UNIVERSITY OF CALIFORNIA

Los Angeles

Statistical Inference Applications in Bioinformatics and Epidemiology

A dissertation submitted in partial satisfaction
of the requirements for the degree
Doctor of Philosophy in Electrical Engineering

by

Sundara Rajan Srinivasavaradhan

2021

ABSTRACT OF THE DISSERTATION

Statistical Inference Applications in Bioinformatics and Epidemiology

by

Sundara Rajan Srinivasavaradhan

Doctor of Philosophy in Electrical Engineering

University of California, Los Angeles, 2021

Professor Christina Panagio Fragouli, Chair

Over the past few decades, engineering and statistical principles have played a significant role in the advancement of biology and medicine. In this dissertation, we study two problems which have applications in bioinformatics and epidemiology, respectively. We look at these problems through the lens of statistical inference, and propose exact and approximate algorithms.

The first line of work is on sequence reconstruction over deletion channels, which lies at the intersection of theoretical computer science, information theory and bioinformatics. Deletion channels have become increasingly relevant in bioinformatics to model impairments in DNA sequencing and DNA data storage. Prior works in theoretical computer science have proposed sequence alignment heuristics for deletion impairments. We take an alternate approach to this problem and propose algorithms that are rooted in statistical principles, such as maximum-likelihood decoding and maximum-aposteriori decoding. Our algorithms outperform the state-of-the-art algorithms in this space. Along the way, we also develop mathematical tools that are of general interest beyond the study of deletion channels.

The second line of work is on group testing which is related to information theory, medical testing and epidemiology. Group testing is a technique that pools together diagnostic samples from multiple individuals to identify infected individuals in a population. Such a technique potentially allows us to use fewer tests than what is required to test everyone individually. Here again, we improve upon the state-of-the-art group testing algorithms by developing a new statistical inference approach to this problem. Moreover, while traditional group testing assumes independent infections, we observe that contagious diseases like COVID-19 are governed by community spread. We show that taking into account the side-information provided by the community structure may lead to significant savings – upto 60% fewer tests compared to traditional test designs in our experiments. We develop new bounds and new approaches to encoding and decoding algorithms that take into account the community structure and integrate group testing into epidemiological modeling.

The dissertation of Sundara Rajan Srinivasavaradhan is approved.

Sriram Sankararaman

Lieven Vandenberghe

Suhas N. Diggavi

Christina Panagio Fragouli, Committee Chair

University of California, Los Angeles

2021

To my parents, brother and late grandmother…

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

As my graduate studies nears its end, it dawns upon me that this journey would not have been possible without the constant support, encouragement and help of a number of people. I am thankful to God in the first place for having given me the strength, a healthy life and a great family.

I would like to express a sincere thank you to my advisor, Prof. Christina Fragouli. Christina has been a continuous source of support and guidance throughout my Ph.D. She is a phenomenal advisor and an incredible person, who always has the best interests of her students at heart. Over the past five years, she helped me grow not only as a researcher, but also as an individual.

I would also like to extend my gratitude to my co-advisor, Prof. Suhas Diggavi. Suhas has been a source of inspiration as an extraordinary academician and an incredible co-advisor. Like Christina, Suhas was also a source of support and guidance throughout my Ph.D. The breadth and depth of his technical knowledge always amazes me.

Almost all of my research works have been team effort and I would like to thank all those with whom I collaborated. In particular, I would like to thank Linqi – who also graduated from ARNI – for guiding me during my earlier stages of Ph.D. I would then like to thank Michelle, who worked with me as an undergraduate researcher and who has not only been a fantastic collaborator but also a dear friend. I would also like to thank Pavlos, who is a terrific researcher and a great collaborator; Pavlos is filled with positivity and is a very helpful person. I would also like to thank my numerous collaborators and mentors at Microsoft Research and at Edwards Lifesciences.

I was fortunate to have been surrounded by a number of incredible individuals at UCLA, many of whom I call my dear friends. In particular my ARNI labmates and friends Gaurav and Yahya have been my go-to individuals for support, help and advice during the major portion of my Ph.D. life and I give my heartfelt thanks to both of them. Special mention also to my friends from LICOS, Deepesh and Dhaivat, and a department friend, Parthe. I would also like to thank a number of other labmates and Ph.D. friends at UCLA – Karmoose, Martina, Navjot, Osama and Mine – for all the technical and non-technical conversations during my stay.

The past few years would not have been as enjoyable without my wonderful friends who have been a constant source of strength and support. My gratitude extends to all these friends, and there are too many for me to individually name. Surely, these friendships will last for a long time to come. My heartfelt thanks goes out to a very special person, SK who is the best friend one could ever hope for. Big shout-out also to my roommates at LA – you people made me call our house a home.

Finally, and most importantly, none of this would have been possible without the love and support of my family and the many sacrifices they made. I will forever be grateful for having them in my life.

VITA

2016	B. Tech + M. Tech. (Electrical Engineering), Indian Institute of Technology Madras, India
2016	Joined Ph.D. program at the ECE Department, UCLA
2016 – 2018	Guru-Krupa fellowship, Department fellowship and Dean’s scholarship, UCLA
2018	Advancement to Ph.D candidacy
2019	Research intern, Edwards Lifesciences, Irvine, CA
2020	Research intern, Microsoft Research, Redmond, WA

SELECTED PUBLICATIONS

Nikolopoulos, P., **Srinivasavaradhan, S. R.**, Fragouli, C., and Diggavi, S., (2021, November) Group testing for community-based infections. IEEE BITS the Information Theory Magazine.

Nikolopoulos, P., **Srinivasavaradhan, S. R.**, Fragouli, C., and Diggavi, S., (2021, November). Community-aware group testing. (*Under submission*).

Srinivasavaradhan, S.R., Nikolopoulos, P., Fragouli, C. and Diggavi, S., (2021, October). Improving Nonadaptive Group Testing via Gradient Descent. (*Under submission*).

Srinivasavaradhan, S.R., Nikolopoulos, P., Fragouli, C. and Diggavi, S., (2021, July). Dynamic group testing to control and monitor disease progression in a population. arXiv preprint arXiv:2106.10765. (*Under submission*).

Srinivasavaradhan, S. R., Nikolopoulos, P., Fragouli, C., and Diggavi, S. (2021, July). An entropy reduction approach to continual testing. In 2021 IEEE International Symposium on Information Theory (ISIT) (pp. 611-616). IEEE.

Nikolopoulos, P., **Srinivasavaradhan, S. R.**, Guo, T., Fragouli, C., and Diggavi, S. (2021, March). Group testing for connected communities. In International Conference on Artificial Intelligence and Statistics (pp. 2341-2349). PMLR.

Nikolopoulos, P., **Srinivasavaradhan, S. R.**, Guo, T., Fragouli, C., and Diggavi, S. (2021, June). Group testing for overlapping communities. In ICC 2021-IEEE International Conference on Communications (pp. 1-7). IEEE.

Srinivasavaradhan, S.R., Du, M., Diggavi, S.N. and Fragouli, C., 2020. Algorithms for reconstruction over single and multiple deletion channels. IEEE Transactions on Information Theory, 67(6), pp.3389-3410.

Srinivasavaradhan, S.R., Du, M., Diggavi, S. and Fragouli, C., 2019, July. Symbolwise MAP for multiple deletion channels. In 2019 IEEE International Symposium on Information Theory (ISIT) (pp. 181-185). IEEE.

Srinivasavaradhan, S.R., Du, M., Diggavi, S. and Fragouli, C., 2018, June. On maximum likelihood reconstruction over multiple deletion channels. In 2018 IEEE International Symposium on Information Theory (ISIT) (pp. 436-440). IEEE.

CHAPTER 1

Introduction

In the past few decades, statistical methods have increasingly gained popularity and proven their use in the understanding of biological systems. Rapid advancements on the front of computational capacity and techniques have enabled the transformation of many areas such as bioinformatics and epidemiology. In fact, the relationship between statistics and biology can be traced further back to the 19th century, when patterns in data were used to identify the source of a cholera outbreak [Sno55]; this study is considered to be the beginning of modern epidemiology. Indeed, the father of modern statistics, Ronald Fisher, himself extensively employed statistics to advance the field of genetics.

Biological models and ideas have also had a tremendous impact in computer science and engineering – genetic algorithms and neural networks are two such ideas inspired by biological systems. Very recently, the idea of storing digital information in DNA sequences has received considerable attention owing to recent developments in biotechnology [CNS19, OAC18]. Progress in the field of DNA data storage is imperative, given the massive amount of data that is produced every single day; it is hypothesized that a coffee mug full of DNA could theoretically store all of the world's data [Tra21].

Motivated by the importance and implications of such inter-disciplinary research, in this dissertation, we study two mathematical problems closely related to bioinformatics and epidemiology, namely *Trace Reconstruction* and *Group Testing.* Trace reconstruction lies at the intersection of theoretical computer science, information theory and bioinformatics and is a problem of interest to all three areas. Group testing, on the other hand, is at the intersection of information theory, compressed sensing and epidemiology. In the following section, we motivate these problems and discuss an overview of the literature. Following this discussion, we then outline the key contributions of this dissertation.

1.1 Background and Motivation

1.1.1 Trace reconstruction

Trace reconstruction studies the problem of reconstructing an unknown sequence by combining information from multiple deleted versions (also called traces) of itself (see Figure 1.1). The problem was first introduced in the theoretical computer science community [BKK04] but is very closely related to the construct of deletion channels, which is a topic of interest in information theory (see [Mit09]). In fact, trace reconstruction is amongst one of the many questions that concern deletion channels. Trace reconstruction and deletion channels have become increasingly relevant in bioinformatics through their connection to DNA sequencing and DNA data storage.

In *de-novo* DNA sequencing, an unknown DNA could be sequenced multiple times, yielding several "reads" of it, but each of these reads suffer from errors (see for example [MDK17]). The nature of these errors also vary widely, depending on the exact DNA sequencing technology used. Exactly modeling these error profiles is tedious, impractical, and also does not provide a future-proof approach to the problem, as DNA sequencing technologies are evolving at a rapid pace. To this end, a zeroth-order approximation to model such errors is used, in the form of insertions, deletions and substitutions. Amongst these, deletions are especially challenging to deal with.

DNA sequencing is an important component of DNA data storage, where it is used to retrieve the stored data. DNA data storage is an exciting area as it promises extremely high storage densities and long-term stability [CGK12]. Trace reconstruction ideas have already been used for data retrieval in the first fully automated prototype of a DNA storage

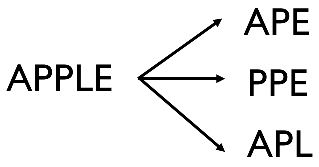


Figure 1.1: Trace reconstruction: given deleted versions (traces) of an input sequence reconstruct the unknown input sequence.

system [OAC18, GYA18, YR20].

The deletion channel by itself, is known to be notoriously difficult to analyse. The capacity of a single deletion channel is still unknown ([DMP, DG06, DG01]); as are optimal coding schemes. Statistical estimation over deletion channels (which is the goal in trace reconstruction) is a difficult problem to analyze due its highly combinatorial nature. To the best of our knowledge, as yet there are no efficient estimation algorithms over deletion channels with statistical guarantees.

