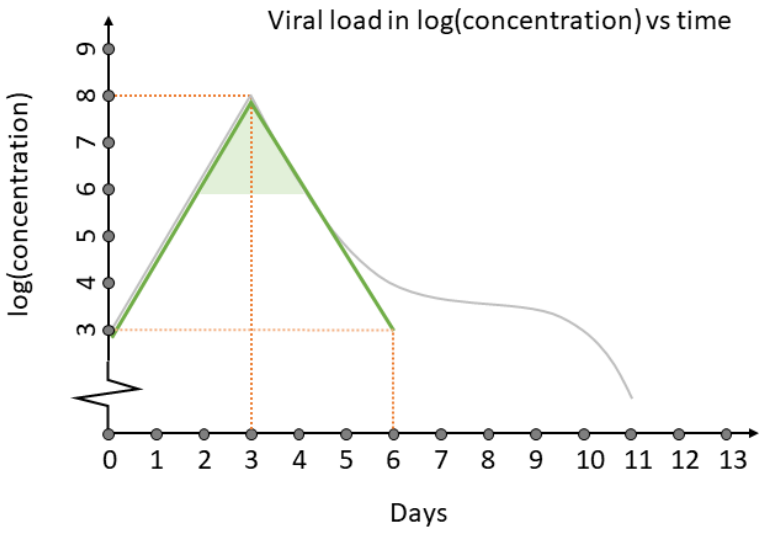


Figure 4.9: A typical viral load dynamics in an infected individual versus the time since infection. The viral load sharply spikes within the first three days of infection and then more gradually decreases. The nonlinear part of the viral load curve can be approximated by a linear component symmetric with respect to the linear component. This linear approximation will be used to determine the probability of heavy hitters, i.e., individuals who have an absolute viral load above 10^6.

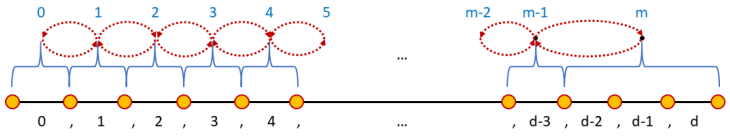


Figure 4.10: The birth-death noise model. Here, the assumption is that the C_t values can be corrupted by noise only insofar as they can be mislabeled as belonging to intervals adjacent to the correct interval (except for the values falling into the first and last quantization region or bin).

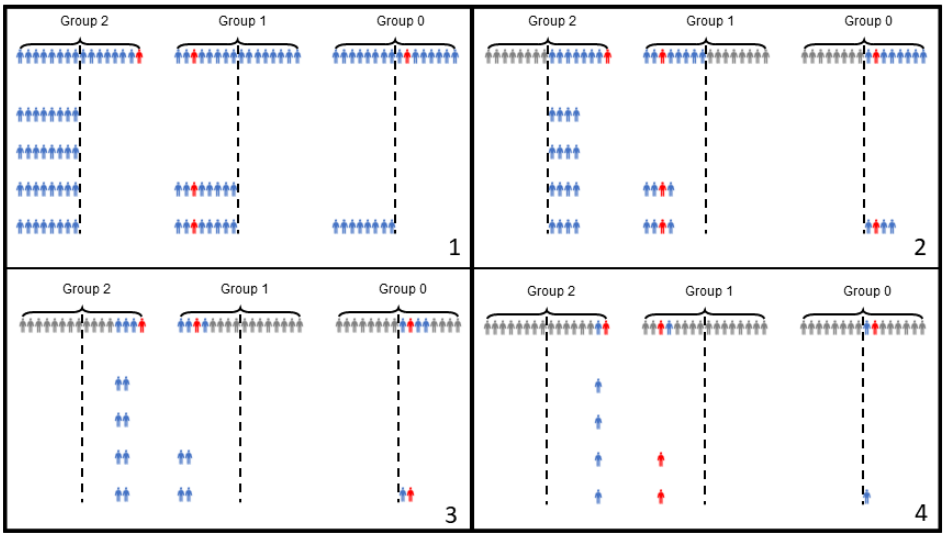


Figure 4.11: Intuitive illustration of the parallel search ASQGT procedure with some details removed for the ease of exposure. In this example, $m = 7$ and exactly one individual is infectious in each of the three groups. The weights of samples in each test round are set to $4, 2, 1$ as seen in Frame 1. A binary search procedure is implemented to find the infected individual in each group. In Frame 1, the test outcome for the first round is 2, implying that there is one infected individual in the second group. Thus, the subgroups from groups 1 and 3 that were probed in Frame 1 are discarded as illustrated in Frame 2. Similarly, the second subgroup of group 2 that was not tested is discarded as well. The subgroups that contain an infected individual are further probed as seen in Frames $2, 3$ and 4.

REFERENCES

[1] V. Zhirnov, R. M. Zadegan, G. S. Sandhu, G. M. Church, and W. L. Hughes, "Nucleic acid memory," *Nature Materials*, vol. 15, no. 4, p. 366, 2016.

[2] A. Al Ouahabi, J.-A. Amalian, L. Charles, and J.-F. Lutz, "Mass spectrometry sequencing of long digital polymers facilitated by programmed inter-byte fragmentation," *Nature Communications*, vol. 8, no. 1, p. 967, 2017.

[3] N. Goldman, P. Bertone, S. Chen, C. Dessimoz, E. M. LeProust, B. Sipos, and E. Birney, "Towards practical, high-capacity, low-maintenance information storage in synthesized DNA," *Nature*, vol. 494, no. 7435, p. 77, 2013.

[4] R. N. Grass, R. Heckel, M. Puddu, D. Paunescu, and W. J. Stark, "Robust chemical preservation of digital information on DNA in silica with error-correcting codes," *Angewandte Chemie International Edition*, vol. 54, no. 8, pp. 2552–2555, 2015.

[5] S. H. T. Yazdi, Y. Yuan, J. Ma, H. Zhao, and O. Milenkovic, "A rewritable, random-access DNA-based storage system," *Scientific Reports*, vol. 5, p. 14138, 2015.

[6] S. H. T. Yazdi, R. Gabrys, and O. Milenkovic, "Portable and error-free DNA-based data storage," *Scientific Reports*, vol. 7, no. 1, p. 5011, 2017.

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

[8] "Coronavirus Pandemic (COVID-19) ," https://ourworldindata.org/coronavirus-data, 2020, [Online; accessed 11-November-2020].

[9] "Here's how COVID-19 compares to past outbreaks," https://www.healthline.com/health-news/how-deadly-is-the-coronavirus-compared-to-past-outbreaks, 2020, [Online; accessed 20-April-2020].

[10] B. Abdalhamid, C. R. Bilder, E. L. McCutchen, S. H. Hinrichs, S. A. Koepsell, and P. C. Iwen, "Assessment of specimen pooling to conserve SARS CoV-2 testing resources," *American Journal of Clinical Pathology*, 04 2020, aqaa064. [Online]. Available: https://doi.org/10.1093/ajcp/aqaa064

[11] A. Z. Broder and R. Kumar, "A note on double pooling tests," *arXiv e-prints*, Apr. 2020.

[12] C. Gollier, "Optimal group testing to exit the Covid confinement," Toulouse School of Economics, Tech. Rep., Mar. 2020. [Online]. Available: https://www.tse-fr.eu/optimal-group-testing-exit-covid-confinement

[13] O. Gossner, "Group testing against COVID-19," Center for Research in Economics and Statistics, Working Papers 2020-04, Mar. 2020. [Online]. Available: https://ideas.repec.org/p/crs/wpaper/2020-04.html

[14] R. Hanel and S. Thurner, "Boosting test-efficiency by pooled testing strategies for SARS-CoV-2," *arXiv e-prints*, Mar. 2020.

[15] K. R. Narayanan, A. Heidarzadeh, and R. Laxminarayan, "On accelerated testing for COVID-19 using group testing," *arXiv e-prints*, Apr. 2020.

[16] M. Täufer, "Rapid, large-scale, and effective detection of COVID-19 via non-adaptive testing," *bioRxiv*, 2020. [Online]. Available: https://www.biorxiv.org/content/early/2020/04/13/2020.04.06.028431

[17] N. Shental, S. Levy, S. Skorniakov, V. Wuvshet, Y. Shemer-Avni, A. Porgador, and T. Hertz, "Efficient high throughput SARS-CoV-2 testing to detect asymptomatic carriers," *medRxiv*, 2020. [Online]. Available: https://www.medrxiv.org/content/early/2020/04/20/2020.04.14.20064618

[18] I. Yelin, N. Aharony, E. Shaer-Tamar, A. Argoetti, E. Messer, D. Berenbaum, E. Shafran, A. Kuzli, N. Gandali, T. Hashimshony, Y. Mandel-Gutfreund, M. Halberthal, Y. Geffen, M. Szwarcwort-Cohen, and R. Kishony, "Evaluation of COVID-19 RT-qPCR test in multi-sample pools," *medRxiv*, 2020. [Online]. Available: https://www.medrxiv.org/content/early/2020/03/27/2020.03.26.20039438

[19] J. Zhu, K. Rivera, and D. Baron, "Noisy pooled PCR for virus testing," *arXiv e-prints*, Apr. 2020.

[20] G. H. Hardy, "An introduction to the theory of numbers," *Bull. Amer. Math. Soc.*, vol. 35, no. 6, pp. 778–818, 11 1929. [Online]. Available: https://projecteuclid.org:443/euclid.bams/1183493592

[21] V. I. Levenshtein, "Efficient reconstruction of sequences from their subsequences or supersequences," *Journal of Combinatorial Theory, Series A*, vol. 93, no. 2, pp. 310–332, 2001.

[22] M. Dudık and L. J. Schulman, "Reconstruction from subsequences," *Journal of Combinatorial Theory, Series A*, vol. 103, no. 2, pp. 337–348, 2003.

[23] T. Batu, S. Kannan, S. Khanna, and A. McGregor, "Reconstructing strings from random traces," in *Proceedings of the Fifteenth Annual ACM-SIAM Symposium on Discrete Algorithms*. Society for Industrial and Applied Mathematics, 2004, pp. 910–918.

[24] K. Viswanathan and R. Swaminathan, "Improved string reconstruction over insertion-deletion channels," in *Proceedings of the Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms*. Society for Industrial and Applied Mathematics, 2008, pp. 399–408.

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

[26] R. Gabrys and O. Milenkovic, "Unique reconstruction of coded sequences from multiset substring spectra," in *2018 IEEE International Symposium on Information Theory (ISIT)*. IEEE, 2018, pp. 2540–2544.

[27] M. Cheraghchi, R. Gabrys, O. Milenkovic, and J. Ribeiro, "Coded trace reconstruction," *IEEE Transactions on Information Theory*, vol. 66, no. 10, pp. 6084–6103, 2020.

[28] R. Gabrys, H. M. Kiah, and O. Milenkovic, "Asymmetric lee distance codes for DNA-based storage," *IEEE Transactions on Information Theory*, vol. 63, no. 8, pp. 4982–4995, 2017.

[29] D. E. Speyer, "Upper limit on the central binomial coefficient," MathOverflow. [Online]. Available: https://mathoverflow.net/q/246875

[30] S. Durocher, P. C. Li, D. Mondal, F. Ruskey, and A. Williams, "Cool-lex order and k-ary Catalan structures," *Journal of Discrete Algorithms*, vol. 16, pp. 287–307, 2012.

[31] R. Roth, *Introduction to Coding Theory*. Cambridge University Press, 2006.