A number of works in theoretical computer science have studied the problem of trace reconstruction [HMP08, PZ17, DOS, HPP18, NP17, HL18, Cha19]. The main question these works address is how many deleted reads are needed for perfect reconstruction of an input sequence? They show that the required number of traces (or reads) through independent deletion channels grows with the input length, either exponentially in the worst case or sub-polynomially in the average case.

In practice, as is the case in DNA data storage, one may only be able to observe a few of these reads and as a result, perfect reconstruction may turn out to be infeasible. A very extreme example is if one observes only one trace – it is impossible to reconstruct the original sequence even if a single symbol was deleted. Therefore, a more pragmatic approach to trace reconstruction is to study the following question – *given a fixed number of traces, what is the best estimate of the input sequence?* This question has a more statistical inference flavor and Part I of this dissertation is devoted to answering this question.

1.1.2 Group testing

Our recent experience with COVID-19 has revealed the key role of epidemiological models and testing in the fight against pandemics (e.g. [TRL20, BBL20]). For any new disease or variant of the existing ones, we will always need the ability to expeditiously deploy strategies that allow efficient testing of populations and empower targeted interventions (ideally at an individual level).

Group testing is a technique that can identify the infected individuals in a population with fewer tests than the ones needed to test everyone individually. Instead of testing each person individually, group testing applies *pooled tests* on the top of *groups* of diagnostic samples from multiple individuals. When pooling together these samples, particular care is taken, so that the testing material is not diluted during the mixing process and the sensitivity/specificity of the tests used is not altered significantly. The key insight is that if infections are sparse, then many group-test outcomes are likely to be negative, and therefore all individuals included in them can be deemed healthy. However, if a group test is positive, then one cannot directly tell which individual(s) included in the test are infected; additional testing or careful decoding of other test results is therefore necessary. Accordingly, group testing offers significant benefits for sparse regimes of infection. On the other hand, if infections follow a linear or mildly sublinear regime, then individual testing has been found to be optimal [Ald19, BPS20].

Group testing has a rich history dating back to R. Dorfman in 1943, who first introduced the concept during World War II, when the U.S. military sought to identify soldiers infected with syphilis, but tests were expensive [Dor43]. Then on, a number of variations and setups have been examined [AJS19, DH93, MA16].

Simply stated, the typical group-testing setup assumes a population of N individuals out of which a few are infected, and the goal is to design testing strategies and corresponding decoding algorithms to identify the infections from the test results. Most works revolve around proposing a particular hand-crafted test design (e.g. random Bernoulli design) coupled with a decoding strategy (e.g. Definite Defectives, Definite Non-Defectives), and guarantees are provided on the number of tests required to achieve a vanishing probability of error. Additionally, order-optimality results have been proved for the asymptotic regime, where the population size tends to infinity. For example, in a population of $N \to \infty$ members, if very few people (say $k < N^{1-\Omega(1)}$) are infected, one can identify them with as low as $\mathcal{O}(k \log \frac{N}{k})$ pool tests performed in multiple adaptive stages or $\mathcal{O}(k \log N)$ pool tests performed in a single, non-adaptive stage [AJS19, BPS20] [1].

Interestingly, group testing is being re-invented nowadays in the context of the pandemic [GG20, Bro20, Ell20, Ver20, Gho20, KLL20], and several countries (including India, Germany, US, and China) have already deployed preliminary group-testing strategies [Mal20, FDA20]. Also, companies and schools use pool tests to regularly monitor parts of their population, and then do individual tests once a pool test comes positive (which is similar to Dorfman's approach).

The assumptions traditionally made in group testing have a few drawbacks, in particular, they assume a static, independent model of infection. However, viral diseases among humans have an important characteristic: infections are governed by community spread, and are therefore correlated. As a use case, consider an apartment building consisting of families that have practiced social distancing; clearly, there is a strong correlation on whether members of the same family are infected or not. Moreover, infections are also continually evolving and proliferating through the population, even as tests are being administered. As a result, group testing should also be studied as a "dynamic" problem, where one needs to design testing strategies for each day. The static and dynamic cases are closely interrelated. For instance, the static case can be viewed as identifying the "initial state" in the case of dynamic evolution.

[1] $\mathcal{O}$ and Ω notations denote, respectively, the asymptotic upper and lower bounds, as N tends to infinity.

Given the above discussion, in this dissertation, we investigate the following three questions related to group testing: 1. *For the static and independent infection model, how can we design optimal testing strategies under constraints on the number of tests?* 2. *Does the knowledge of a community structure help in the design of testing strategies and if so, by how much?* 3. *How best can we use group testing to test a population over a time horizon, when the number of tests are limited, test results are delayed, and when the infections are continually proliferating?* Part II of this dissertation is devoted to studying these questions. We next outline our contributions in trace reconstruction and group testing.

1.2 Contributions

As motivated earlier in this chapter, we look at the trace reconstruction problem through the lens of statistical inference. In particular, we ask the following two questions – given a fixed number of traces, what is the most likely realization of the input sequence (sequencewise maximum-likelihood decoding) and what is the most likely value taken by each symbol of the input sequence (symbolwise maximum-aposteriori decoding)? The first question is an important component to further our understanding of deletion channel and its capacity (see [Mit09]), while the second question is directly relevant in the context of DNA data storage (see [RMR17, OAC18]). With this in mind, the high-level contributions of this dissertation towards trace reconstruction are as follows:

Contribution 1: We introduce mathematical tools and constructs to systematically visualize sequence alignments and analyze single and multiple deletion channels. Along the way, we make connections to dynamic programming on graphs, non-commutative algebra and optimization theory. These tools are of general interest beyond the specific problem considered in this work.

Contribution 2: We establish an equivalence between finding the optimal sequencewise maximum-likelihood decoder and a continuous optimization problem. In fact, we establish this equivalence for a more general family of channels, which admit the deletion channel as a special case. We then leverage on this equivalence to derive gradient-based heuristics for trace reconstruction.

Contribution 3: We derive an exact algorithm for symbolwise maximum-aposteriori decoding over deletion channels. To the best of our knowledge, this is the first reconstruction algorithm over deletion channels which is optimal, in the sense that it provably minimizes the expected *Hamming distance* between the true sequence and its estimate.

From the discussion in the previous section, we recall that traditional group testing assumes a static and independent model of infection, while in reality, infections are correlated due to a community structure, and are continually evolving during the testing period. Our contributions on the front of group testing are outlined as follows:

Contribution 4: We improve upon the state-of-the-art test designs in group testing for the static, independent infection model. We study this problem under a resource-constrained setting where the number of available tests are limited. To do this, we again look at the problem through the lens of statistical inference, where we formulate the search for test designs as an optimization problem that aims to minimize the Hamming error rate, similar to our approach in Contribution 2. Consequently, we propose gradient-based heuristics to search for good test designs.

Contribution 5: We show that taking into account the side-information provided by the community structure may lead to significant savings – upto 60% fewer tests compared to traditional test designs in the static infection model. We argue that leveraging the community structure can also enlarge the regime where group testing offers significant benefits over individual testing. To do this, we adapt existing test designs to account for the community structure and also modify existing decoders to incorporate the information provided by the community structure.

Contribution 6: We integrate testing into epidemiological modeling, and study how to dynamically test a population given that test results are delayed by a day, and the disease continually spreads in the population. We show that the dynamic testing problem reduces to the static case given enough testing resources each day. As a result, existing static group testing algorithms can be reused. We derive a new lower bound for static group testing which proves the order-optimality of some known group test designs under certain assumptions; this in turn proves the order-optimality of these test designs for the dynamic case as well.

1.3 Outline of Dissertation

This dissertation is organized in two parts. In **Part I**, we study deletion channels focused on the problem of trace reconstruction, while **Part II** investigates various facets of group testing, including its connection to epidemiology. We further detail Parts I and II below.

1.3.1 Part I: Deletion Channels and Trace Reconstruction

Part I is further subdivided into two chapters (Chapters 2, 3). In **Chapter 2**, we first introduce mathematical tools that aid in visualization and analysis of deletion channels. We then use these tools to derive an exact algorithm that calculates symbolwise posterior marginals. Further, we also formulate the maximum likelihood decoder as a discrete optimization problem and show its equivalence to a continuous optimization problem – we propose a gradient descent heuristic to solve this problem. Finally, we illustrate numerics to demonstrate the improvements our methods provide over existing trace reconstruction algorithms.

Chapter 3 generalizes the equivalence idea introduced in Chapter 2 to fit more general channel models. Further, we use this perspective to propose other gradient-based heuristics for trace reconstruction which improve upon the methods in Chapter 2 in certain situations.

1.3.2 Part II: Group Testing and Epidemiology

Part II is further subdivided into three chapters (Chapters 4, 5 and 6). In **Chapter 4** we study group testing for a static infection model with non-uniform priors. Unlike prior works, we take a practical approach to this problem: we fix the decoder and the number of tests, and we ask what is the best test design one could use? We examine this problem for the definite non-defectives (DND) decoder and formulate it as a (non-convex) optimization problem, where the objective function is the expected number of errors for a particular design. We propose an approximate solution via gradient descent, which we further optimize with informed initialization.

In **Chapter 5**, we study group testing in the presence of correlations, where the correlation is induced by a community structure. Our goal in this chapter is to indicate what are potential benefits, and describe what are some first ways group testing can leverage community knowledge, both in terms of designing tests as well as incorporating this knowledge on the side of the decoder. In **Chapter 6**, we consider the dynamics of disease spread and study how static group testing can be used in such a setting. We first prove our new lower bound for the static case and then show the order-optimality of existing group testing algorithms. We then make precise the conditions under which the dynamic testing problem reduces to the static case, enabling the use of static group testing algorithms.

Parts of this dissertation are presented in [SDDa, SDDb, SDD20, SDF20, NRG21, NSG21, SNF21b, SNF21a, NSF21].

Part I

Deletion Channels and Trace Reconstruction

CHAPTER 2

Sequencewise and symbolwise inference over multiple deletion channels

Summary: *In this chapter, we study the trace reconstruction problem over multiple deletion channels through the lens of statistical inference. We first introduce mathematical tools that aid in visualization and analysis of deletion channels. We then use these tools to derive an exact algorithm that calculates symbolwise posterior marginals. Further, we also formulate the maximum likelihood decoder as a discrete optimization problem and show its equivalence to a continuous optimization problem – we propose a gradient descent heuristic to solve this problem. Finally, we illustrate numerics to demonstrate the improvements our methods provide over existing trace reconstruction algorithms.*

Sequence reconstruction over deletion channels, both with and without a codebook, has received considerable attention in the information theory as well as in the theoretical computer science literature. From an information theory perspective, reconstruction over the deletion channel, or more specifically a maximum-likelihood (ML) argument for the deletion channel, would give further insight on the capacity of the deletion channel, a long-standing open problem (see [Mit09]). To quote [Mit09] – "at the very least, progress in this direction would likely surpass previous results on the capacity of the deletion channels". Yet, there are no results on reconstruction over a deletion channel with statistical guarantees. In this chapter, we take steps in this direction.

In this space, the problem of *trace reconstruction*, as introduced in [BKK04], has also received renewed interest in the past few years (see [HMP08,PZ17], [DOS], [HPP18], [NP17],

[HL18], [Cha19]). The problem of trace reconstruction can be stated simply as follows: consider a sequence X which is simultaneously passed through t independent deletion channels to yield t output subsequences (also called *traces*) of X (see Fig. 2.1). How many such traces are needed to reconstruct X perfectly? A variety of upper and lower bounds for this problem have been proposed, both for worst case and average case reconstruction. Our problem formulation is complementary to this, as we discuss next.

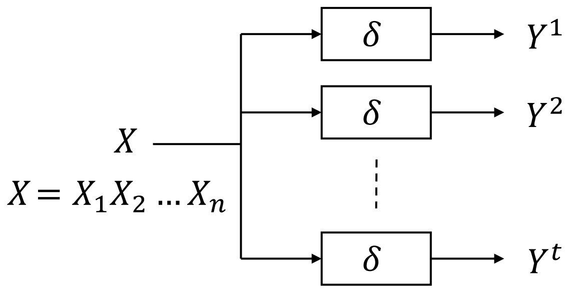


Figure 2.1: The t-trace deletion channel model: the sequence X is passed through t independent deletion channels to yield t *traces*. We aim to estimate X from the Y^is.

Problem formulation. Given an input sequence of length n (known apriori), the independently and identically distributed (i.i.d.) deletion channel deletes each input symbol indepedently with probability δ, producing at its output a subsequence of the input sequence. Consider a sequence X passed through t (t is fixed) such deletion channels as shown in Fig. 2.1. We call this the t-trace deletion channel model. We ask four main questions:

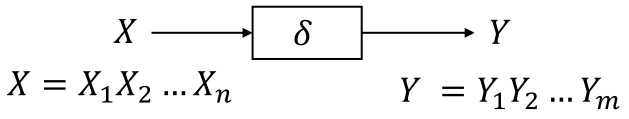


Figure 2.2: The single-trace deletion channel model.

1. **Sequencewise maximum-likelihood with one trace:** For $t = 1$ (also called *single-trace deletion channel*, see Fig. 2.2), what is the maximum-likelihood estimate of X having

observed $Y = y$, i.e., a solution to $\underset{x \in \{0,1\}^n}{\arg\max} \ \Pr(Y = y|X = x)$.

2. **Sequencewise maximum-likelihood with multiple traces:** For a fixed t, with $t > 1$, what is the maximum-likelihood estimate of X having observed $Y^1 = y^1, Y^2 = y^2, ..., Y^t = y^t$, i.e.,
$$\underset{x \in \{0,1\}^n}{\arg\max} \ \Pr(Y^1 = y^1, Y^2 = y^2, ..., Y^t = y^t|X = x).$$

3. **Symbolwise MAP with one trace:** For $t = 1$ and $X_i \sim$ ind. Ber(p_i) in Fig. 2.2, what are the posterior distributions of X_i given the trace $Y = y$, i.e., compute $\Pr(X_i = \alpha|Y = y)$.

4. **Symbolwise MAP with multiple traces:** For a fixed t, with $t > 1$ and $X_i \sim$ i.i.d. Ber(0.5) in Fig. 2.1, what are the posterior distributions of X_i given all traces $Y^1 = y^1, Y^2 = y^2, ..., Y^t = y^t$, i.e., compute $\Pr(X_i = \alpha|Y^1 = y^1, Y^2 = y^2, ..., Y^t = y^t)$.

We make a few notes.

- For a channel with memory such as the deletion channel, the symbolwise MAP/ML estimate and sequencewise MAP/ML estimate are not equivalent. For example, consider $t = 1$, $n = 6$ in Fig. 2.2 and say we observe the trace $Y = 1010$. The symbolwise MAP estimate with uniform priors for this case can be computed to be $\hat{X}_{smap} = 100110$ whereas the sequencewise ML estimate is $\hat{X}_{ml} = 101010$.

- An answer to 3) above doesn't lead to a natural solution for 4) which is also due to deletion channels possessing memory. In particular, for a memoryless channel, we have $Y_i^j - X_i - Y_i^k$ and hence $\Pr(X_i = \alpha|Y^j, Y^k) \propto \Pr(Y_i^j, Y_i^k|X_i = \alpha) = \Pr(Y_i^j|X_i = \alpha) \Pr(Y_i^k|X_i = \alpha) \propto \Pr(X_i = \alpha|Y^j) \Pr(X_i = \alpha|Y^k)$; so one could first obtain the posterior probabilities from each independent observation and combine them after. However, this is not the case for deletion channels since the markov chain $Y_i^j - X_i - Y_i^k$ no longer holds. As a result, one first needs to "align" all the observations in order to compute the likelihoods.

- Solving 2) and 4) naturally leads to two different algorithms for average-case trace reconstruction – one that selects the most likely sequence X and the other that selects the most likely value for each symbol X_i. However, the problem formulations in 3) and 4) ask a question complementary to that of trace reconstruction: given a fixed (possibly a few) number of traces, what is our "best" guess of X? The two problems 2) and 4) have different quantification of the word "best". Unlike trace reconstruction, we are not concerned with perfect reconstruction (since perfect reconstruction may not be possible with just a few traces). We also note that error rate guarantees for our algorithms (not a part of this dissertation) would naturally lead to upper bounds for trace reconstruction.

- The challenges associated with solving 1) and 2) and solving 3) and 4) are very different. On the one hand, solving 1) and 2) amounts to discovering alternate, equivalent or approximate formulations for the seemingly difficult discrete optimization problems. On the other hand, the challenge with 3) and 4) involves the design of efficient algorithms that are capable of exactly computing/approximating the symbolwise posterior probabilities, for which "closed form" expressions can be derived.

Contributions. Our main contributions are as follows.

- We introduce mathematical tools and constructs to visualize and analyze single-trace and t-trace deletion error events (see Section 2.1).

- For the single-trace deletion channel, we establish an equivalence between finding the optimal ML decoder and a continuous optimization problem we introduce (see Section 2.2). This equivalence allows for the use of existing techniques for continuous optimization to be employed for a seemingly difficult discrete optimization problem. This continuous optimization problem also turns out to be a signomial optimization. Furthermore we also provide a polynomial time trace reconstruction heuristic with multiple traces that exploits this formulation.

- In Section 2.3, we prove the following:

  Theorem 2.1. *For the single-trace deletion channel model with priors $X_i \sim$ ind. Ber(p_i) and observed trace $Y = y$, the symbolwise posterior probabilities* $\Pr(X_i = 1|Y = y)\ \forall\ i$ *can be computed in $O(n^2)$ time complexity.*

- In Section 2.4, we prove the following:

  Theorem 2.2. *For the t-trace deletion channel model with priors $X_i \sim$ i.i.d. Ber(0.5) and observed traces $Y^1 = y^1, ..., Y^t = y^t$, the symbolwise posterior probabilities* $\Pr(X_i = 1|Y^1 = y^1, ..., Y^t = y^t)\ \forall\ i$ *can be computed in $O(2^t n^{t+2})$ time complexity.*

Tools and techniques. In terms of theoretical tools, the series of books by Lothaire ([Lot97, Lot02, Lot05]) extensively use algebraic tools for problems in the combinatorics of sequences (or *words*), and this chapter borrows some notation and leverages on a few of their results.

Biological motivation. Trace reconstruction in itself was motivated, in part, by problems in DNA sequence reconstruction. One such problem was to infer the DNA sequence of a common ancestor from the samples of its descendants. Our problem definition, that considers a fixed value of t, would fit naturally in a scenario with a fixed number of descendants where perfect reconstruction may not be possible. Our motivation for considering this problem also comes from a recent DNA sequencing technology called *nanopore sequencing*. The t-trace deletion channel model is a simplistic model to approximately capture the process of a DNA sequence passed through a nanopore sequencer[1].

More related work. This chapter falls under the general umbrella of sequence reconstruction over deletion channels (also see Levenshtein's work [Lev01]), where we offer, to the best of our knowledge, the first non-trivial results on maximum likelihood and maximum aposteriori estimates for the single and multiple deletion channel. As mentioned earlier, the complementary problem of trace reconstruction falls closest to this work.

The deletion channel by itself is known to be notoriously difficult to analyse. As stated earlier, the capacity of a single deletion channel is still unknown ([DMP, DG06, DG01]); as are optimal coding schemes. Prior works have looked at the design of codes for deletion channels ([Rat05, RM00, TTV17]); these works consider use of a codebook (we do not). Statistical estimation over deletion channels is a difficult problem to analyze due its highly combinatorial nature. To the best of our knowledge, as yet there are no efficient estimation algorithms over deletion channels with statistical guarantees.

Very recently, a variant of the trace reconstruction problem called *coded trace reconstruction* has been proposed, motivated by portable DNA-based data storage systems using DNA nanopores (see [AVD19, CGM19, BLS19, SGP21]) and we believe that the ideas in this chapter may prove useful in such a setting.

There are other works on sequence assembly (see for example, [LD09], [SKC16]), where multiple short reads (from different segments of a sequence) are used to reconstruct the bigger sequence. The work in this chapter differs from sequence assembly since we are interested in inferring the entire length sequence and not just small segments of it (which are then "stitched" together in sequence assembly).

Chapter organization. Section 2.1 introduces our notation and visualization tools for the

[1]As seen in [MDK17] there are more complicated effects of the nanopore reader not captured in this simple representation.

single and t-trace channel error events; Section 2.2 provides a result concerning questions 1) and 2) wherein we prove the equivalence of ML decoding in question 1) to solving a continuous optimization problem; Section 2.3 answers question 3) for the single-trace channel; Section 2.4) answers question 4) for the t-deletion channel; Section 2.5 gives numerical evaluations; and Section 2.6 concludes the chapter and discusses open questions.

2.1 Notation and Tools

Basic notation: We borrow some notation from [Lot97] which deals with non-commutative algebra; we restate them here for convenience. Calligraphic letters refer to sets, capitalized letters correspond to random variables and bold letters are used for functions. Let $\mathcal{A}$ be the set of all symbols. Throughout this chapter, we will focus on the case where $\mathcal{A} = \{0, 1\}$, though our methods extend to arbitrarily large sets of finite size. Define $\mathcal{A}^n$ to be the set of all n-length sequences and $\mathcal{A}^*$ to be the set of all finite length sequences with symbols in $\mathcal{A}$. For a sequence f, $|f|$ denotes the length of f.

For integers i, j, we define $[i : j] \triangleq \{i, i+1, ..., j\}$ if $j \geq i$ and $[i : j] \triangleq \varnothing$ otherwise. We also define $[i] \triangleq [1 : i]$.

For a vector or sequence $x = (x_1, x_2, ..., x_{i-1}, x_i, x_{i+1}, ..., x_n)$, define

$$x^{(i \to s)} \triangleq (x_1, x_2, ..., x_{i-1}, s, x_{i+1}, ..., x_n),$$

where the i^{th} coordinate of x is replaced by symbol s.