[32] A. Lenstra, "Factoring multivariate polynomials over finite fields," *Journal of Computer and System Sciences*, vol. 30, no. 2, pp. 235 – 248, 1985. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0022000085900169

[33] D. Y. Grigoryev, "Factoring polynomials over a finite field and solution of systems of algebraic equations," *Theory of the complexity of computations, II., Zap. Nauchn. Sem. Leningrad. Otdel. Mat. Inst. Steklov.(LOMI)*, vol. 137, pp. 124–188, 1984.

[34] E. Kaltofen, "Polynomial-time reductions from multivariate to bi- and univariate integral polynomial factorization," *SIAM Journal on Computing*, vol. 14, no. 2, pp. 469–489, 1985. [Online]. Available: https://doi.org/10.1137/0214035

[35] E. Kaltofen and B. M. Trager, "Computing with polynomials given by black boxes for their evaluations: Greatest common divisors, factorization, separation of numerators and denominators," *Journal of Symbolic Computation*, vol. 9, no. 3, pp. 301 – 320, 1990, computational algebraic complexity editorial. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0747717108800156

[36] K. Launay, J.-A. Amalian, E. Laurent, L. Oswald, A. Al Ouahabi, A. Burel, F. Dufour, C. Carapito, J.-L. Clément, J.-F. Lutz et al., "Precise alkoxyamine-design enables automated tandem mass spectrometry sequencing of digital poly(phosphodiester)s," *Angewandte Chemie*.

[37] S. K. Tabatabaei, B. Wang, N. B. M. Athreya, B. Enghiad, A. G. Hernandez, C. J. Fields, J.-P. Leburton, D. Soloveichik, H. Zhao, and O. Milenkovic, "DNA punch cards for storing data on native DNA sequences via enzymatic nicking," *Nature Communications*, vol. 11, no. 1, pp. 1–10, 2020.

[38] C. Pan, K. Tabatabaei, S. H. T. Yazdi, A. G. Hernandez, C. M. Schroeder, and O. Milenkovic, "Rewritable two-dimensional DNA-based data storage with machine learning reconstruction," *bioRxiv*, 2021.

[39] R. Gabrys, E. Yaakobi, and O. Milenkovic, "Codes in the Damerau distance for deletion and adjacent transposition correction," *IEEE Transactions on Information Theory*, vol. 64, no. 4, pp. 2550–2570, 2017.

[40] R. Gabrys and O. Milenkovic, "Unique reconstruction of coded strings from multiset substring spectra," *IEEE Transactions on Information Theory*, vol. 65, no. 12, pp. 7682–7696, 2019.

[41] R. Gabrys, H. Dau, C. J. Colbourn, and O. Milenkovic, "Set-codes with small intersections and small discrepancies," *SIAM Journal on Discrete Mathematics*, vol. 34, no. 2, pp. 1148–1171, 2020.

[42] A. Agarwal, O. Milenkovic, S. Pattabiraman, and J. Ribeiro, "Group testing with runlength constraints for topological molecular storage," in *2020 IEEE International Symposium on Information Theory (ISIT)*. IEEE, 2020, pp. 132–137.

[43] M. Cheraghchi, R. Gabrys, O. Milenkovic, and J. Ribeiro, "Coded trace reconstruction," *IEEE Transactions on Information Theory*, vol. 66, no. 10, pp. 6084–6103, 2020.

[44] S. Jain, F. F. Hassanzadeh, M. Schwartz, and J. Bruck, "Duplication-correcting codes for data storage in the DNA of living organisms," *IEEE Transactions on Information Theory*, vol. 63, no. 8, pp. 4996–5010, 2017.

[45] N. Raviv, M. Schwartz, and E. Yaakobi, "Rank-modulation codes for DNA storage with shotgun sequencing," *IEEE Transactions on Information Theory*, vol. 65, no. 1, pp. 50–64, 2018.

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

[47] A. Lenz, P. H. Siegel, A. Wachter-Zeh, and E. Yaakobi, "Anchor-based correction of substitutions in indexed sets," in *2019 IEEE International Symposium on Information Theory (ISIT)*. IEEE, 2019, pp. 757–761.

[48] Z. Chang, J. Chrisnata, M. F. Ezerman, and H. M. Kiah, "Rates of DNA sequence profiles for practical values of read lengths," *IEEE Transactions on Information Theory*, vol. 63, no. 11, pp. 7166–7177, 2017.

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

[50] S. Pattabiraman, R. Gabrys, and O. Milenkovic, "Reconstruction and error-correction codes for polymer-based data storage," in *2019 IEEE Information Theory Workshop, ITW 2019, Visby, Sweden, August 25-28, 2019*. IEEE, 2019. [Online]. Available: https://doi.org/10.1109/ITW44776.2019.8989171 pp. 1–5.

[51] R. Gabrys, S. Pattabiraman, and O. Milenkovic, "Mass error-correction codes for polymer-based data storage," in *IEEE International Symposium on Information Theory, ISIT 2020, Los Angeles, CA, USA, June 21-26, 2020*. IEEE, 2020. [Online]. Available: https://doi.org/10.1109/ISIT44484.2020.9174404 pp. 25–30.

[52] S. Pattabiraman, R. Gabrys, and O. Milenkovic, "Coding for polymer-based data storage," *CoRR*, vol. abs/2003.02121, 2020. [Online]. Available: https://arxiv.org/abs/2003.02121

[53] B. Lindstrom, "Determining subsets by unramified experiments," *A Survey of Statistical Design and Linear Models*, 1975.

[54] B. Lindström, "Determination of two vectors from the sum," *Journal of Combinatorial Theory*, vol. 6, no. 4, pp. 402–407, 1969.

[55] B. Lindström, "On b2-sequences of vectors," *Journal of Number Theory*, vol. 4, no. 3, pp. 261–265, 1972.

[56] G. Cohen, S. Litsyn, and G. Zémor, *Journal of Combinatorial Theory, Series A*, vol. 94, no. 1, pp. 152–155, 2001.

[57] A. Naor and J. Verstraëte, "A note on bipartite graphs without 2k-cycles," *Combinatorics, Probability and Computing*, vol. 14, pp. 845 – 849, 2005.

[58] V. Gritsenko, G. Kabatiansky, V. Lebedev, and A. Maevskiy, "On codes for multiple access adder channel with noise and feedback," in *WCC2015 - 9th International Workshop on Coding and Cryptography 2015*, ser. Proceedings of the 9th International Workshop on Coding and Cryptography, 2015.

[59] "Testing the key to reopening economy, returning life to normal, officials say," https://www.deseret.com/utah/2020/4/21/21229734/coronavirus-covid-19-testing-key-economy-reopening- returning-life-normal, 2020, [Online; accessed 20-April-2020].

[60] R. F. Service, "New drool-based tests are replacing the dreaded coronavirus nasal swab," *Science Magazine*, 2020. [Online]. Available: https://www.sciencemag.org/news/2020/08/new-drool-based-tests-are-replacing-dreaded-coronavirus-nasal-swab#

[61] A. Stone, "Nebraska public health lab begins pool testing COVID-19 samples," *KETV Omaha*, 2020.

[62] R. Dorfman, "The detection of defective members of large populations," *The Annals of Mathematical Statistics*, vol. 14, no. 4, pp. 436–440, 1943.

[63] M. Aldridge, O. Johnson, and J. Scarlett, "Group testing: An information theory perspective," *Foundations and Trends in Communications and Information Theory*, vol. 15, no. 3-4, pp. 196–392, 2019.

[64] "Does covid-19 hit women and men differently? U.S. isn't keeping track," https://www.nytimes.com/2020/04/03/us/coronavirus-male-female-data-bias.html, 2020, [Online; accessed 20-April-2020].

[65] "Covid-19's devastating toll on black and Latino Americans, in one chart," www.deseret.com/utah/2020/4/21/21229734/coronavirus-covid-19-te-
sting- key-economy-reopening- returning-life-normal, 2020, [Online; accessed 20-April-2020].

[66] Centers for Disease Control, "CDC 2019-novel coronavirus (2019-nCoV) real-time RT-PCR diagnostic panel," 2020.

[67] C. S. Booth, E. Pienaar, J. R. Termaat, S. E. Whitney, T. M. Louw, and H. J. Viljoen, "Efficiency of the polymerase chain reaction," *Chemical Engineering Science*, vol. 65, no. 17, pp. 4996–5006, 2010.

[68] V. Rana, E. Chien, J. Peng, and O. Milenkovic, "How fast does the SARS-Cov-2 virus really mutate in heterogeneous populations?" *medRxiv*, 2020.

[69] R. Gabrys, S. Pattabiraman, V. Rana, J. Ribeiro, M. Cheraghchi, V. Guruswami, and O. Milenkovic, "Ac-dc: Amplification curve diagnostics for covid-19 group testing," 2020. [Online]. Available: https://arxiv.org/pdf/2011.05223.pdf

[70] J. Yi, R. Mudumbai, and W. Xu, "Low-cost and high-throughput testing of COVID-19 viruses and antibodies via compressed sensing: System concepts and computational experiments," *arXiv e-prints*, Apr. 2020.

[71] H. Bernd Petersen, B. Bah, and P. Jung, "Efficient noise-blind ℓ_1-regression of nonnegative compressible signals," *arXiv e-prints*, Mar. 2020.

[72] S. Ghosh, A. Rajwade, S. Krishna, N. Gopalkrishnan, T. E. Schaus, A. Chakravarthy, S. Varahan, V. Appu, R. Ramakrishnan, S. Ch, M. Jindal, V. Bhupathi, A. Gupta, A. Jain, R. Agarwal, S. Pathak, M. A. Rehan, S. Consul, Y. Gupta, N. Gupta, P. Agarwal, R. Goyal, V. Sagar, U. Ramakrishnan, S. Krishna, P. Yin, D. Palakodeti, and M. Gopalkrishnan, "Tapestry: A single-round smart pooling technique for COVID-19 testing," *medRxiv*, 2020. [Online]. Available: https://www.medrxiv.org/content/early/2020/04/29/2020.04.23.20077727

[73] N. Shental, S. Levy, V. Wuvshet, S. Skorniakov, B. Shalem, A. Ottolenghi, Y. Greenshpan, R. Steinberg, A. Edri, R. Gillis, M. Goldhirsh, K. Moscovici, S. Sachren, L. M. Friedman, L. Nesher, Y. Shemer-Avni, A. Porgador, and T. Hertz, "Efficient high-throughput SARS-CoV-2 testing to detect asymptomatic carriers," *Science Advances*, 2020. [Online]. Available: https://advances.sciencemag.org/content/early/2020/08/20/sciadv.abc5961

[74] M. J. Mina, R. Parker, and D. B. Larremore, "Rethinking covid-19 test sensitivity - A strategy for containment," *New England Journal of Medicine*, 2020. [Online]. Available: https://doi.org/10.1056/NEJMp2025631

[75] R. Arnaout, R. A. Lee, G. R. Lee, C. Callahan, C. F. Yen, K. P. Smith, R. Arora, and J. E. Kirby, "SARS-CoV2 testing: The limit of detection matters," *bioRxiv*, 2020. [Online]. Available: https://www.biorxiv.org/content/early/2020/06/04/2020.06.02.131144

[76] P. Nikolopoulos, T. Guo, C. Fragouli, and S. Diggavi, "Community aware group testing," 2020. [Online]. Available: https://arxiv.org/abs/2007.08111v1

[77] A. Emad and O. Milenkovic, "Semiquantitative group testing," *IEEE Transactions on Information Theory*, vol. 60, no. 8, pp. 4614–4636, 2014.

[78] A. Emad and O. Milenkovic, "Group testing for non-uniformly quantized adder channels," in *2014 IEEE International Symposium on Information Theory*, 2014, pp. 2351–2355.

[79] A. Emad and O. Milenkovic, "Code construction and decoding algorithms for semi-quantitative group testing with nonuniform thresholds," *IEEE Transactions on Information Theory*, vol. 62, no. 4, pp. 1674–1687, 2016.