Binomial coefficient (section 6.3 in [Lot97]): Given sequences f and g in $\mathcal{A}^*$, the number of subsequence patterns of f that are equal to g is called the *binomial coefficient* of g in f and is denoted by $\binom{f}{g}$. For example, $\binom{'apple'}{'ape'} = 2$ since $'ape'$ can be obtained from two (overlapping) subsequences of $'apple'$. This quantity has also been referred to as the *embedding number* by another line of work [ERW08]. For two sequences of lengths n and m, the binomial coefficient can be computed using a dynamic programming approach in $O(nm)$ (see [ERW08] or Proposition 6.3.2 in [Lot97]). When the alphabet $\mathcal{A}$ is of cardinality 1, $\binom{f}{g} = \binom{|f|}{|g|}$, the classical binomial coefficient with their respective lengths as the parameters. This definition hence could be thought of as a generalization of the classical binomial coefficients. We will denote by e the sequence of length 0, and define $\binom{f}{e} \triangleq 1 \ \forall \ f \ \in \ \mathcal{A}^*$. We also define the classical binomial coefficient $\binom{a}{b} \triangleq 0$, whenever $b > a$ or $b < 0$ for ease of use.

The binomial coefficient forms the backbone for the probabilistic analysis of deletion channels since the input-output relation for a deletion channel (with deletion probability δ, input X and output Y) can be expressed as

$$\Pr(Y = y|X = x) = \binom{x}{y} \delta^{|x|-|y|}(1-\delta)^{|y|}. \tag{2.1}$$

The proof is straightforward – the number of distinct error events that give rise to y from x is exactly the number of subsequences of x which are equal to y. Each of these error events has a probability $\delta^{|x|-|y|}(1-\delta)^{|y|}$, wherein the exponent of δ corresponds to the deleted symbols and the exponent of $1-\delta$ to the undeleted symbols.

Maximum Likelihood (ML) estimate: Given the definition of the binomial coefficient, the maximum-likelihood (ML) estimate over a deletion channel with observed output $Y = y$ can be cast in the following form:

$$\underset{x \in \{0,1\}^n}{\arg\max} \binom{x}{y}. \tag{2.2}$$

In the case of multiple deletion channels with observed traces $Y^1 = y^1, ..., Y^t = y^t$, the ML formulation is similar:

$$\underset{x \in \{0,1\}^n}{\arg\max} \prod_{j=1}^{t} \binom{x}{y^j}. \tag{2.3}$$

As yet, there is no known efficient way to come up with a solution for either of the above two formulations (see [Mit09]).

Relaxed binomial coefficient. We now introduce the function $\mathbf{F}(\cdot)$ which can be thought of as a real-valued extension of the binomial coefficient. This function is used in sections 2.2 and 2.3.

An intuitive definition is as follows: Consider a random vector $Z \in \{0,1\}^n$ such that $Z_i \sim$ ind. Ber(p_i), and let p be the vector of probabilities of length n. Then $\mathbf{F}(p,v) = \mathbb{E}_{Z\sim p} \binom{Z}{v}$, i.e., $\mathbf{F}(p,v)$ is the expected number of times v appears as a subsequence of Z. If $p \in \{0,1\}^n$, then $Z = p$ with probability 1 and $\mathbf{F}(p,v) = \binom{p}{v}$. More precisely, $\mathbf{F}(\cdot)$ is defined as:

Definition 2.1.

$$\mathbf{F} : [0,1]^n \times \{0,1\}^m \to \mathbb{R},$$

$$\mathbf{F}(p,v) \triangleq \begin{cases} \sum\limits_{\substack{\mathcal{S}\mid\mathcal{S}\subseteq[n],\\ |\mathcal{S}|=m}} \quad \prod\limits_{i=1}^{m} p_{\mathcal{S}_i}^{v_i}(1-p_{\mathcal{S}_i})^{1-v_i} & 1 \le m \le n \\ 1 & 0 = m \le n \\ 0 & \text{else.} \end{cases}$$

Though at first sight $\mathbf{F}(p,v)$ sums over an exponential number of subsets, a dynamic programming approach can be used to compute it in $O(nm)$ time complexity (see Appendix 2.7.5). Note that this is the same complexity as computing the binomial coefficient.

Decomposition of the t-trace deletion channel: The following definitions and ideas are relevant to the results pertaining to multiple traces. We first state a result that aids in thinking about error events in multiple deletion channels.

The events occurring in the t-deletion channel model can be categorized into two groups:

1. an input symbol is deleted in *all* the t-traces,

2. an input symbol is reflected in at least one of the traces.

The error events of the first kind are in some sense "not correctable" or even "detectable" in any situation since it is impossible to tell with absolute certainty what and where the deleted symbol could have been (although the probabilities need not be uniform). The events of the second kind, however, can be detected and corrected in some situations. This thought process gives rise to a natural decomposition of the t-deletion channel model into a cascade of two channels: the first one being a deletion channel which captures error events of the first kind and the second one is what we call the *remnant channel* which captures events of the second kind (see Fig. 2.3). More precisely, we define the remnant channel as follows:

Definition 2.2. *Remnant channel:* an input symbol to the remnant channel is reflected in any $k > 0$ uniformly random traces and deleted in the rest with a probability $\binom{t}{k}\frac{\delta^{t-k}(1-\delta)^k}{1-\delta^t}$. Thus, the probability of an input symbol reflected in a *fixed* set of $k > 0$ traces is equal to $\frac{\delta^{t-k}(1-\delta)^k}{1-\delta^t}$.

Note that probability of the union of all possible events here is $\sum_{k=1}^{t}\binom{t}{k}\frac{\delta^{t-k}(1-\delta)^k}{1-\delta^t} = 1$, validating our definition.

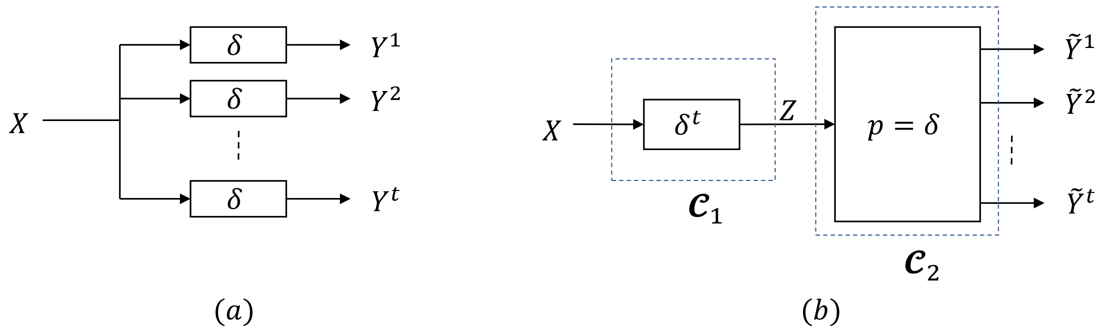


Figure 2.3: A channel equivalence result: the t-trace deletion channel model in (a) is probabilistically equivalent to the the cascade of a deletion channel with the *remnant channel* ($\mathcal{C}_2$) in (b).

Theorem 2.3. *The t-deletion channel model and the cascade of the deletion channel with remnant channel shown in Fig. 2.3 are probabilistically equivalent, i.e.,*

$$\Pr(Y^1 = y^1, Y^2 = y^2, ..., Y^t = y^t | X = x) = \Pr(\tilde{Y}^1 = y^1, \tilde{Y}^2 = y^2, ..., \tilde{Y}^t = y^t | X = x).$$

A rigorous proof of this theorem for arbitrary length sequences can be found in Appendix 2.7.1. A similar, though not equivalent, decomposition has been exploited in [HM14] albeit for the purpose of characterizing the capacity of multiple deletion channels – there the authors consider deletion patterns which are "undetectable"; for example, a deletion in the deletion channel $\mathcal{C}_1$ in the cascade model is undetectable since none of the traces will reflect that input symbol. However, our channel decomposition result does not appear in [HM14].

Edit graph ([Gus97]): Similar graph constructs have been defined in related problems on common supersequences and subsequences (see [NO01] for example). This graph is closely related to the error events in the remnant channel. We start with a simple case and generalize subsequently. Define a directed graph called *edit graph* given two sequences f and g, where every path connecting the "origin" to the "destination" on the edit graph yields a supersequence h of f, g, where h is "covered" by f, g – i.e., each symbol of h comes from either f or g or both. In other words, given that f and g are the outputs of the remnant channel (with two outputs), each path from the origin of the edit graph to the destination corresponds to a possible input h to the remnant channel and to an error event which resulted in outputs f, g with input h.

For f and g in $\mathcal{A}^*$, we form a directed graph $\mathcal{G}(f, g)$ with $(|f|+1)(|g|+1)$ vertices each labelled with a distinct pair $(i, j), 0 \leq i \leq |f|,\ 0 \leq j \leq |g|$. A directed edge $(i_1, j_1) \rightarrow (i_2, j_2)$ exists iff at least one of the following holds:

1. $i_2 - i_1 = 1$ and $j_1 = j_2$, or
2. $j_2 - j_1 = 1$ and $i_1 = i_2$, or

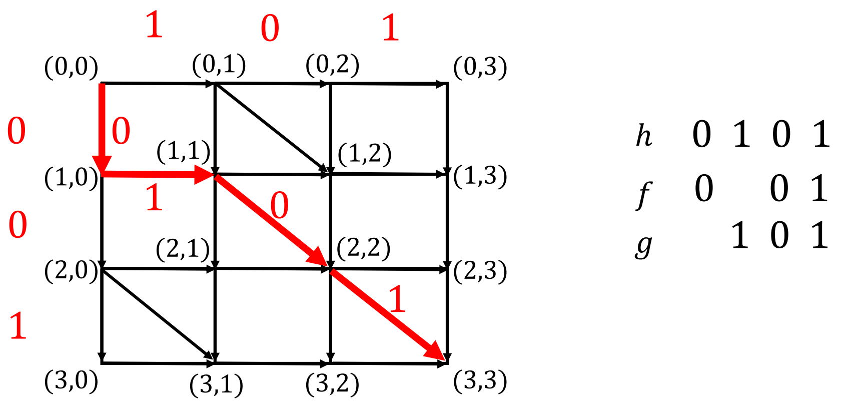


Figure 2.4: Edit graph for sequences f = '001' and g = '101'. Make a grid so the vertical edges are aligned with a symbol in f and horizontal edges with g as shown. A diagonal edge $(i-1, j-1) \to (i, j)$ exists if $f_i = g_j$. The thick red edges form a path from the origin to the destination; this path corresponds to h ='0101' – sequentially append the corresponding symbol to which each edge is aligned. It can also be verified that h is a supersequence of both f and g, and could be obtained as a covering of f and g; the path itself gives one such covering. This covering also corresponds to an error event (or a deletion pattern) in the remnant channel which would result in outputs f and g with input h = '0101' – the deletion pattern is shown in the figure.

3. $i_2 - i_1 = 1$, $j_2 - j_1 = 1$ and $f_{i_2} = g_{j_2}$,

where f_i is the i^{th} symbol of the sequence f. The origin is the vertex $(0,0)$ and the destination $(|f|,|g|)$.