[80] F. Hwang, "A generalized binomial group testing problem," *Journal of the American Statistical Association*, vol. 70, no. 352, pp. 923–926, 1975.

[81] A. Emad and O. Milenkovic, "Poisson group testing: A probabilistic model for nonadaptive streaming boolean compressed sensing," in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2014, pp. 3335–3339.

[82] J. Wolf, "Born again group testing: Multiaccess communications," *IEEE Transactions on Information Theory*, vol. 31, no. 2, pp. 185–191, 1985.

[83] A. Dyachkov and V. Rykov, "Generalized superimposed codes and their application to random multiple access," in *Proc. 6th Int. Symp. Inf. Theory*, vol. 1, 1984, pp. 62–64.

[84] S. Neidler, "What are the differences between PCR, RT-PCR, qPCR, and RT-qPCR?" https://www.enzolifesciences.com/science-center/technotes/2017/march/what-are-the-differences-between-pcr-rt-pcr-qpcr-and-rt-qpcr?/, 2020.

[85] A. E. Platts, G. D. Johnson, A. K. Linnemann, and S. A. Krawetz, "Real-time PCR quantification using a variable reaction efficiency model," *Analytical Biochemistry*, vol. 380, no. 2, pp. 315–322, 2008.

[86] "GISAID," https://www.gisaid.org.

[87] A. Goyal, E. F. Cardozo-Ojeda, and J. T. Schiffer, "Potency and timing of antiviral therapy as determinants of duration of SARS CoV-2 shedding and intensity of inflammatory response," *medRxiv*, 2020.

[88] Y. Liu, L.-M. Yan, L. Wan, T.-X. Xiang, A. Le, J.-M. Liu, M. Peiris, L. L. M. Poon, and W. Zhang, "Viral dynamics in mild and severe cases of COVID-19," *The Lancet Infectious Diseases*, vol. 20, no. 6, pp. 656 – 657, 2020. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1473309920302322

[89] "On-campus COVID-19 testing," https://covid19.illinois.edu/health-and-support/on-campus-covid-19-testing/, 2020.

[90] P. Indyk, "Sketching via hashing: from heavy hitters to compressed sensing to sparse Fourier transform," in *Proceedings of the 32nd ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, 2013, pp. 87–90.

[91] A. G. D'yachkov and V. V. Rykov, "Bounds on the length of disjunctive codes," *Problemy Peredachi Informatsii*, vol. 18, no. 3, pp. 7–13, 1982.

[92] E. Porat and A. Rothschild, "Explicit non-adaptive combinatorial group testing schemes," in *Automata, Languages and Programming*, L. Aceto, I. Damgård, L. A. Goldberg, M. M. Halldórsson, A. Ingólfsdóttir, and I. Walukiewicz, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 748–759.

[93] J. Scarlett, "An efficient algorithm for capacity-approaching noisy adaptive group testing," in *2019 IEEE International Symposium on Information Theory (ISIT)*. IEEE, 2019, pp. 2679–2683.

[94] F. K. Hwang, "A method for detecting all defective members in a population by group testing," *Journal of the American Statistical Association*, vol. 67, no. 339, pp. 605–608, 1972.

[95] P. Damaschke, "Threshold group testing," in *General Theory of Information Transfer and Combinatorics*. Springer, 2006, pp. 707–718.

[96] M. Cheraghchi, "Improved constructions for non-adaptive threshold group testing," *Algorithmica*, vol. 67, no. 3, pp. 384–417, 2013.

[97] S. Aeron, V. Saligrama, and M. Zhao, "Information theoretic bounds for compressed sensing," *IEEE Transactions on Information Theory*, vol. 56, no. 10, pp. 5111–5130, 2010.

[98] R. G. Baraniuk, "Compressive sensing [lecture notes]," *IEEE Signal Processing Magazine*, vol. 24, no. 4, pp. 118–121, 2007.

[99] J. A. Tropp, "Greed is good: Algorithmic results for sparse approximation," *IEEE Transactions on Information Theory*, vol. 50, no. 10, pp. 2231–2242, 2004.

[100] W. Dai and O. Milenkovic, "Subspace pursuit for compressive sensing signal reconstruction," *IEEE Transactions on Information Theory*, vol. 55, no. 5, pp. 2230–2249, 2009.

[101] S. Ghosh, R. Agarwal, M. A. Rehan, S. Pathak, P. Agrawal, Y. Gupta, S. Consul, N. Gupta, R. Goyal, and A. Rajwade, "A compressed sensing approach to group-testing for COVID-19 detection," *arXiv preprint arXiv:2005.07895*, 2020.

[102] W. Dai, M. A. Sheikh, O. Milenkovic, and R. G. Baraniuk, "Compressive sensing DNA microarrays," *EURASIP Journal on Bioinformatics and Systems Biology*, vol. 2009, no. 1, p. 162824, 2008.

[103] W. Dai, H. V. Pham, and O. Milenkovic, "A comparative study of quantized compressive sensing schemes," in *2009 IEEE International Symposium on Information Theory*. IEEE, 2009, pp. 11–15.

[104] W. Dai and O. Milenkovic, "Information theoretical and algorithmic approaches to quantized compressive sensing," *IEEE Transactions on Communications*, vol. 59, no. 7, pp. 1857–1866, 2011.

[105] A. Heidarzadeh and K. R. Narayanan, "Two-stage adaptive pooling with RT-qPCR for COVID-19 screening," *arXiv preprint arXiv:2007.02695*, 2020.

[106] W. Dai and O. Milenkovic, "Weighted superimposed codes and constrained integer compressed sensing," *IEEE Transactions on Information Theory*, vol. 55, no. 5, pp. 2215–2229, 2009.

[107] M. Cheraghchi, A. Karbasi, S. Mohajer, and V. Saligrama, "Graph-constrained group testing," *IEEE Transactions on Information Theory*, vol. 58, no. 1, pp. 248–262, 2012.

[108] S. Ahn, W.-N. Chen, and A. Ozgur, "Adaptive group testing on networks with community structure," *arXiv preprint arXiv:2101.02405*, 2021.

[109] P. Nikolopoulos, S. R. Srinivasavaradhan, T. Guo, C. Fragouli, and S. Diggavi, "Group testing for overlapping communities," *arXiv preprint arXiv:2012.02804*, 2020.

[110] Y.-J. Lin, C.-H. Yu, T.-H. Liu, C.-S. Chang, and W.-T. Chen, "Positively correlated samples save pooled testing costs," 2020. [Online]. Available: https://arxiv.org/pdf/2011.09794.pdf

[111] C. J. Colbourn, "Group testing for consecutive positives," https://link.springer.com/article/10.1007/BF01609873, 1999.

[112] D. Du, F. K. Hwang, and F. Hwang, *Combinatorial Group Testing and Its Applications*. World Scientific, 2000.

[113] M. Cheraghchi, R. Gabrys, and O. Milenkovic, "Semiquantitative group testing in at most two rounds," *arXiv preprint arXiv:2102.04519*, 2021.

[114] M. Hahn-Klimroth and P. Loick, "Optimal adaptive group testing," *arXiv e-prints*, Nov. 2019.

APPENDIX A

SUPPLEMENTARY MATERIAL TO CHAPTER 2

A.1 Proof of the Second Part of Theorem 6

Theorem. *The central binomial coefficient* $\binom{2m}{m}$ *counts the following types of binary strings of length* $2m$*.*
(A) Those whose every prefix has at least as many 0*s as* 1*s.*
(B) Those whose every prefix has strictly more 0*s than* 1*s, or vice-versa.*

Proof. The number of binary strings of Type (A) of length $\ell = a + b$, with ℓ possibly odd, such that the number of 0s is greater or equal to the number of 1s, i.e., $a \geqslant b$ is given by $\binom{\ell}{a} - \binom{\ell}{a+1}$. The number of strings for which every prefix has at least as many 0s as 1s is given by $\sum_{a \geqslant \lceil \frac{\ell}{2} \rceil} \binom{\ell}{a} - \binom{\ell}{a+1}$, which is a telescoping sum that equals $\binom{\ell}{\lceil \frac{\ell}{2} \rceil}$.
To prove (B), let us consider strings of length $2m$ whose every prefix has strictly more 0s than 1s. In this case, the first bit of any string $\mathbf{s}$ is always 0. Thus, the remaining length-$(2m-1)$ binary string $\mathbf{s}_2^{2m}$ is such that for every prefix, the number of 0s is at least as large as the number of 1s in that same prefix. Thus, the number of strings of length $2m$ whose every prefix has strictly more 0s than 1s is $\binom{2m-1}{m}$. As a result, the total number of binary strings of length $2m$ whose every prefix has strictly more 0s than 1s or vice-versa is equal to $2\binom{2m-1}{m} = \binom{2m}{m}$. □

A.2 Derivation of the Lower Bound of $|\mathcal{S}_R(n)|$

Let n be even. Note that all strings $\mathbf{s} \in \mathcal{S}_R(n)$ satisfy $s_1 = 0$ and $s_n = 1$. Let $|\left[\frac{n}{2}\right] \cap I| = i + 1$. Thus, the indices corresponding to the Catalan-Bertrand string can be chosen in $\binom{\frac{n}{2}-1}{i}$ ways. Since $s_1 = 0$, it must be that every prefix of $\mathbf{s}_{\left[\frac{n}{2}\right] \cap I \setminus \{1\}}$ contains at least as many 0s as 1s. There are $\binom{i}{\lfloor \frac{i}{2} \rfloor}$ such binary strings of length i. Therefore,

$$|\mathcal{S}_R(n)| = \sum_{i=0}^{\frac{n}{2}-1} \binom{\frac{n}{2}-1}{i} 2^{\frac{n}{2}-1-i} \binom{i}{\lfloor \frac{i}{2} \rfloor}.$$

As a result,

$$\sum_{i=0}^{\frac{n}{2}-1} \binom{\frac{n}{2}-1}{i} 2^{\frac{n}{2}-1-i} \binom{i}{\lfloor \frac{i}{2} \rfloor} \tag{A.1}$$

$$\geqslant \sum_{i=2}^{\frac{n}{2}-1} \binom{\frac{n}{2}-1}{i} 2^{\frac{n}{2}-1-i} \frac{2^{i-1}}{\sqrt{\pi(i+1)}} + \binom{\frac{n}{2}-1}{1} 2^{\frac{n}{2}-1-1} + \binom{\frac{n}{2}-1}{0} 2^{\frac{n}{2}-1} \tag{A.2}$$

$$\geqslant \frac{2^{\frac{n}{2}-2}}{\sqrt{\pi n}} \sum_{i=0}^{\frac{n}{2}-1} \binom{\frac{n}{2}-1}{i} \tag{A.3}$$

$$= \frac{2^{\frac{n}{2}-2}}{\sqrt{\pi n}} 2^{\frac{n}{2}-1} = \frac{1}{\sqrt{\pi n}} 2^{n-3}. \tag{A.4}$$

Expression (A.1) follows from the description of the codebook. Also, $\binom{2\ell+1}{\ell} \geqslant \binom{2\ell}{\ell}$ clearly holds. As a result, inequality (A.2) follows from Proposition 1, for all $i \geqslant 2$. Inequality (A.3) holds since for all $0 \leqslant i \leqslant \frac{n}{2}$, $(i+1) \leqslant n$. The next two equalities in (A.4) follow from the fact that $\sum_{i=0}^{\ell} \binom{\ell}{i} = 2^{\ell}$, and some rearrangements of terms.

For odd n, $|\mathcal{S}_R(n)| = 2|\mathcal{S}_R(n-1)| \geqslant 2\frac{2^{n-1-3}}{\sqrt{\pi(n-1)}} \geqslant \frac{2^{n-3}}{\sqrt{\pi n}}$.