Let $p = ((i_1, j_1), (i_2, j_2), ..., (i_m, j_m))$ be a path in $\mathcal{G}(f,g)$. We define $s(p)$ to be the sequence corresponding to the path. Intuitively, $s(p)$ is formed by appending symbols in the following way: append the corresponding f symbol for a vertical edge, g symbol for horizontal edge, and f or g symbol for diagonal edge (see example Fig. 2.4). Any path from $(0,0)$ to $(|f|,|g|)$ corresponds to a supersequence of f and g and which is covered by f and

g. More formally, define $s(p) \triangleq x_1x_2...x_{m-1}$ where

$$x_k = \begin{cases} f_{i_{k+1}} & \text{if } j_k = j_{k+1}, \\ g_{j_{k+1}} & \text{if } i_k = i_{k+1}, \\ f_{i_{k+1}} & \text{else.} \end{cases}$$

The construct of edit graph can be extended to more than 2 sequences with the same idea. For sequences $f_1, f_2, ..., f_t$, construct a t-dimensional grid with a number of vertices $(|f_1| + 1)(|f_2| + 1)...(|f_t| + 1)$ labeled from $(0, 0, ..., 0)$ to $(|f_1|, |f_2|, ..., |f_t|)$. A vertex $u = (i_1, i_2, ..., i_t)$ is connected to $v = (j_1, j_2, ..., j_t)$ (we say $u \to v$) iff both of the following conditions are met:

- $j_l = i_l$ or $j_l = i_l + 1 \ \forall\ l \in [t]$, i.e., $(i_1, ..., i_t)$ and $(j_1, ..., j_t)$ are vertices of a particular unit cube. Only these type of vertices can share an edge in the grid graph.
- Let $\mathcal{T} \subseteq [t]$ be the collection of indices where $j_l = i_l + 1$. Then $f_{l_{j_l}}$ is equal $\forall\ l \in \mathcal{T}$. For example in 4 dimensional grid, consider the two vertices $(10, 5, 8, 2)$ and $(10, 6, 9, 2)$. In this case $\mathcal{T} = \{2, 3\}$ since the second and third coordinates differ by 1. Therefore $(10, 5, 8, 2) \to (10, 6, 9, 2)$ iff $f_{2_5} = f_{3_9}$. Note that if only one coordinate differs by 1 in the two vertices, a directed edge always exists (in other words all non-diagonal edges exist).

Define the vertex $(0, ..., 0)$ to be the origin of this graph and the vertex $(|f_1|, ..., |f_t|)$ to be the destination. If $|f_j| = O(n)\ \forall\ j$, this graph has a number of vertices $O(n^t)$ and a maximum number of edges $O((2n)^t)$ since each vertex has at most $2^t - 1$ outgoing edges.

Infiltration product (introduced in section 6.3 of [Lot97]): The infiltration product has been extensively used in [Lot97], as a tool in non-commutative algebra. Here, we give an edit-graph interpretation of this tool. A formal algebraic definition of the infiltration product is in Appendix 2.7.7. Using the edit graph we can construct the set of possible supersequences $\mathcal{S}(f,g)$ of f, g that are covered by the symbols in f and g. Indeed, multiple paths could yield the same supersequence and we can count the number of distinct ways $\mathbf{N}(h;f,g)$ one can construct the same supersequence h from f, g. We can informally define the *infiltration product* $f \uparrow g$ of f and g, as a polynomial with monomials the supersequences h in $\mathcal{S}(f,g)$ and coefficients $\langle f \uparrow g, h\rangle$ equal to $\mathbf{N}(h;f,g)$. For the example in Fig. 2.4, there is exactly one path corresponding to '101001' and hence $\langle 001 \uparrow 101, 101001\rangle = 1$ and similarly $\langle 001 \uparrow 101, 01001\rangle = 2$. One could find these coefficients for all relevant sequences and form the polynomial as described. We now give additional examples (see 6.3.14 in [Lot97]). Let $\mathcal{A} = \{a,b\}$, then

- $ab \uparrow ab = ab + 2aab + 2abb + 4aabb + 2abab$,
- $ab \uparrow ba = aba + bab + abab + 2abba + 2baab + baba$.

The infiltration operation is commutative and associative, and infiltration of two sequences $f \uparrow g$ is a polynomial with variables of length (or *degree*) at most $|f| + |g|$; see [Lot97]. The definition of infiltration extends to two polynomials via distributivity (precisely defined in Appendix 2.7.7), and consequently to multiple sequences as well. For multiple sequences, infiltration has the same edit graph interpretation: $\langle f_1 \uparrow f_2 \uparrow ... \uparrow f_t, w\rangle$ is the number of distinct ways of constructing w as a supersequence of $f_1, f_2, ..., f_t$ so that the construction covers w, i.e., construct the t-dimensional edit graph of $f_1, f_2, ..., f_t$ and count the number of paths corresponding to w.

Table of notation	
$\mathcal{A}$	A set
X	A random variable or a random vector
x	A scalar or a vector variable
$\lvert x \rvert$	Length of the sequence x
$[i:j]$	$\{i, i+1, ..., j\}$
$x^{(i\rightarrow s)}$	$(x_1, x_2, ..., x_{i-1}, s, x_{i+1}, ..., x_n)$
$\binom{f}{g}$	Binomial coefficient: number of subsequence patters of f equal to g
$\mathbf{F}(p, v)$	Relaxed binomial coefficient: $\mathbb{E}_{Z\sim p}\binom{Z}{v}$
$\langle f \uparrow g, h\rangle$	Infiltration product: number of ways of obtaining sequence h as a "covered" supersequence of f and g

2.2 Sequencewise ML for the deletion channel

2.2.1 A continuous optimization formulation for the single trace ML

We here consider the single-trace ML decoding in (2.2), assuming that the output sequence $Y = y$ is non-empty. To the best of our knowledge, the only known method to solve (2.2) involves solving a combinatorial optimization, essentially iterating over all possible choices of x and computing the objective value for each of the choices. The reason is that there seems to be no discernible pattern exhibited by the true ML sequence; as we see in the table below, the true ML sequence at times extends a few runs, and at times even introduces new runs! Here, we list a few examples of the trace and the corresponding 10-length ML sequences.

y	The set of all x_{ml} sequences
10111	1100111111
1010	1101010100
000100	0000001000, 0000010000, 0000011000
111101	1111111001, 1111111011

In this section, we show that one could equivalently solve the continuous formulation of (2.2) to obtain a solution for (2.2). Before presenting the main result, we first state a useful lemma which factors a given coordinate p_i out of the relaxed binomial coefficient $\mathbf{F}(p, y)$ we introduced in Definition 2.1.

Lemma 2.1. *For $p = (p_1, p_2, .., p_i, ..., p_n)$ and $Y = y = y_1...y_m$ with $n \geq m > 0$, we have*

$$\mathbf{F}(p, y) = \mathbf{F}(p_{[n]\backslash\{i\}}, y) + p_i \sum_{k|y_k=1} \mathbf{F}(p_{[1:i-1]}, y_{[1:k-1]})\mathbf{F}(p_{[i+1:n]}, y_{[k+1,m]})$$
$$+(1-p_i) \sum_{k|y_k=0} \mathbf{F}(p_{[1:i-1]}, y_{[1:k-1]})\mathbf{F}(p_{[i+1:n]}, y_{[k+1,m]}).$$

Recall that $\mathbf{F}(p, y)$ sums over all m-length subsets $\mathcal{S}$ and associates $p_{\mathcal{S}}$ with y. Intuitively, this recursive relationship considers separately the cases where

- $i \notin \mathcal{S}$,
- $i \in \mathcal{S}$ and is associated with a particular y_k where $y_k = 1$,
- $i \in \mathcal{S}$ and is associated with a particular y_k where $y_k = 0$.

The detailed proof can be found in Appendix 2.7.2. It is clear from Lemma 2.1 that $\mathbf{F}(p, y)$ is affine when projected onto each coordinate p_i. Thus, the extrema of $\mathbf{F}(p, y)$ must occur at the boundary of the support set of p_i; i.e., at either $p_i = 0$ or $p_i = 1$. Combining this with the fact that $\mathbf{F}(\cdot)$ is a relaxed version of the binomial coefficient, we observe that the maximization problem in (2.2) is equivalent to its real-valued extension. The following result makes this precise.

Theorem 2.4. *The ML decoding problem for the single-trace deletion channel*

$$\max_{x\in\{0,1\}^n} \binom{x}{y} \tag{2.4}$$

is equivalent to the problem

$$\max_{p\in[0,1]^n} \mathbf{F}(p,y). \tag{2.5}$$

Furthermore, given any non-integral $p^* \in [0,1]^n$ that maximizes $\mathbf{F}(p,y)$, we can construct a corresponding integral solution $x^* \in \{0,1\}^n$ that maximizes $\mathbf{F}(x,y)$ and consequently also maximizes $\binom{x}{y}$.

Proof. As noted earlier, we have $\binom{x}{y} = \mathbf{F}(x,y)$. Therefore, we are interested in proving the following:

$$\max_{x\in\{0,1\}^n} \mathbf{F}(x,y) \equiv \max_{p\in[0,1]^n} \mathbf{F}(p,y), \tag{2.6}$$

where $\equiv$ refers to that the two problems are equivalent (have the same optimal objective value). We prove this by applying the following claim.

Claim: Given any feasible $p = (p_1, p_2, ..., p_i, ..., p_n)$, at least one of the following holds true:

- $\mathbf{F}(p^{(i\to 0)}, y) \geq \mathbf{F}(p,y)$. Recall from notation that $p^{(i\to 0)} = (p_1, p_2, ..., p_{i-1}, 0, p_{i+1} ..., p_n)$ is the vector where the i^{th} coordinate is replaced by 0.
- $\mathbf{F}(p^{(i\to 1)}, y) \geq \mathbf{F}(p,y)$.

Thus if p^* is an optimal solution to (2.5) with $p_i \in (0,1)$, then at least one of $p^{(i\to 0)}$ or $p^{(i\to 1)}$ is also an optimal solution. Sequentially applying this argument for each coordinate of p shows that there exists a point in $\{0,1\}^n$ which is an optimal solution to (2.5) and consequently to (2.4).

It remains to prove our claim. We use Lemma 2.1 to factor out p_i terms in $\mathbf{F}(p, Y)$:

$$\mathbf{F}(p, y) = \mathbf{F}(p_{[n]\setminus\{i\}}, y) + p_i \sum_{k|y_k=1} \mathbf{F}(p_{[1:i-1]}, y_{[1:k-1]})\mathbf{F}(p_{[i+1:n]}, y_{[k+1,m]})$$

$$+(1-p_i) \sum_{k|y_k=0} \mathbf{F}(p_{[1:i-1]}, y_{[1:k-1]})\mathbf{F}(p_{[i+1:n]}, y_{[k+1,m]}).$$

Now we express $\mathbf{F}(p^{(i\to0)}, y)$ and $\mathbf{F}(p^{(i\to1)}, y)$ as

$$\mathbf{F}(p^{(i\to0)}, y) = \mathbf{F}(p_{[n]\setminus\{i\}}, y) + \sum_{k|y_k=0} \mathbf{F}(p_{[1:i-1]}, y_{[1:k-1]})\mathbf{F}(p_{[i+1:n]}, y_{[k+1,m]}),$$

$$\mathbf{F}(p^{(i\to1)}, y) = \mathbf{F}(p_{[n]\setminus\{i\}}, y) + \sum_{k|y_k=1} \mathbf{F}(p_{[1:i-1]}, y_{[1:k-1]})\mathbf{F}(p_{[i+1:n]}, y_{[k+1,m]}).$$

Because $0 \le p_i \le 1$ it directly follows that

$$\min\left\{\mathbf{F}(p^{(i\to0)}, y), \mathbf{F}(p^{(i\to1)}, y)\right\} \le \mathbf{F}(p, y) \le \max\left\{\mathbf{F}(p^{(i\to0)}, y), \mathbf{F}(p^{(i\to1)}, y)\right\},$$

thus proving our claim.

□

The real-valued optimization problem in (2.5) falls under the umbrella of signomial optimization which is, in general, NP-hard (see for example, [Xu14], [CS16]). A standard technique for signomial optimization uses convexification strategies to approximate the optimal value. In particular, as stated in [CS16], the main observation underlying their methods is that certifying the nonnegativity of a signomial with at most *one negative coefficient* can be accomplished efficiently. However, there are two problems with this approach in relation to our work – 1. when expressed as a signomial optimization problem, *all* the coefficients are negative in the ML optimization objective function, and 2. the objective function has an exponential number of signomial terms as can be seen from Definition 2.1. As a result, such strategies turn out to not be useful for the ML optimization problem. For instance, the techniques in [CS16] resulted in the bound $\mathbf{F}(p, Y) \le \binom{|p|}{|Y|}$ for most instances of p and Y, where $|\cdot|$ denotes the length of the vector/sequence. This is a trivial bound that uses no information about p and Y other than their lengths. Moreover, with a slight change of variables, (2.5) could also be expressed as a maximization of a convex function in a convex set. With that being said, it is still unclear if (2.5) is solvable in polynomial time or not.

2.2.2 ML via gradient ascent

Given the continuous variable formulation of the ML problem in (2.5), a natural heuristic to find an estimate of the ML sequence is to employ *projected gradient ascent* to solve (2.5). The algorithm, in short, can be described as follows (the exact algorithm is detailed as Alg. 2.1):
Step I: Start from a randomly chosen interior point (in our case, we start from
$p = (0.5, 0.5, ..., 0.5)$, the point corresponding to the uniform distribution).
Step II: Take a small step in the direction of the gradient $\nabla_p \, \mathbf{F}(p, y)$.
Step III: If the gradient step results in p moving out of $[0, 1]^n$, project it back onto $[0, 1]^n$.
Repeat Steps II and III until convergence.
Step IV: From the final p, determine the closest binary sequence to be the reconstructed sequence.

Moreover in Appendix 2.7.6, we show using Lemma 2.1 that $\nabla_p \, \mathbf{F}(p, y)$ can be computed in $O(n^2)$ as a "by-product" of computing $\mathbf{F}(p, y)$.

2.2.3 A heuristic for multiple traces

The continuous variable ML formulation in (2.5) optimizes over the distributions p, instead of sequences x. In particular, we proved the following:

$$\max_{x \in \{0,1\}^n} \binom{x}{y} \equiv \max_{p \in [0,1]^n} \mathbf{F}(p, y) \equiv \max_{p \in [0,1]^n} \mathbb{E}_{Z \sim p} \binom{Z}{y}.$$

At this point, one could ask how this formulation extends to multiple traces $Y^1 = y^1, Y^2 = y^2, ..., Y^t = y^t$. The following theorem gives such a continuous optimization formulation with multiple traces.

Algorithm 2.1 Single trace projected gradient ascent for ML

```
1: Input: Blocklength $n$, Trace $Y = y$, Initial point $p = (p_1, p_2, ..., p_n)$, step-size $\epsilon$, Max
   iterations $M$, Convergence criteria $C$
2: Outputs: Estimated sequence $\hat{X}$
3: Iteration count $j = 0$
4: while $C$ is FALSE and $j < M$ do
5:     $p \leftarrow p + \epsilon \frac{\nabla_p \mathbf{F}(p,y)}{\mathbf{F}(p,y)}$
6:     Replace $p_i \leftarrow 1$ for all $i : p_i > 1$
7:     Replace $p_i \leftarrow 0$ for all $i : p_i < 0$
8:     $j \leftarrow j + 1$
9: For each $i$, set $\hat{X}_i = \mathbb{1}\{p_i > 0.5\}$.
10: return $\hat{X} = \hat{X}_1 \hat{X}_2 ... \hat{X}_n$
```

Theorem 2.5. *The ML decoding with multiple traces*

$$\max_{x \in \{0,1\}^n} \binom{x}{y^1} \binom{x}{y^2} ... \binom{x}{y^t} \tag{2.7}$$

is equivalent to

$$\max_{p \in [0,1]^n} \mathbb{E}_{Z \sim p} \left[\binom{Z}{y^1} \binom{Z}{y^2} ... \binom{Z}{y^t} \right]. \tag{2.8}$$

Furthermore, given any non-integral $p^* \in [0,1]^n$ that maximizes $\mathbb{E}_{Z \sim p} \left[\binom{Z}{y^1} \binom{Z}{y^2} ... \binom{Z}{y^t} \right]$, we can construct a corresponding integral solution $x^* \in \{0,1\}^n$ that also maximizes $\binom{x}{y^1} \binom{x}{y^2} ... \binom{x}{y^t}$.

Proof. This theorem can be proved in the same way as Theorem 2.4, by showing that $\mathbb{E}_{Z \sim p} \left[\binom{Z}{y^1} \binom{Z}{y^2} ... \binom{Z}{y^t} \right]$ is an affine function of each p_i; here we only prove this fact and the rest of the arguments follow exactly as in the proof of Theorem 2.4.

To show this we use Lemma 2.2 stated below; this Lemma is also closely related to the channel equivalence of Theorem 2.3 (see Appendix 2.7.3).

Lemma 2.2. *For $h, f_1, f_2, ..., f_m \in \mathcal{A}^*$,*

$$\binom{h}{f_1}\binom{h}{f_2}...\binom{h}{f_m} = \sum_{w\in\mathcal{A}^*} \langle f_1 \uparrow f_2 \uparrow ... \uparrow f_m, w\rangle \binom{h}{w}.$$

Using Lemma 2.2, we now have

$$\begin{aligned}\mathbb{E}_{Z\sim p}\left[\binom{Z}{y^1}\binom{Z}{y^2}...\binom{Z}{y^t}\right] &= \mathbb{E}_{Z\sim p}\sum_{w\in\mathcal{A}^*}\langle y^1 \uparrow y^2 \uparrow ... \uparrow y^t, w\rangle\binom{Z}{w}\\ &= \sum_{w\in\mathcal{A}^*}\langle y^1 \uparrow y^2 \uparrow ... \uparrow y^t, w\rangle\mathbb{E}_{Z\sim p}\binom{Z}{w}\\ &= \sum_{w\in\mathcal{A}^*}\langle y^1 \uparrow y^2 \uparrow ... \uparrow y^t, w\rangle\mathbf{F}(p,w).\end{aligned}$$

Note that $\mathbf{F}(p, w)$ is affine in each p_i. Thus $\mathbb{E}_{Z\sim p}\left[\binom{Z}{y^1}\binom{Z}{y^2}...\binom{Z}{y^t}\right]$ is a linear combination of affine functions of each p_i, and hence is also affine in each p_i. □

The formulation of (2.8), by itself, is not very useful as it is unclear on how to efficiently compute $\mathbb{E}_{Z\sim p}\left[\binom{Z}{y^1}\binom{Z}{y^2}\dots\binom{Z}{y^t}\right]$. Indeed, if $\binom{Z}{y^i}$ and $\binom{Z}{y^j}$ are independent, the expectation of products would decompose into a product of expectations, i.e., $\prod_j \mathbb{E}_{Z\sim p}\binom{Z}{y^j} = \prod_j \mathbf{F}(p, y^j)$, and each of the terms in the product can be computed in $O(n^2)$ as detailed in Appendix 2.7.5 – this is however not the case as $\binom{Z}{y^i}$ and $\binom{Z}{y^j}$ are not independent.

Having said that, we can now solve the maximization problem $\arg\max_{p\in[0,1]^n}\prod_{j=1}^t \mathbf{F}(p, y^j)$ and hope that the resultant solution is also a good solution for $\arg\max_{p\in[0,1]^n}\mathbb{E}_{Z\sim p}\left[\binom{Z}{y^1}...\binom{Z}{y^t}\right]$; Algorithm 2.2 makes this idea precise. Moreover, instead of maximizing $\prod_{j=1}^t \mathbf{F}(p, y^j)$, we can further simplify the gradient computations by taking the log of the objective function, i.e., we solve $\arg\max_{p\in[0,1]^n}\sum_{j=1}^t \log\mathbf{F}(p, y^j)$. This heuristic turns out to perform well in a variety of situations, as illustrated in Section 2.5. As for the complexity, note that Alg. 2.2 involves the computation of t gradients (each of which takes $O(n^2)$) at each gradient iteration. For a fixed number of max iterations M, the complexity of the algorithm is $O(n^2t)$.

Algorithm 2.2 Trace reconstruction heuristic via projected gradient ascent

```
1: Input: Blocklength $n$, Traces $Y^1 = y^1, Y^2 = y^2, ..., Y^t = y^t$, Initial point $p = (p_1, p_2, ..., p_n)$, step-size $\epsilon$, Max iterations $M$, Convergence criteria $C$
2: Outputs: Estimated sequence $\hat{X}$
3: Iteration count $j = 0$
4: while $C$ is FALSE and $j < M$ do
5:     $p \leftarrow p + \epsilon \sum_{j=1}^{t} \frac{\nabla_p \mathbf{F}(p, y^j)}{\mathbf{F}(p, y^j)}$
6:     Replace $p_i \leftarrow 1$ for all $i : p_i > 1$
7:     Replace $p_i \leftarrow 0$ for all $i : p_i < 0$
8:     $j \leftarrow j + 1$
9: For each $i$, set $\hat{X}_i = \mathbb{1}\{p_i > 0.5\}$.
10: return $\hat{X} = \hat{X}_1 \hat{X}_2 ... \hat{X}_n$
```

2.3 Symbolwise MAP for the single-trace deletion channel

We here develop an algorithm to compute the symbolwise posterior probabilities for the single-trace deletion channel when the input symbols are independently generated with arbitrary priors. Consider the single deletion channel model in Fig. 2.2, where $X = X_1...X_n$, each input symbol is generated $X_i \sim$ ind. Ber (p_i), and we observe the trace $Y = y = y_1 y_2 ... y_m$ with $m \leq n$. Define the vector of priors as $p \triangleq (p_1, p_2, ..., p_n)$. We first give an $O(n^2)$ algorithm to calculate the posterior probabilities $\Pr(X_i = 1|Y = y)$, which in turn provides the symbolwise MAP estimate for the considered model. We then show how this algorithm can be used for trace reconstruction. We take three steps to present the algorithm.