A.3 A Bijective Map between Information Strings and Reconstructable Strings

An optimal approach for performing encoding of information strings into Catalan string was first described in [30] and it relies on using a ranking/unranking scheme of complexity $\mathcal{O}(n)$. However, we provide a much simpler method to order and retrieve the reconstructable strings in additive $\mathcal{O}(n^2)$ time, which is still absorbed in the leading complexity term of $\mathcal{O}(n^3)$ incurred by the Backtracking algorithm.

Recall that the reconstruction code is obtained by interleaving arbitrary, unconstrained strings with Catalan-Bertrand strings and then mirroring the interleaved string around what will be the midpoint of the resulting code-string.

1) The construction starts by partitioning the first $\lfloor \frac{n}{2} \rfloor$ indices into two sets, say $\mathcal{I}_0$ and $\mathcal{I}_1$, the cardinalities of which are in $\{0, \ldots, \lfloor \frac{n}{2} \rfloor\}$. Let $\mathcal{I}_0$ denote the set of indices that describe the locations of the string to be interleaved, and let $\mathcal{I}_1$ denote the indices that describe the locations of the Catalan-Bertrand string.

Next, order all possible partitions according to the cardinality of their corresponding $\mathcal{I}_0$ sets, in increasing order. For example, if 000111010 and 001111010 are the labels of two partitions of a string of length 9, than 001111010 appears in the rank-ordered list before 000111010 (the first partition has $|\mathcal{I}_0| = 4$, while the second partition has $|\mathcal{I}_0| = 5 > 4$).

In the next step, order the partitions with the same value of $|\mathcal{I}_0|$. Given a partition described using the binary alphabet as above, one can convert the binary strings into integers and arrange them in increasing order which naturally induces a ranking of the partitions themselves. Finding the index of a partition in this ranking takes $\mathcal{O}(n)$ time. To see this, consider a partition Π of $m = \lfloor \frac{n}{2} \rfloor$ indices, and let $|\mathcal{I}_0| = i$.

Thus, the rank of this partition is an integer in the interval

$$\left[\sum_{j=0}^{i-1} \binom{m}{j} + 1, \sum_{j=0}^{i} \binom{m}{j} \right].$$

Assume that the set $\mathcal{I}_0$ contains the indices $(\ell_1, \ell_2, \ldots, \ell_i)$ arranged in increasing order. The rank of the partition Π is given by

$$\sum_{j=0}^{i-1} \binom{m}{j} + \left[\binom{\ell_i - 1}{i} + \binom{\ell_{i-1} - 1}{i-1} + \binom{\ell_{i-2} - 1}{i-2} + \cdots + \binom{\ell_1 - 1}{1} + 1 \right].$$

Therefore, given the index of a partition, one can determine the actual partition in time $\mathcal{O}(n^2)$.

2) Next, given the indices in $\mathcal{I}_0$, place unrestricted binary strings in the corresponding locations according to the lexicographical order.

3) At indices in $\mathcal{I}_1$, place bits of a Catalan-Bertrand string. Let us now assume that there exists a bijective map $F_m(\cdot)$ that for all natural numbers m orders all Catalan-Bertrand strings of length m efficiently. In particular, we assume that given an index `ind`, $F_m(\texttt{ind})$ returns the corresponding Catalan-Bertrand string in time $\mathcal{O}(n^2)$. Further, given a Catalan-Bertrand string $\mathbf{s}$, $F_m^{-1}(\mathbf{s})$ returns its index `ind` in $\mathcal{O}(n)$ time. We defer the description of the map to the end of this exposition.

Let $f_m(i)$ denote the number of Catalan-Bertrand strings with $m - i$ 0s and i 1s. Then, $f_m = \sum_{i=0}^{\lfloor \frac{m}{2} \rfloor} f_m(i)$ is the number of all Catalan-Bertrand strings of length m. Note that $f_m(i)$ has a closed form expression as given in Theorem 6, and f_m equals $\frac{1}{2}\binom{m}{\lfloor \frac{m}{2} \rfloor}$.

The ordering for the codestrings of the reconstruction code is obtained as follows:

a) Given two reconstructable codestrings $\mathbf{s}_1$, and $\mathbf{s}_2$, and their corresponding partitions Π_1 and Π_2 from 1, if Π_1 is ranked lower than Π_2, then $\mathbf{s}_1$ is ranked lower than $\mathbf{s}_2$.

b) Given two reconstructable codestrings $\mathbf{s}_1$, and $\mathbf{s}_2$ such that $\Pi_1 = \Pi_2$, if the string of $\mathbf{s}_1$ indexed by $\mathcal{I}_0$ is ranked lower than that of $\mathbf{s}_2$ (as per 2)), then $\mathbf{s}_1$ is ranked lower than $\mathbf{s}_2$.

c) Given two reconstructable codestrings $\mathbf{s}_1$ and $\mathbf{s}_2$ such that $\Pi_1 = \Pi_2$, and the strings of $\mathbf{s}_1$ and $\mathbf{s}_2$ indexed by $\mathcal{I}_0$ are the same, if the string indexed by $\mathcal{I}_2$ in $\mathbf{s}_1$ is ranked lower than the string in $\mathbf{s}_2$, then the string $\mathbf{s}_1$ is ranked lower than $\mathbf{s}_2$.

In summary, the reconstructable codestrings are encoded and decoded as described below.

Encoding:

A $k-$bit binary string is converted into an index `ind`. The time taken to find the corresponding partition and Catalan-Bertrand string is $\mathcal{O}(n)$. Combining this result with the result pertaining to the ranking map proves that the information string can be encoded in $\mathcal{O}(n^2)$ time.

Decoding:

Given a reconstructable codestring, its index can be computed in $\mathcal{O}(n)$ time. The $k-$bit binary expansion of the index uniquely determines the information string. Since the Backtracking algorithm takes $\mathcal{O}(n^3)$ time, the overall decoding time equals $\mathcal{O}(n^3)$.

It remains to show that encoding and decoding of the Catalan-Bertrand strings can be performed in time $\mathcal{O}(n^2)$. Since the decoding process is easier to describe and leads to a straightforward approach for encoding, we start with the description of the decoding algorithm.

Decoding Catalan-Bertrand strings: Let $\mathbf{s} = s_1 s_2 \dots s_{m-1} s_m$ denote a Catalan-Bertrand string of length m that contains $m - i$ 0s and i 1s and recall that $f_m(i)$ denotes the number of such Catalan-Bertrand strings. We start by ranking the string $\mathbf{s}$ against the set of all Catalan-Bertrand strings of length m that contain $m - i$ 0s and i 1s. The following simple algorithm determines the temporary index for $\mathbf{s}$ in $\mathcal{O}(n)$ time.

$\texttt{ind}_{temp} \leftarrow f_m(i)$

$l \leftarrow i$

`for` j `from` 0 `to` $m-1$:

$\quad \texttt{ind}_{temp} = \texttt{ind}_{temp} - \mathbf{1}_{\{s_{m-j}==0\}} f_{m-1-j}(l)$

$\quad$ `if` $s_{m-j} == 1$:

$\quad\quad l \leftarrow l - 1,$

where **1** denotes the indicator function. Note that $F_m(\cdot)$ then assigns the final index value $\sum_{\ell<i} f_m(i) + \texttt{ind}_{temp}$ to the given Catalan-Bertrand string $\mathbf{s}$ of length m in time $\mathcal{O}(n)$.

Encoding Catalan-Bertrand strings: From the decoding procedure it is easy to deduce how to perform the encoding: Given m and $\texttt{ind}$, we first find an i such that $\mathbf{s}$ has $m-i$ 0s and i 1s. Then, iteratively, the bits s_m through s_1 are computed using the correspondence between the bit value and the index range as described in the decoding process. Hence, encoding takes $\mathcal{O}(n^2)$ time.

A.4 Proof of Lemma 4

Recall that we consider asymmetric errors, in which case a single error may occur either in C_j or C_{n+1-j} but not both multisets. Furthermore, up to t such errors are allowed. The presented code corrects such errors with at most $c_1 t \log k + c_2$ bits of redundancy, where k is the length of the information string, and c_1, and c_2 are two positive constants.

The code construction involves two parts:
(1) String $\mathbf{s} \in \mathcal{S}_R(n_1)$ is padded with a prefix of t 0s and a suffix of t 1s to form an intermediate string $\mathbf{s}'$ of length $n' + 2t$.
(2) The $\Sigma^{\frac{n_1}{2}}$ is then encoded using a systematic $t-$ error correcting code and the redundant bits are placed in the middle of the string in manner such that the resultant string $\mathbf{s}'' \in \mathcal{S}_R(n)$.

We show through a case-by-case analysis that the code is indeed a $t-$asymmetric error correcting code.

Our analysis proceeds through multiple steps addressing different possible choices for the values of $\sigma_i, i = 1, \ldots, \frac{n}{2}$, and the currently reconstructed bits (i.e., prefixes and suffixes of the codestring). The initial setting is depicted in Figure A.1. Each subsequent figure (Figures A.2, A.3, A.4, A.5, A.6, A.7, A.8 and A.9) explains how to extend two partially reconstructed strings from their prefix and suffix pairs so as to minimize the number of compositions they disagree in. For simplicity, such pairs are termed "confusable" and finding confusable pairs allows us to determine the minimum composition set differences between codestrings based on the Catalan-Bertrand construction. The final result establishes that the previous construction ensures a minimum composition set difference $\geqslant 2(t+1)$.

First, we observe from Construction (2.4) that any pair of distinct strings $\mathbf{s}, \mathbf{v} \in \mathcal{S}_R^{(t)}(m)$ shares a prefix-suffix pair of length at least t as all strings are padded by 0s and 1s on the left and right, respectively.

Next, we characterize the conditions that allow one to identify strings that are "closest" to a codestring $\mathbf{s}$. More precisely, we construct a set $\mathcal{V}_{\mathbf{s}}$ of strings such that for all $\mathbf{v} \in \mathcal{V}_{\mathbf{s}}$: (1) $\mathbf{v}$ and $\mathbf{s}$ share the same $\Sigma^{\frac{m}{2}}$ sequence. (2) If the length of the longest shared prefix-suffix pair of $\mathbf{v}$ and $\mathbf{s}$ equals i, then for all $j \in \{m-i-1, m-i-2, \ldots, m-i-t-1\}$ the inequality $|C_j(\mathbf{s}) \setminus C_j(\mathbf{v})| \leqslant 2$ holds. These conditions summarize when a string may be confused with $\mathbf{s}$ during the backtracking reconstruction procedure.

Recall that $c(\cdot)$ refers to the composition of its argument string. The substrings $\{\mathbf{s}_i^{i+j-1}\}, i = 1, \ldots, m-j+1$ of $\mathbf{s}$ of length j share a common substring $\mathbf{s}_{m+1-j}^{j}$, provided that $j > \frac{m}{2}$. For simplicity of notation, denote the composition of the common substring $\mathbf{s}_{m+1-j}^{j}$ by c_j, *i.e.*, let $c_j = c(\mathbf{s}_{m+1-j}^{j})$.

We start with the following observation. If $\sigma_{i+1} \neq 1$, the two strings $\mathbf{s}$ and $\mathbf{v}$ necessarily share a prefix-suffix pair of length $i+1$, which contradicts the assumption that the longest prefix-suffix pair shared by the two strings is of length i. Thus, we have $\sigma_{i+1} = 1$ and $|C_{m-i-1}(\mathbf{s}) \setminus C_{m-i-1}(\mathbf{v})| = 2$, where the latter claim follows from the discussion pertaining to the single error-correction case: The compositions of length $m-i-1$ that are not shared by the two strings include $\{c(\mathbf{s}_1^i), 0, c_{m-i-1}\}$, $\{c(\mathbf{s}_{m+1-i}^m), 1, c_{m-i-1}\}$, $\{c(\mathbf{v}_1^i), 1, c_{m-i-1}\}$, $\{c(\mathbf{v}_{m+1-i}^m), 0, c_{m-i-1}\}$, and these differ by construction.

Next, we describe how to simultaneously reconstruct a pair of prefix-suffix bits and update the set $\mathcal{V}_{\mathbf{s}}$ when taking a step in the Backtracking algorithm. We show that under the conditions of the lemma, $|C_{m-i-1-j}(\mathbf{s}) \setminus C_{m-i-1-j}(\mathbf{v})| = 2$ for all $\mathbf{v} \in \mathcal{V}_{\mathbf{s}}, 1 \leqslant j \leqslant t$. For notational simplicity, at every step of the reconstruction algorithm we use the index "$+$" to denote the next bit in the prefix and "$-$" to denote the next bit in the suffix to be reconstructed. As an example, for a reconstructed prefix-suffix pair of length $i+1$, $+$ corresponds to $i+2$ and $-$ corresponds to $m-i-1$, *i.e.*, $s_+ = s_{i+2}$ and $s_- = s_{m-i-1}$.

Let $\sigma_+ = \mathrm{wt}(s_+ s_-) = \mathrm{wt}(v_+ v_-)$. We analyze the two cases $\sigma_+ = 1$ and $\sigma_+ \in \{0, 2\}$ separately, as depicted in Figure A.1.

Consider the case that $\sigma_+ = 1$. Note that for any substring $\mathbf{s}_{\ell_1}^{\ell_2}$ such that $\ell_1 \leqslant i+1, m-i \leqslant \ell_2$, the corresponding substring $\mathbf{v}_{\ell_1}^{\ell_2}$ of $\mathbf{v}$ has the same composition. The compositions in $C_{m-i-2}(\mathbf{s})$ and $C_{m-i-2}(\mathbf{v})$ that may be confused are listed below on the left and right hand side of the equality, respectively:

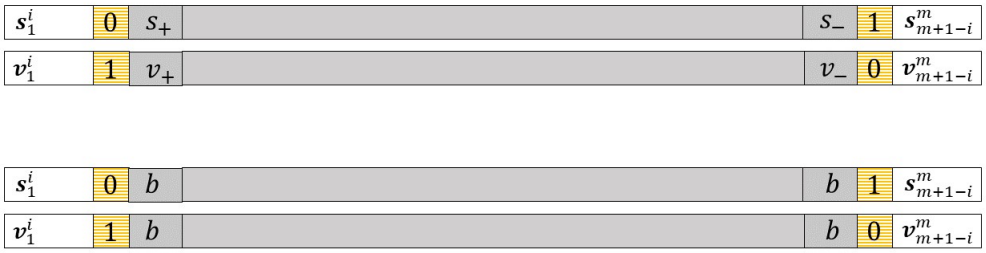


Figure A.1: Illustration of two strings $\mathbf{s}$ and $\mathbf{v}$ that share the same $\Sigma^{\frac{m}{2}}$ sequence. Furthermore, the two strings also satisfy $\mathbf{s}_1^i = \mathbf{v}_1^i$, $\mathbf{s}_{m+1-i}^m = \mathbf{v}_{m+1-i}^m$ and $s_{i+1} \neq v_{i+1}$, *i.e.*, the longest prefix-suffix pair that the strings share is of length i. The top pair of strings corresponds to the case $\sigma_{i+2} = 1$, while the bottom pair of strings corresponds to the case $\sigma_{i+2} \in \{0, 2\}$.

$$\left\{\begin{array}{l} \{c(\mathbf{s}_1^i), 0, s_+, c_{m-i-2}\}, \\ \{c(\mathbf{s}_2^i), 0, s_+, c_{m-i-2}, 1-s_+\}, \\ \{c(\mathbf{s}_{m-i+1}^m), 1, 1-s_+, c_{m-i-2}\}, \\ \{c(\mathbf{s}_{m-i+1}^{m-1}), 1, 1-s_+, c_{m-i-2}, s_+\} \end{array}\right\} = \left\{\begin{array}{l} \{c(\mathbf{v}_1^i), 1, v_+, c_{m-i-2}\}, \\ \{c(\mathbf{v}_2^i), 1, v_+, c_{m-i-2}, 1-v_+\}, \\ \{c(\mathbf{v}_{m-i+1}^m), 0, 1-v_+, c_{m-i-2}\}, \\ \{c(\mathbf{v}_{m-i+1}^{m-1}), 0, 1-v_+, c_{m-i-2}, v_+\} \end{array}\right\}.$$

We want to determine under which conditions the terms on the two sides of the equality can be perfectly matched; in the process, we will show that $|c_{m-i-2}(\mathbf{s}) \setminus c_{m-i-2}(\mathbf{v})| \leqslant 2$.

The above sets may be more succinctly written as:

$$\left\{\begin{array}{l}\{c(\mathbf{s}_1^i), 0, s_+, c_{m-i-2}\},\\ \{c(\mathbf{s}_2^i), 0^2 1, c_{m-i-2}\},\\ \{c(\mathbf{s}_{m-i+1}^m), 1, 1-s_+, c_{m-i-2}\},\\ \{c(\mathbf{s}_{m-i+1}^{m-1}), 01^2, c_{m-i-2}\}\end{array}\right\} = \left\{\begin{array}{l}\{c(\mathbf{v}_1^i), 1, v_+, c_{m-i-2}\},\\ \{c(\mathbf{v}_2^i), 01^2, c_{m-i-2}\},\\ \{c(\mathbf{v}_{m-i+1}^m), 0, 1-v_+, c_{m-i-2}\},\\ \{c(\mathbf{v}_{m-i+1}^{m-1}), 0^2 1, c_{m-i-2}\}\end{array}\right\}.$$

Regrouping the a priori known extension bits with the prefixes and suffixes simplifies the sets to be matched as

$$\left\{\begin{array}{l}\{c(\mathbf{s}_1^i), 0, s_+, c_{m-i-2}\},\\ \{c(\mathbf{s}_1^i), 01, c_{m-i-2}\},\\ \{c(\mathbf{s}_{m-i+1}^m), 1, 1-s_+, c_{m-i-2}\},\\ \{c(\mathbf{s}_{m-i+1}^m), 01, c_{m-i-2}\}\end{array}\right\} = \left\{\begin{array}{l}\{c(\mathbf{v}_1^i), 1, v_+, c_{m-i-2}\},\\ \{c(\mathbf{v}_1^i), 1^2, c_{m-i-2}\},\\ \{c(\mathbf{v}_{m-i+1}^m), 0, 1-v_+, c_{m-i-2}\},\\ \{c(\mathbf{v}_{m-i+1}^m), 0^2, c_{m-i-2}\}\end{array}\right\}.$$

For example, $\{c(\mathbf{s}_2^i), 0^2 1, c_{m-i-2}\}$ is rewritten as $\{c(\mathbf{s}_1^i), 0^1 1, c_{m-i-2}\}$ by moving one 0 to the prefix composition.

Next, we remove the compositions c_{m-i-2} shared by the two sets. Then we identify which compositions cannot be matched as follows. First, it follows from the construction that the composition of a prefix of length $i > t$ includes at least $t+1$ 0s. As a result, $c(\mathbf{s}_1^i)$ is composed of at least $t+1$ more 0s than $c(\mathbf{s}_{m-i+1}^m)$. Similarly, $c(\mathbf{s}_{m-i+1}^m)$ is composed of at least $t+1$ more 1s than $c(\mathbf{s}_1^i)$. Hence, a composition involving less than $i+t+1$ bits that contains a composition of a prefix of length $i > t$ is composed of more 0s than a composition of the same length that contains a composition of a suffix of length i. Thus, compositions $\{c(\mathbf{s}_1^i), 0, s_+)\}, \{c(\mathbf{s}_1^i), 01)\}$ are not the same as either of the compositions $\{c(\mathbf{v}_{m-i+1}^m), 0, 1-v_+\}, \{c(\mathbf{v}_{m-i+1}^m), 0^2\}$, since $c(\mathbf{s}_1^i)$ contains at least $t+1$ more 0s than $c(\mathbf{v}_{m-i+1}^m)$. Therefore, we only need to consider the two reduced set equalities:

$$\left\{\begin{array}{l}\{c(\mathbf{s}_1^i), 0, s_+\},\\ \{c(\mathbf{s}_1^i), 01\}\end{array}\right\} = \left\{\begin{array}{l}\{c(\mathbf{v}_1^i), 1, v_+\},\\ \{c(\mathbf{v}_1^i), 1^2\}\end{array}\right\},$$

and

$$\begin{Bmatrix} \{c(\mathbf{s}_{m-i+1}^m), 1, 1-s_+\}, \\ \{c(\mathbf{s}_{m-i+1}^m), 01\} \end{Bmatrix} = \begin{Bmatrix} \{c(\mathbf{v}_{m-i+1}^m), 0, 1-v_+\}, \\ \{c(\mathbf{v}_{m-i+1}^m), 0^2\} \end{Bmatrix}.$$

Clearly, $\{c(\mathbf{v}_1^i), 1^2\}$ and $\{c(\mathbf{v}_{m-i+1}^m), 0^2\}$ cannot be equal to any other composition in the two sets. The possible values for the set difference $|C_{m-i-2}(\mathbf{s}) \setminus C_{m-i-2}(\mathbf{v})|$ for four different assignments of values for (s_+, v_+) are summarized in Table A.1. Based on the table, if $\sigma_{i+2}(\mathbf{s}) = 1$, then all strings $\mathbf{v} \in \mathcal{V}_\mathbf{s}$ satisfy $(s_+, v_+) = (s_{i+2}, v_{i+2}) \in \{(0,0), (1,0)\}$.

Next, we consider the case $\sigma_+ \in \{0, 2\}$. As before, we focus on $C_{m-i-2}(\mathbf{s})$ and $C_{m-i-2}(\mathbf{v})$ in order to establish conditions under which $|C_{m-i-2}(\mathbf{s}) \setminus C_{m-i-2}(\mathbf{v})|$ is minimized.

To this end, let $b = s_+ = v_+ = s_- = v_-$. Following the previously outlined line of reasoning, it suffices to find when the following set equalities hold:

$$\begin{Bmatrix} \{c(\mathbf{s}_1^i), 0, b\}, \\ \{c(\mathbf{s}_1^i), 01\} \end{Bmatrix} = \begin{Bmatrix} \{c(\mathbf{v}_1^i), 1, b\}, \\ \{c(\mathbf{v}_1^i), 1^2\} \end{Bmatrix}$$

and

$$\begin{Bmatrix} \{c(\mathbf{s}_{m-i+1}^m), 1, b\}, \\ \{c(\mathbf{s}_{m-i+1}^m), 01\} \end{Bmatrix} = \begin{Bmatrix} \{c(\mathbf{v}_{m-i+1}^m), 0, b\}, \\ \{c(\mathbf{v}_{m-i+1}^m), 0^2\} \end{Bmatrix}.$$

It can be easily seen that the compositions cannot be matched. The possible cardinalities of the set difference $|C_{m-i-2}(\mathbf{s}) \setminus C_{m-i-2}(\mathbf{v})|$ are summarized in Table A.2.

As a result of the above discussion, for any $\mathbf{v} \in \mathcal{V}_\mathbf{s}$ we necessarily have $(s_{i+2}, v_{i+2}) \in \{(0,0), (1,0)\}$ and $\sigma_{i+2} = 1$. This consequently determines the pair of bits s_{m-i-1} and v_{m-i-1}.