An expression for $\Pr(X_i = 1|Y = y)$**.** Let $\Pr(X_i = 1) = p_i$. As a first step, we have

$$\Pr(X_i = 1|Y = y) = \frac{\Pr(X_i = 1, Y = y)}{\Pr(Y = y)} = \frac{\sum\limits_{x|x_i=1} \Pr(X = x) \Pr(Y = y|X = x)}{\sum_x \Pr(X = x) \Pr(Y = y|X = x)}$$

$$\overset{(a)}{=} \frac{\sum\limits_{x|x_i=1} \Pr(X=x)\binom{x}{y}}{\sum_x \Pr(X=x)\binom{x}{y}}, \tag{2.9}$$

where (a) is because for a deletion channel $\Pr(Y=y|X=x) = \binom{x}{y}\delta^{|x|-|y|}(1-\delta)^{|y|}$. To proceed, we need to evaluate the summation in the numerator and the denominator. Theorem 2.6 expresses (2.9) in terms of relaxed binomial coefficient terms $\mathbf{F}(\cdot)$. Recall that $\mathbf{F}(p,y) \triangleq \mathbb{E}_{X\sim p}\binom{X}{y}$, which is the denominator term in (2.9).

Theorem 2.6. *Let $X = X_1...X_n$ where $X_i \sim$ ind. Ber (p_i), and let $Y = y$ be the observed trace when X is passed through a deletion channel. Then,*

$$\Pr(X_i=1|Y=y) = \frac{p_i}{\mathbf{F}(p,y)}\left(\mathbf{F}(p_{[n]\setminus\{i\}},y) + \sum_{k|y_k=1} \mathbf{F}(p_{[1:i-1]},y_{[1:k-1]})\mathbf{F}(p_{[i+1:n]},y_{[k+1,m]})\right). \tag{2.10}$$

Proof. The proof of this theorem employs the same trick used in the proof of Lemma 2.1. From (2.9), we have

$$\Pr(X_i=1|Y=y) = \frac{\sum\limits_{x|x_i=1} \Pr(X=x)\binom{x}{y}}{\mathbf{F}(p,y)}.$$

Now,

$$\begin{aligned}\sum_{x|x_i=1} \Pr(X=x)\binom{x}{y} &= \sum_{x|x_i=1} \Pr(X=x) \sum_{\substack{\mathcal{S}\subseteq[n]\\|\mathcal{S}|=m}} \mathbb{1}\{x_{\mathcal{S}}=y\}\\ &= \sum_{\substack{\mathcal{S}\subseteq[n]\\|\mathcal{S}|=m}} \sum_{\substack{x|x_i=1\\x_{\mathcal{S}}=y}} \Pr(X=x). \end{aligned} \tag{2.11}$$

We first separate the outer summation into two cases: (a) $\mathcal{S}|i \notin \mathcal{S}$ and (b) $\mathcal{S}|i \in \mathcal{S}$. We can express the first case as

$$\sum_{\substack{\mathcal{S}\subseteq[n]\\|\mathcal{S}|=m, i\notin\mathcal{S}}} \sum_{\substack{x|x_i=1\\x_{\mathcal{S}}=y}} \Pr(X=x) = \sum_{\substack{\mathcal{S}\subseteq[n]\setminus\{i\}\\|\mathcal{S}|=m}} \sum_{\substack{x|x_i=1\\x_{\mathcal{S}}=y}} \Pr(X=x)$$

$$= \sum_{\substack{\mathcal{S}\subseteq[n]\setminus\{i\}\\|\mathcal{S}|=m}} \sum_{\substack{x|x_i=1\\x_\mathcal{S}=y}} \left(\Pr(X_i=1)\Pr(X_\mathcal{S}=y)\Pr(X_{[n]\setminus\mathcal{S}\cup\{i\}} = x_{[n]\setminus\mathcal{S}\cup\{i\}}) \right)$$

$$= \sum_{\substack{\mathcal{S}\subseteq[n]\setminus\{i\}\\|\mathcal{S}|=m}} p_i \Pr(X_\mathcal{S}=y) \left(\sum_{\substack{x|x_i=1\\x_\mathcal{S}=y}} \Pr(X_{[n]\setminus\mathcal{S}\cup\{i\}} = x_{[n]\setminus\mathcal{S}\cup\{i\}}) \right)$$

$$= \sum_{\substack{\mathcal{S}\subseteq[n]\setminus\{i\}\\|\mathcal{S}|=m}} p_i \Pr(X_\mathcal{S}=y) \left(\sum_{(x_j|j\in[n]\setminus\mathcal{S}\cup\{i\})} \Pr(X_{[n]\setminus\mathcal{S}\cup\{i\}} = x_{[n]\setminus\mathcal{S}\cup\{i\}}) \right)$$

$$= p_i \sum_{\substack{\mathcal{S}\subseteq[n]\setminus\{i\}\\|\mathcal{S}|=m}} \Pr(X_\mathcal{S}=y) = p_i \mathbf{F}(p_{[n]\setminus\{i\}}, y). \tag{2.12}$$

For the second term, we express the set $\mathcal{S}$ as a union $\mathcal{S}=\mathcal{S}'\cup\{i\}\cup\mathcal{S}''$ such that $\mathcal{S}'\subseteq[i-1]$ and $\mathcal{S}''\subseteq[i+1:n]$ to get:

$$\sum_{\substack{\mathcal{S}\subseteq[n]\\|\mathcal{S}|=m,\\i\in\mathcal{S}}} \sum_{\substack{x|x_i=1\\x_\mathcal{S}=y}} \Pr(X=x) = \sum_{k=1}^{m} \sum_{\substack{\mathcal{S}\subseteq[n],\\|\mathcal{S}|=m,\\\mathcal{S}_k=i}} \sum_{\substack{x|x_i=1\\x_\mathcal{S}=y}} \Pr(X=x)$$

$$= \sum_{k=1}^{m} \sum_{\substack{\mathcal{S}'\subseteq[i-1]\\|\mathcal{S}'|=k-1}} \sum_{\substack{\mathcal{S}''\subseteq[i+1:n]\\|\mathcal{S}''|=m-k}} \sum_{\substack{x|x_i=1\\x_\mathcal{S}=y}} \mathbb{1}_{\{y_k=1\}} \Pr(X=x)$$

$$= \sum_{k:y_k=1} \sum_{\substack{\mathcal{S}'\subseteq[i-1]\\|\mathcal{S}'|=k-1}} \sum_{\substack{\mathcal{S}''\subseteq[i+1:n]\\|\mathcal{S}''|=m-k}} \sum_{\substack{x|x_i=1\\x_{\mathcal{S}'}=y_{[1:k-1]}\\x_{\mathcal{S}''}=y_{[k+1:m]}}} \left(\Pr(X_i=1)\Pr(X_{\mathcal{S}'}=y_{[1:k-1]})\Pr(X_{\mathcal{S}''}=y_{[k+1:m]}) \right.$$

$$\left. \Pr(X_{[n]\setminus\mathcal{S}'\cup\mathcal{S}''\cup\{i\}} = x_{[n]\setminus\mathcal{S}'\cup\mathcal{S}''\cup\{i\}}) \right)$$

$$= p_i \sum_{k:y_k=1} \left(\left(\sum_{\substack{\mathcal{S}'\subseteq[i-1]\\|\mathcal{S}'|=k-1}} \Pr(X_{\mathcal{S}'}=y_{[1:k-1]}) \right) \left(\sum_{\substack{\mathcal{S}''\subseteq[i+1:n]\\|\mathcal{S}''|=m-k}} \Pr(X_{\mathcal{S}''}=y_{[k+1:m]}) \right) \right.$$

Algorithm 2.3 Symbolwise posterior probabilities with one trace

```
1: Input: Trace $Y = y$, priors $p$
2: Outputs: Posteriors $\Pr(X_i = 1|Y = y)\ \forall\ i$
3: Compute $\mathbf{F}(p_{[1:k]}, y_{[1:j]})\ \forall\ k, j$ and $\mathbf{F}(p_{[k:n]}, y_{[j:m]})\ \forall\ k, j$ via Alg. 2.11
4: for $i = 1 : n$ do
5:     Use (2.10) to compute $\Pr(X_i = 1|Y = y)$
```

$$\left(\sum_{\substack{x|x_i=1 \\ x_{\mathcal{S}'}=y_{[1:k-1]} \\ x_{\mathcal{S}''}=y_{[k+1:m]}}} \Pr(X_{[n]\setminus\mathcal{S}'\cup\mathcal{S}''\cup\{i\}} = x_{[n]\setminus\mathcal{S}'\cup\mathcal{S}''\cup\{i\}}) \right) \Bigg)$$

$$= p_i \sum_{k|y_k=1} \left(\left(\sum_{\substack{\mathcal{S}'\subseteq[i-1] \\ |\mathcal{S}'|=k-1}} \Pr(X_{\mathcal{S}'} = y_{[1:k-1]}) \right) \left(\sum_{\substack{\mathcal{S}''\subseteq[i+1:n] \\ |\mathcal{S}''|=m-k}} \Pr(X_{\mathcal{S}''} = y_{[k+1:m]}) \right) \right)$$

$$= p_i \sum_{k|y_k=1} \mathbf{F}(p_{[1:i-1]}, y_{[1:k-1]})\mathbf{F}(p_{[i+1:n]}, y_{[k+1,m]}). \tag{2.13}$$

Plugging in (2.12) and (2.13) in (2.9) proves the theorem. □

Alg. 2.3 summarizes the computation of $\Pr(X_i = 1|Y = y)$.

A trace reconstruction heuristic with t traces. The posterior probability computation in Alg. 2.3 naturally gives rise to a trace reconstruction heuristic that updates the symbolwise statistics sequentially on the traces, where we use Alg. 2.3 with one trace at a time to continually update $\Pr(X_i = 1|Y = y)$. The overall heuristic is described in Alg. 2.4. Note that the algorithm first needs to compute $\mathbf{F}(p_{[1:k]}, y_{[1:j]})\ \forall\ k, j$ and $\mathbf{F}(p_{[k:n]}, y_{[j:m]})\ \forall\ k, j$ which requires $O(n^2)$ operations, as described in Appendix 2.7.5. Given this, the algorithm iterates over the n indices and computes the posteriors in $O(n)$ for each of the index. Thus, the complexity of the algorithm is $O(n^2)$; note that $m = O(n)$ since y is a deleted version of the input.

Algorithm 2.4 Trace reconstruction via iterative single-trace posterior probabilities

```
1: Input: Traces $Y^1 = y^1, ..., Y^t = y^t$, input length $n$
2: Outputs: Estimate of the input $\hat{X}$
3: Initialize priors $p^{old} = p^{new} \leftarrow (0.5, 0.5, ..., 0.5)$
4: for $l = 1 : t$ do
5:     Use Alg. 2.3 with $p^{old}$ and $y^l$ to update $p^{new}$
6:     $p^{old} \leftarrow p^{new}$
7: for $i = 1 : n$ do
8:     if $p_i^{new} \geq 0.5$ then $\hat{X}_i \leftarrow 1$
9:     else $\hat{X}_i \leftarrow 0$
10: return $\hat{X}_1 \hat{X}_2 ... \hat{X}_n$
```

2.4 Symbolwise MAP for the t-trace deletion channel

In this section, we put to use the ideas and constructs introduced in section 2.1 to exactly compute the symbolwise posterior probabilities given t-traces, which in turn gives a symbolwise MAP estimate with uniform input priors (motivated by average case trace reconstruction). With this formulation the symbolwise MAP with uniform priors can be seen as a minimizer of the symbol error rate in the context of average case trace reconstruction. In Appendix 2.7.8, we also provide a method to compute the symbolwise posterior probabilities for the remnant channel – we encourage the reader to use this appendix as a warm-up. For the t-trace deletion channel, similar expressions arise due to the channel equivalence result of Theorem 2.3.