To determine s_{i+3},s_{m-i-2}, v_{i+3} and v_{m-i-2} we need to once again analyze two cases, one for which we assume that $\sigma_{i+3} = 1$ and another for which we assume that $\sigma_{i+3} \in \{0, 2\}$. This analysis has to be performed in the context depicted in Figure A.1, and under the constraints imposed by Tables A.1 and A.2.

We focus on the bits $s_{i+2+i'}$ and $v_{i+2+i'}$ for some i' such that $t-1 \geqslant i' \geqslant 0$, in the following inductive setting:

- Assume that starting from the index $i+2$, the values of σ corresponding i' consecutive positions all equal to 1. More precisely, $\sigma_{i+2}^{i+1+i'} = (1, 1, \ldots 1)$.
- The bits s_{i+1} and v_{i+1} are followed by a run of i' 0s, *i.e.*, $\mathbf{s}_{i+2}^{i+1+i'} = \mathbf{v}_{i+2}^{i+1+i'} = \mathbf{0}$.

This setting is depicted in Figure A.2. We proceed to characterize the conditions under which $|C_{m-i-i'-2}(\mathbf{s}) \setminus C_{m-i-i'-2}(\mathbf{v})|$ is minimized. As done before, we consider the cases $\sigma_{i+2+i'} = 1$ and $\sigma_{i+3+i'} \in \{0, 2\}$ separately.

When $\sigma_+ \in \{0, 2\}$, we assume that $s_+ = s_- = v_+ = v_- = b$. The set equality of interest reads as:

$$\left\{\begin{array}{l} \{c(\mathbf{s}_1^i), 0, 0^{i'}, s_+\}, \\ \{c(\mathbf{s}_2^i), 0, 0^{i'}, 01\}, \\ \{c(\mathbf{s}_3^i), 0, 0^{i'}, 01^2\} \\ \{c(\mathbf{s}_4^i), 0, 0^{i'}, 01^3\} \\ \qquad \vdots \\ \{c(\mathbf{s}_{i'+2}^i), 0, 0^{i'}, 01^{i'+1}\} \\ \{c(\mathbf{s}_{m-i+1}^m), 1, 1^{i'}, 1 - s_+\}, \\ \{c(\mathbf{s}_{m-i+1}^{m-1}), 1, 1^{i'}, 01\}, \\ \{c(\mathbf{s}_{m-i+1}^{m-2}), 1, 1^{i'}, 0^2 1\}, \\ \{c(\mathbf{s}_{m-i+1}^{m-3}), 1, 1^{i'}, 0^3 1\}, \\ \qquad \vdots \\ \{c(\mathbf{s}_{m-i+1}^{m-i'-1}), 1, 1^{i'}, 0^{i'+1} 1\} \end{array}\right\} = \left\{\begin{array}{l} \{c(\mathbf{v}_1^i), 1, 0^{i'}, v_+\}, \\ \{c(\mathbf{v}_2^i), 1, 0^{i'}, 01\}, \\ \{c(\mathbf{v}_3^i), 1, 0^{i'}, 01^2\} \\ \{c(\mathbf{v}_4^i), 1, 0^{i'}, 01^3\} \\ \qquad \vdots \\ \{c(\mathbf{v}_{i'+2}^i), 1, 0^{i'}, 01^{i'+1}\} \\ \{c(\mathbf{v}_{m-i+1}^m), 0, 1^{i'}, 1 - v_+\}, \\ \{c(\mathbf{v}_{m-i+1}^{m-1}), 0, 1^{i'}, 01\}, \\ \{c(\mathbf{v}_{m-i+1}^{m-2}), 0, 1^{i'}, 0^2 1\}, \\ \{c(\mathbf{v}_{m-i+1}^{m-3}), 0, 1^{i'}, 0^3 1\}, \\ \qquad \vdots \\ \{c(\mathbf{v}_{m-i+1}^{m-i'-1}), 0, 1^{i'}, 0^{i'+1} 1\} \end{array}\right\}.$$

Using the same line of reasoning as presented earlier, one can show that it suffices to focus on two reduced set equalities, namely

$$\left\{\begin{array}{l} \{c(\mathbf{s}_1^i), 0^{i'+1}, s_+\}, \\ \{c(\mathbf{s}_1^i), 0^{i'+1} 1\} \end{array}\right\} = \left\{\begin{array}{l} \{c(\mathbf{v}_1^i), 0^{i'} 1, v_+\}, \\ \{c(\mathbf{v}_1^i), 1^{i'+2}\} \end{array}\right\},$$

and

$$\left\{\begin{array}{l} \{c(\mathbf{s}_{m-i+1}^m), 01^{i'+1}\}, \\ \{c(\mathbf{s}_{m-i+1}^m), 1^{i'+1}, 1 - s_+\} \end{array}\right\} = \left\{\begin{array}{l} \{c(\mathbf{v}_{m-i+1}^m), 0^{i'+2}\}, \\ \{c(\mathbf{v}_{m-i+1}^m), 01^{i'}, 1 - v_+\} \end{array}\right\}.$$

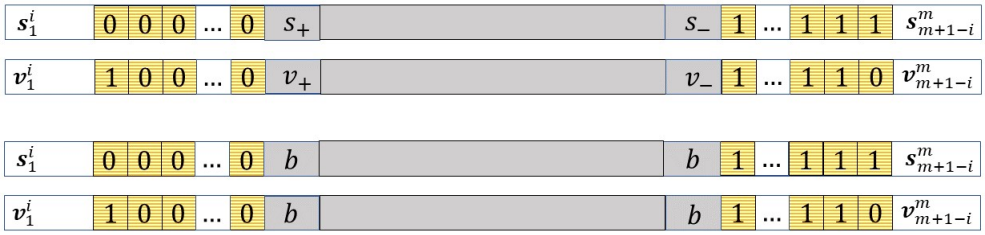


Figure A.2: Illustration of the setup for determining the bits s_+, s_-, v_+ and v_- under the conditions that the bits s_{i+1} and v_{i+1} are followed by a run of i' 0s, and $\sigma_{i+2}^{i+1+i'} = (1, 1, \ldots, 1)$. The second pair of strings illustrates the setting for which $\sigma_{i+2+i'} \in \{0, 2\}$, and $s_+ = s_- = v_+ = v_- = b$.

The possible values of $|C_{m-i-i'-2}(\mathbf{s}) \setminus C_{m-i-i'-2}(\mathbf{v})|$ are summarized in Table A.3.

We now turn our attention to the case $\sigma_{i+i'+2} \in \{0, 2\}$. Again, let $b = s_+ = s_- = v_+ = v_-$. It suffices to consider the following sets:

$$\left\{ \begin{matrix} \{c(\mathbf{s}_1^i), 0^{i'+1}, b\}, \\ \{c(\mathbf{s}_1^i), 0^{i'}, b^2\} \end{matrix} \right\} = \left\{ \begin{matrix} \{c(\mathbf{v}_1^i), 0^{i'}1, b\}, \\ \{c(\mathbf{v}_2^i), b^2, 1^{i'+1}\} \end{matrix} \right\}$$

and

$$\left\{ \begin{matrix} \{c(\mathbf{s}_{m-i+1}^m), 1^{i'+1}, b\}, \\ \{c(\mathbf{s}_{m-i+1}^m), 1^{i'}, b^2\} \end{matrix} \right\} = \left\{ \begin{matrix} \{c(\mathbf{v}_{m-i+1}^m), 01^{i'}, b\}, \\ \{c(\mathbf{v}_{m-i+1}^{m-1}), 0^{i'+1}, b^2\} \end{matrix} \right\}.$$

The possible values of $|C_{m-i-i'-2}(\mathbf{s}) \setminus C_{m-i-i'-2}(\mathbf{v})|$ are summarized in Table A.4.

From the above analysis we can conclude that exactly one of the following two conditions holds:

1. The strings $\mathbf{s}$ and $\mathbf{v}$ satisfy $\mathbf{s}_{i+2}^{i+t+1} = \mathbf{v}_{i+2}^{i+t+1} = \mathbf{0}$ and $\sigma_{i+1}^{i+t+1} = (1, 1, \ldots, 1)$. Their corresponding composition multisets $C_{m-i-1}, C_{m-i-2}, \ldots, C_{m-i-t}$, $C_{m-i-t-1}$ each differ in exactly 2 compositions.

2. The strings $\mathbf{s}$ and $\mathbf{v}$ satisfy $\mathbf{s}_{i+2}^{i+1+i'} = \mathbf{v}_{i+2}^{i+1+i'} = \mathbf{0}$, $\sigma_{i+2}^{i+2+i'} = (1, 1, \ldots, 1)$, and $(s_{i+i'+2}, v_{i+i'+2}) = (1, 0)$, where $t > i' \geqslant 0$. Their corresponding composition multisets $C_{m-i-1}, C_{m-i-2}, \ldots, C_{m-i-i'-1}, C_{m-i-i'-2}$ each differ in exactly 2 compositions.

Figure A.3 illustrates the observations. The longest substring such that $(s_{i+1}, v_{i+1}) = (0,1)$, $(s_{i+2}, v_{i+2}) = (0,0)$, ..., $(s_{i+i'+1}, v_{i+i'+1}) = (0,0)$ and $\sigma_{i+2}^{i+i'+2} = (1,\ldots,1)$ is depicted by a horizontal block in Figure A.3. The bits $s_{i+i'+2}$, $s_{m-i-i'-1}$, $v_{i+i'+2}$, $v_{m-i-i'-1}$ that terminate the $00\ldots0$ (in $\mathbf{s}$) and $10\ldots0$ (in $\mathbf{v}$) substrings in the prefix and the $1\ldots11$ (in $\mathbf{s}$) and $1\ldots10$ (in $\mathbf{v}$) substrings in the suffix are represented by vertical shades in Figure A.3.

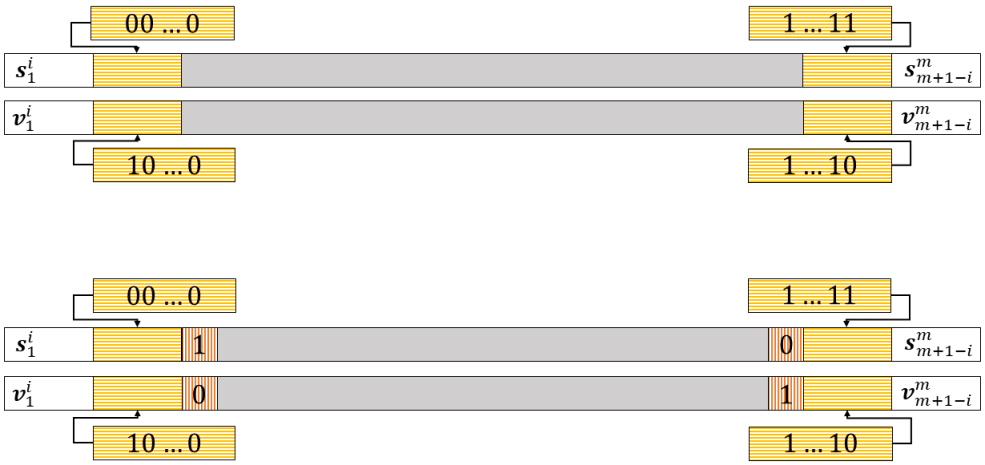


Figure A.3: Illustration of the procedure for determining the set $\mathcal{V}_{\mathbf{s}}$ based on several special cases. For the first case, we have $\mathbf{s}_{i+2}^{i+t+1} = \mathbf{v}_{i+2}^{i+t+1} = \mathbf{0}$ and $\mathbf{s}_{m-i-t}^{m-i-1} = \mathbf{v}_{m-i-t}^{m-i-1} = \mathbf{1}$. For the second case, there exist two identical substrings $\mathbf{s}_{i+2}^{i+1+i'} = \mathbf{v}_{i+2}^{i+1+i'} = \mathbf{0}$ of length $t > i' \geqslant 0$ each and it holds that $(s_{i+i'+2}, v_{i+i'+2}) = (1,0)$.

Assume that the running reconstructions of the distinct strings $\mathbf{s}$ and $\mathbf{v}$ are as depicted in the second pair of blocks in Figure A.3. In the next step, illustrated in Figure A.4, we extend the prefixes and suffixes and identify the conditions under which $|C_{m-i-i'-3}(\mathbf{s}) \setminus C_{m-i-i'-3}(\mathbf{v})|$ is minimized. The results are summarized in Tables A.5 and A.6. In this step, we examine the bits $s_{i+i'+r+3}, v_{i+i'+r+3}$, $s_{m-i-i'-r-2}$ and $v_{m-i-i'-r-2}$.

Assume that

$$\mathbf{s}_{i+i'+3}^{i+i'+r+2} = \mathbf{v}_{i+i'+3}^{i+i'+r+2} = b_1 \ldots b_r,$$

where $r > 0$ and $r = 0$ corresponds to a string of length 0. We have

$$\mathbf{s}_{m-i-i'-r-1}^{m-i-i'-2} = \mathbf{v}_{m-i-i'-r-1}^{m-i-i'-2} = \bar{b}_r \ldots \bar{b}_1,$$

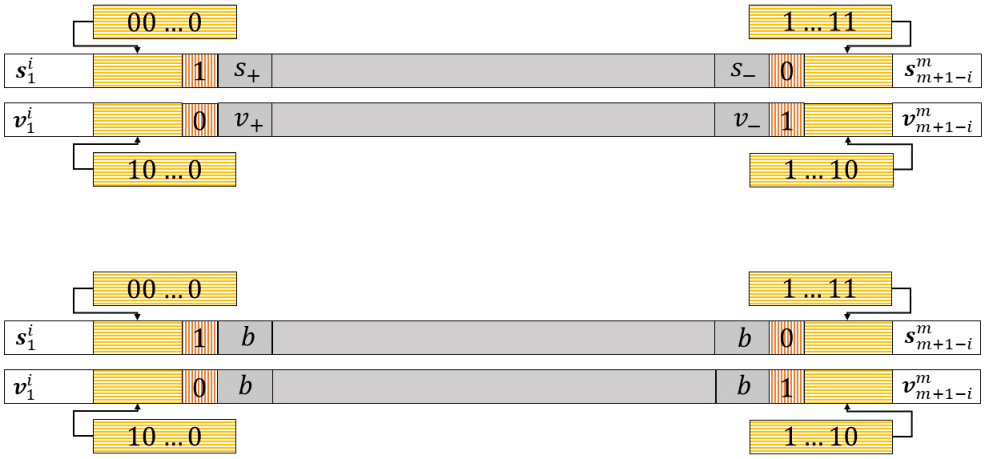


Figure A.4: Illustration of the reconstruction step following the one depicted in Figure A.3. The first pair of strings corresponds to the case $\sigma_+ = 1$, while the second pair of strings corresponds to the case $\sigma_+ \in \{0, 2\}$ and $s_+ = s_- = v_+ = v_- = b$.

where

$$\bar{b}_i = \begin{cases} b_i, \text{ if } \sigma_i \neq 1, \\ 1 - b_i, \text{ if } \sigma_i = 1, \end{cases} \qquad \text{for all } 1 \leqslant i \leqslant r.$$

Such a structure is illustrated in Figure A.5.

For the case $(s_+, v_+) \neq (0, 1)$, it is straightforward to see using arguments similar to the ones previously described that the possible set differences are as listed in Tables A.7 and A.8.

For the case $(s_+, v_+) = (0, 1)$ depicted in Figure A.6, the conditions that ensure that the composition multisets of $\mathbf{s}$ and $\mathbf{v}$ differ by at most 2 introduce the restrictions $b_1, \ldots, b_r = 1 \ldots 1$ and $\bar{b}_1, \ldots, \bar{b}_r = 0 \ldots 0$.

We now extend the description of the set $\mathcal{V}_\mathbf{s}$ illustrated in Figure A.3 as shown in Figure A.7.

Given a pair of distinct strings depicted in the second row of Figure A.3, one of the conditions must hold:

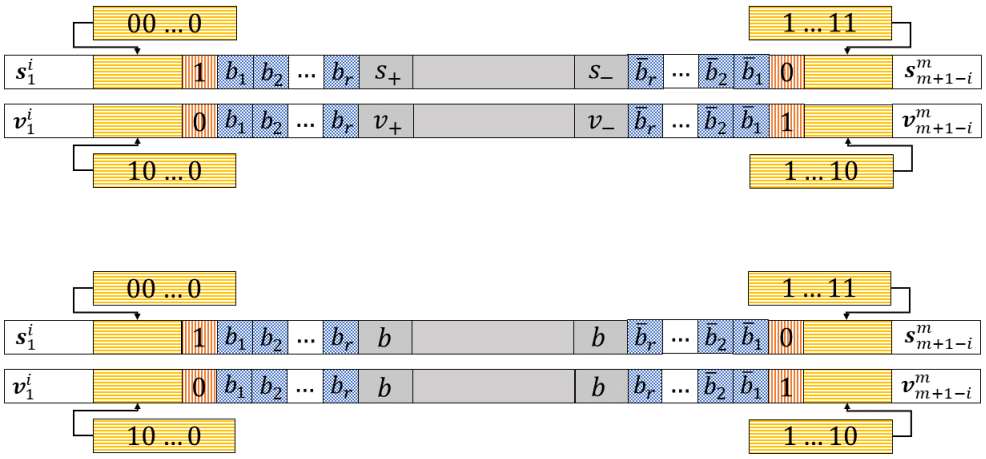


Figure A.5: Two pairs of strings explaining how to extend the partially reconstructed strings illustrated in Figure A.4. The r bits that follow the substring $00\ldots01$ in $\mathbf{s}$ are equal to the corresponding r bits in $\mathbf{v}$. For all $r \geqslant i \geqslant 1$, $\bar{b}_i = b_i$ or $\bar{b}_i = 1 - b_i$. The first pair corresponds to $\sigma_+ = 1$, while the second pair corresponds to $\sigma_+ \in \{0, 2\}$.

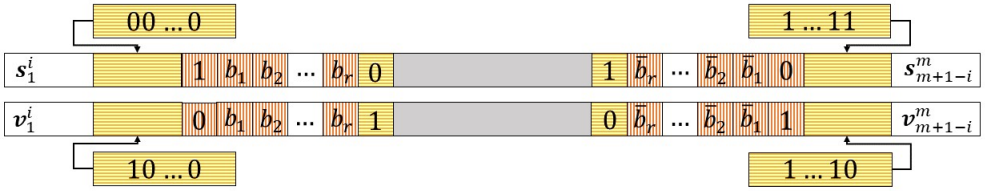


Figure A.6: Conditions on the values of b_i and $\bar{b}_i$ for all i such that $r \geqslant i \geqslant 1$ that ensure that the partially reconstructed strings from the previous step can be compatibly extended when $\sigma_+ = 1$ and $(s_+ = 0, v_+ = 1)$.

- The reconstructed prefix of $\mathbf{s}$ is followed by a substring $b_1 b_2 \ldots b_r$ that is shared by the two strings and is such that the length of the substrings $00\ldots01\,b_1 b_2 \ldots b_r$ (in $\mathbf{s}$) and $10\ldots00\,b_1 b_2 \ldots b_r$ (in $\mathbf{v}$) in the prefixes equals $t+1$. In this case, each pair of composition multisets in $C_{m-i-1}, C_{m-i-2}, \ldots, C_{m-i-t}, C_{m-i-t-1}$ differs in exactly 2 compositions.

- The reconstructed prefix in $\mathbf{s}$ is followed by the substring $1\ldots1\,0$ and the reconstructed prefix in $\mathbf{v}$ is followed by the substring $1\ldots1\,1$. The length of the substrings $00\ldots0\,11\ldots1\,0$ and $10\ldots0\,01\ldots1\,1$ is equal to some $0 < j' < t$. In this case, each pair of composition multi-sets in $C_{m-i-1}, C_{m-i-2}, \ldots, C_{m-i-j'+1}, C_{m-i-j'}$ also differs in exactly 2 compositions.

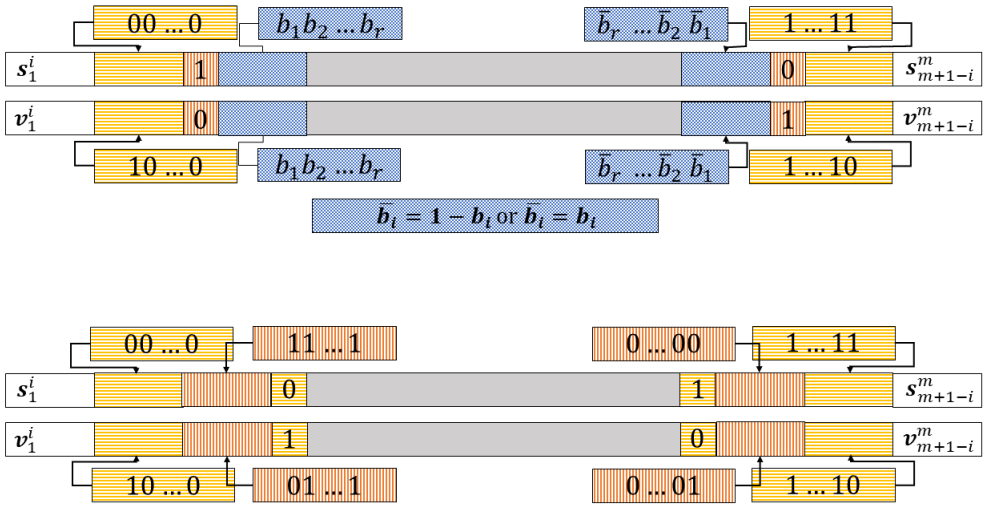


Figure A.7: The procedure for constructing the set $\mathcal{V}_{\mathbf{s}}$ for several special cases of σ values. The first pair depicts the setting in which $\mathbf{s}$ and $\mathbf{v}$ share a substring $b_1 b_2 \ldots b_r$ that follows the substring $00\ldots0\,1$ in $\mathbf{s}$ and $10\ldots0\,0$ in $\mathbf{v}$. The length of the substrings $00\ldots0\,1\,b_1 b_2 \ldots b_r$ and $10\ldots0\,0\,b_1 b_2 \ldots b_r$ in the prefix of $\mathbf{s}$ and $\mathbf{v}$, respectively, equals $t+1$. The second pair depicts a setting in which the substring $11\ldots1\,0$ follows the $00\ldots0$ substring in $\mathbf{s}$ and the substring $01\ldots1\,1$ follows the $10\ldots0$ substring in $\mathbf{s}$. Here, the lengths of the substrings $00\ldots0\,11\ldots1\,0$ in $\mathbf{s}$ and $10\ldots0\,01\ldots1\,0$ in $\mathbf{v}$ may be less than $t+1$.

The bits that were most recently reconstructed in Figure A.7 reestablish the initial problem we started with and the analysis henceforth parallels our previous discussion. The pertinent explanations are summarized in Figures A.8 and A.9.

Combining the results of all the intermediary steps allows us to describe the set $\mathcal{V}_{\mathbf{s}}$ as satisfying one of the two conditions:

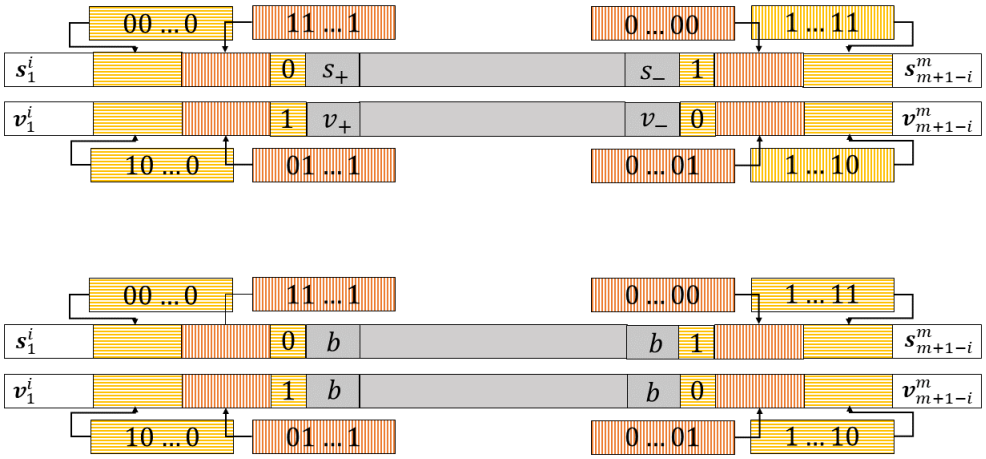


Figure A.8: Two pairs of strings explaining how to extend the partially reconstructed strings illustrated in Figure A.7. The first pair corresponds to the case $\sigma_+ = 1$, while the second pair corresponds to the case $\sigma_+ \in \{0, 2\}$ and $s_+ = s_- = v_+ = v_- = b$.

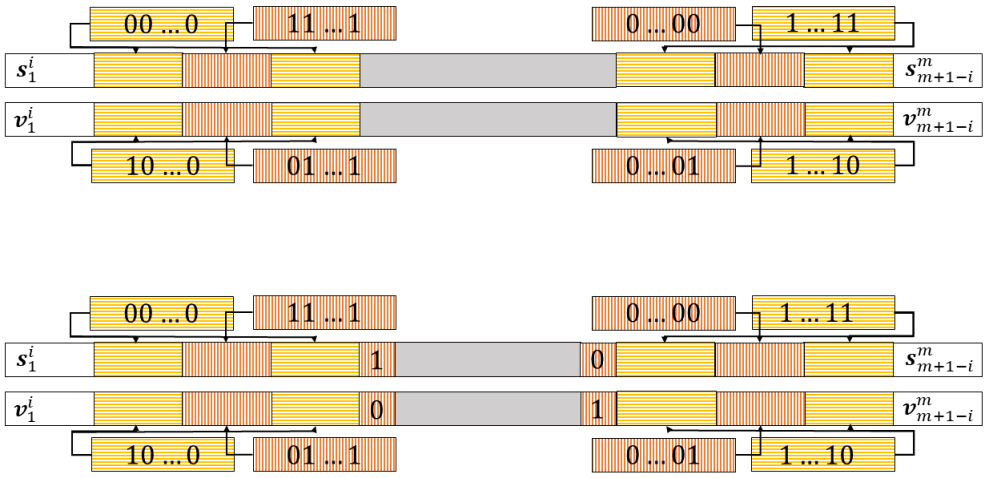


Figure A.9: The partial structure of strings in $\mathcal{V}_{\mathbf{s}}$ as inferred by the previous analysis and the setup shown in Figure A.8.

- The string $\mathbf{s}$ and a string $\mathbf{v} \in \mathcal{V}_\mathbf{s}$ share a prefix-suffix pair that is followed by a certain number of alternating substrings $00\ldots0$ and $11\ldots1$ (in $\mathbf{s}$) and alternating substrings $10\ldots0$ and $01\ldots1$ (in $\mathbf{v}$) in the prefixes. The length of the alternating substrings may vary as described in the analysis, and the substrings are induced by σ values equal to 1. The last of the alternating substrings in the prefixes (equal to either $11\ldots1$ of $01\ldots1$) is followed by a shared substring. The number of bits in the previously described substrings equals $t+1$. The corresponding composition multisets $C_{m-i-1}, C_{m-i-2}, \ldots, C_{m-i-t}, C_{m-i-t-1}$ of the string $\mathbf{s}$ and $\mathbf{v} \in \mathcal{V}_\mathbf{s}$ differ in exactly 2 compositions.

- The string $\mathbf{s}$ and a string $\mathbf{v} \in \mathcal{V}_\mathbf{s}$ share a prefix-suffix pair that is followed by a certain number of alternating substrings $00\ldots0$ and $11\ldots1$ (in $\mathbf{s}$) and alternating substrings $10\ldots0$ and $01\ldots1$ (in $\mathbf{v}$) in the prefixes. The length of the alternating substrings may vary as described in the analysis. The last of the alternating substrings in the prefixes (equal to either $11\ldots1$ or $01\ldots1$) is followed by either the substring $00\ldots0$ (in $\mathbf{s}$) or $10\ldots0$ (in $\mathbf{v}$). The number of bits covered by these cases totals $t+1$ and all underlying values of σ are equal to 1. The corresponding composition multisets $C_{m-i-1}, C_{m-i-2}, \ldots, C_{m-i-t}, C_{m-i-t-1}$ of the string $\mathbf{s}$ and $\mathbf{v} \in \mathcal{V}_\mathbf{s}$ differ in exactly 2 compositions.

Figure A.10 summarizes the structure of the set $\mathcal{V}_\mathbf{s}$ and concludes our proof.

Table A.1: Four different assignments of values for (s_+, v_+) and the resulting set cardinalities $|C_{m-i-2}(\mathbf{s}) \setminus C_{m-i-2}(\mathbf{v})|$.

s_+	0	0	1	1
v_+	0	1	0	1
Set Difference	2	4	2	4

Table A.2: Cardinalities of the set difference $|C_{m-i-2}(\mathbf{s}) \setminus C_{m-i-2}(\mathbf{v})|$ for different choices of b.

b	0	1
Set Difference	3	3

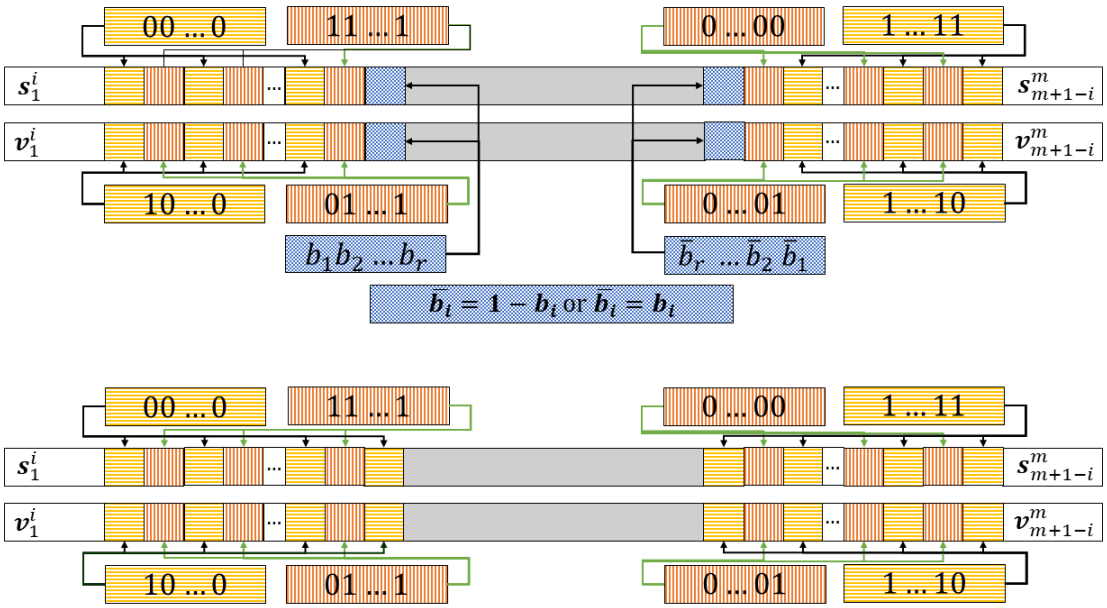


Figure A.10: The structure of the strings $\mathbf{v} \in \mathcal{V}_{\mathbf{s}}$ that are closest to a given string $\mathbf{s}$. The first pair of strings illustrates the case where the substrings $00\ldots0$ and $11\ldots1$ in the prefix of $\mathbf{s}$, and $10\ldots0$ and $01\ldots1$ in the prefix of $\mathbf{v}$ occur in pairs, ending with a shared substring $b_1 b_2 \ldots b_r$. The second pair illustrates the case where the substrings $00\ldots0$ and $11\ldots1$ in the prefix of $\mathbf{s}$, and $10\ldots0$ and $01\ldots1$ in the prefix of $\mathbf{v}$ occur in pairs ending with the substring $00\ldots0$ in the prefix of $\mathbf{s}$ and $10\ldots0$ in the prefix of $\mathbf{v}$. Note that the substrings may not be of equal lengths. With the exception of the final shared substring (i.e., shared substring $b_1 b_2 \ldots b_r$ for the first pair, and the substrings $00\ldots0$ in $\mathbf{s}$ and $10\ldots0$ in $\mathbf{v}$ for the second pair) all strings are of length at least one. The number of bits in the prefix of each string obtained by alternating the above substrings equals $t+1$.

Table A.3: Values of $|C_{m-i-i'-2}(\mathbf{s}) \setminus C_{m-i-i'-2}(\mathbf{v})|$ for the setting where the bits s_{i+1} and v_{i+1} are followed by a run of i' 0s, and where $\sigma_{i+2}^{i+1+i'} = (1, 1, \ldots 1)$ and $\sigma_+ = 1$.

s_+	0	0	1	1
v_+	0	1	0	1
Set Difference	2	4	2	4

Table A.4: The possible values of $|C_{m-i-i'-2}(\mathbf{s}) \setminus C_{m-i-i'-2}(\mathbf{v})|$ for the case that the bits s_{i+1} and v_{i+1} are followed by a run of i' 0s, and such that $\sigma_{i+2}^{i+1+i'} = (1, 1, \ldots 1)$ and $\sigma_+ \in \{0, 2\}$.

b	0	1
Set Difference	3	3

Table A.5: The possible values of $|C_{m-i-i'-3}(\mathbf{s}) \setminus C_{m-i-i'-3}(\mathbf{v})|$ for $\sigma_+ = 1$ corresponding to the four binary assignments for (s_+, v_+) under the setting illustrated in Figure A.4.

s_+	0	0	1	1
v_+	0	1	0	1
Set Difference	2	2	4	2

Table A.6: The possible values of $|C_{m-i-i'-3}(\mathbf{s}) \setminus C_{m-i-i'-3}(\mathbf{v})|$ for different choices of b under the setting illustrated in Figure A.4.

b	0	1
Set Difference	2	2

Table A.7: The possible values of $|C_{m-i-i'-r-2}(\mathbf{s}) \setminus C_{m-i-i'-r-2}(\mathbf{v})|$ for $\sigma_+ = 1$ corresponding to three binary assignments (s_+, v_+) under the setting illustrated in Figure A.5.

s_+	0	1	1
v_+	0	0	1
Set Difference	2	4	2

Table A.8: Cardinalities of the set difference $|C_{m-i-i'-r-2}(\mathbf{s}) \setminus C_{m-i-i'-r-2}(\mathbf{v})|$ for different choices of b under the setting illustrated in Figure A.5.

b	0	1
Set Difference	2	2