Let $\mathcal{A} = \{0, 1\}$, and assume that $X \sim$ Uniform $\mathcal{A}^n$. Our goal is to compute the symbolwise posterior probabilities $\Pr(X_i = 1 | Y^1 = y^1, ..., Y^t = y^t)$, where Y^j is the j^{th} trace. Our proposed algorithm is provided in Alg. 2.7 and estimates the symbolwise MAP (with uniform priors). We can directly leverage Alg. 2.7 to reconstruct the input as follows: for each index i, compute $\Pr(X_i = 1|Y^1 = y^1, ..., Y^t = y^t)$ and decide

$$\hat{X}_i = \begin{cases} 1, & \text{if } \Pr(X_i = 1|Y^1 = y^1, ..., Y^t = y^t) \geq 0.5 \\ 0, & \text{otherwise.} \end{cases}$$

Through the rest of this section, we show how to compute $\Pr(X_i = 1|Y^1 = y^1, ..., Y^t = y^t)$ in two steps:

- We first give an expression for $\Pr(X_i = 1|Y^1 = y^1, ..., Y^t = y^t)$ which sums over potentially an exponential number of terms.
- We then show that this summation can be computed in polynomial time (polynomial in the blocklength n).

Step 1: An expression for $\Pr(X_i = 1|Y^1 = y^1, ..., Y^t = y^t)$**.**

Theorem 2.7. *Assume $X \sim$ Uniform $\mathcal{A}^n$ or equivalently $X_i \sim$ Ber(0.5). The posterior probability of the i^{th} bit given the t traces can be expressed as*

$$\begin{aligned}
&\Pr(X_i = 1|Y^1 = y^1, ..., Y^t = y^t) \\
&\qquad = \Bigg[\sum_{k=0}^{n} 2^{n-k-1} \binom{n-1}{k} \sum_{w||w|=k} \langle y^1 \uparrow ... \uparrow y^t, w \rangle \\
&\qquad\quad + \sum_{k=0}^{n} \sum_{j=1}^{k} 2^{n-k} \binom{i-1}{j-1} \binom{n-i}{k-j} \sum_{\substack{w||w|=k, \\ w_j=1}} \langle y^1 \uparrow ... \uparrow y^t, w \rangle \Bigg] \Bigg/ \\
&\qquad\quad \Bigg[\sum_{k=0}^{n} 2^{n-k} \binom{n}{k} \sum_{w||w|=k} \langle y^1 \uparrow ... \uparrow y^t, w \rangle \Bigg]. \qquad (2.14)
\end{aligned}$$

Note that the summation index, $w||w|{=}k$ is over all sequences w of length k; this is an alternate expression for $w|w{\in}\mathcal{A}^k$. We follow this convention throughout the rest of the paper.

Proof.

$$\begin{aligned}\Pr(X_i = 1|Y^1 = y^1, ..., Y^t = y^t) &= \sum_{\substack{x||x|=n,\\ x_i=1}} \Pr(X = x|Y^1 = y^1, ..., Y^t = y^t)\\ &\overset{(a)}{=} \frac{1}{2^n \Pr(Y^1 = y^1, ..., Y^t = y^t)} \sum_{\substack{x||x|=n,\\ x_i=1}} \Pr(Y^1 = y^1, ..., Y^t = y^t|X = x)\\ &\overset{(b)}{=} \frac{1}{2^n \Pr(Y^1 = y^1, ..., Y^t = y^t)} \sum_{\substack{x||x|=n,\\ x_i=1}} \prod_{j=1}^{t} \Pr(Y^j = y^j|X = x),\end{aligned}$$

where (a) uses Bayes' principle and (b) is because each deletion channel acts independently. Recall that for a deletion channel with deletion probability δ, $\Pr(Y = y|X = x) = \binom{x}{y}\delta^{|x|-|y|}(1-\delta)^{|y|}$. Also, using the fact that $\Pr(Y^1 = y^1, ..., Y^t = y^t) = \sum_{x||x|=n} \Pr(x)\Pr(Y^1 = y^1, ..., Y^t = y^t|X = x)$ we have,

$$\Pr(X_i = 1|Y^1 = y^1, ..., Y^t = y^t) = \frac{\sum_{\substack{x||x|=n,\\ x_i=1}} \binom{x}{y^1}...\binom{x}{y^t}}{\sum_{x||x|=n} \binom{x}{y^1}...\binom{x}{y^t}}. \tag{2.15}$$

We first simplify the numerator $\sum_{\substack{x||x|=n,\\ x_i=1}} \binom{x}{y^1}...\binom{x}{y^t}$; the denominator can be simplified using the same approach. Now,

$$\begin{aligned}\sum_{\substack{x||x|=n,\\ x_i=1}} \binom{x}{y^1}...\binom{x}{y^t} &\overset{(a)}{=} \sum_{\substack{x||x|=n,\\ x_i=1}} \sum_{w\in\{0,1\}^*} \binom{x}{w} \langle y^1 \uparrow ... \uparrow y^t, w\rangle\\ &= \sum_{w\in\mathcal{A}^*} \langle y^1 \uparrow ... \uparrow y^t, w\rangle \sum_{\substack{x||x|=n,\\ x_i=1}} \binom{x}{w}\\ &\overset{(b)}{=} \sum_{w\in\mathcal{A}^*} 2^{n-|w|}\langle y^1 \uparrow ... \uparrow y^t, w\rangle \left(\frac{1}{2}\binom{n-1}{|w|} + \sum_{j|w_j=1} \binom{i-1}{j-1}\binom{n-i}{|w|-j}\right)\end{aligned}$$

where (a) is due to Lemma 2.2 and (b) due to Lemma 2.3 (both introduced in [SDDa]); see Appendix 2.7.3 and Appendix 2.7.4 for the statement and proof.

Therefore we have,

$$\sum_{\substack{x||x|=n,\\ x_i=1}} \binom{x}{y^1}\cdots\binom{x}{y^t} \overset{(a)}{=} \sum_{k=0}^{\infty} 2^{n-k-1}\binom{n-1}{k}\sum_{w||w|=k}\langle y^1\uparrow\ldots\uparrow y^t, w\rangle$$
$$+\sum_{k=0}^{\infty}\sum_{j=1}^{k} 2^{n-k}\binom{i-1}{j-1}\binom{n-i}{k-j}\sum_{\substack{w||w|=k,\\ w_j=1}}\langle y^1\uparrow\ldots\uparrow y^t, w\rangle$$
$$\overset{(b)}{=} \sum_{k=0}^{n} 2^{n-k-1}\binom{n-1}{k}\sum_{w||w|=k}\langle y^1\uparrow\ldots\uparrow y^t, w\rangle$$
$$+\sum_{k=0}^{n}\sum_{j=1}^{k} 2^{n-k}\binom{i-1}{j-1}\binom{n-i}{k-j}\sum_{\substack{w||w|=k,\\ w_j=1}}\langle y^1\uparrow\ldots\uparrow y^t, w\rangle, \quad (2.16)$$

where in (a) we first fix $|w|$ and then sum over all w of the given length and (b) holds because the combinatorial terms are 0 when $k > n$. A similar analysis gives

$$\sum_{x||x|=n}\binom{x}{y^1}\cdots\binom{x}{y^t} = \sum_{k=0}^{n} 2^{n-k}\binom{n}{k}\sum_{w||w|=k}\langle y^1\uparrow\ldots\uparrow y^t, w\rangle. \quad (2.17)$$

Plugging (2.16) and (2.17) in (2.15), we get the expression in Theorem 2.7,

$$\Pr(X_i = 1|Y^1 = y^1, \ldots, Y^t = y^t)$$
$$= \left[\sum_{k=0}^{n} 2^{n-k-1}\binom{n-1}{k}\sum_{w||w|=k}\langle y^1\uparrow\ldots\uparrow y^t, w\rangle\right.$$
$$\left.+\sum_{k=0}^{n}\sum_{j=1}^{k} 2^{n-k}\binom{i-1}{j-1}\binom{n-i}{k-j}\sum_{\substack{w||w|=k,\\ w_j=1}}\langle y^1\uparrow\ldots\uparrow y^t, w\rangle\right]\Bigg/$$
$$\left[\sum_{k=0}^{n} 2^{n-k}\binom{n}{k}\sum_{w||w|=k}\langle y^1\uparrow\ldots\uparrow y^t, w\rangle\right].$$

□

Step 2: Dynamic program to compute $\sum_{w||w|=k}\langle y^1\uparrow\ldots\uparrow y^t, w\rangle$ and $\sum_{\substack{w||w|=k,\\ w_j=1}}\langle y^1\uparrow\ldots\uparrow y^t, w\rangle$. Note that the number of sequences w such that $|w| = k$ is $O(2^k)$ so a naive evaluation is exponential in the blocklength n. We can, however, exploit the edit graph to come up with a dynamic program resulting in an algorithm which is polynomial in n.

Recall that in the edit graph, $\langle y^1 \uparrow ... \uparrow y^t, w\rangle$ is equal to the number of distinct paths from the origin $(0, ..., 0)$ to the destination $(|y^1|, ..., |y^t|)$ and which correspond to w. Hence,

(a) $\sum_{w||w|=k} \langle y^1 \uparrow ... \uparrow y^t, w\rangle$ is the number of distinct paths of length k from origin to destination and,

(b) $\sum_{\substack{w||w|=k,\\ w_j=1}} \langle y^1 \uparrow ... \uparrow y^t, w\rangle$ is the number of such paths of length k such that the j^{th} edge of the path corresponds to a '1'.

With this interpretation, the dynamic program for (a) follows naturally – the number of k-length paths from the origin to any vertex is the sum of the number of $(k-1)$-length paths from the origin to all incoming neighbors of the vertex. To make this formal, associate a polynomial (in λ) for each vertex, such that the coefficient of λ^k is equal to the number of paths of length k from the origin to v: we call it the "forward-potential" polynomial $p_v^{for}(\lambda)$ for vertex v, the coefficient of λ^k as earlier is denoted by $\langle p_v^{for}(\lambda), \lambda^k\rangle$. The dynamic program to compute $p_v^{for}(\lambda)$ for all v can be expressed as:

$$p_v^{for}(\lambda) = \sum_{u|u\rightarrow v} \lambda p_u^{for}(\lambda). \tag{2.18}$$

With this definition, we have

$$\sum_{w||w|=k} \langle y^1 \uparrow ... \uparrow y^t, w\rangle = \langle p_{destination}^{for}(\lambda), \lambda^k\rangle.$$

In the example in Fig. 2.4, one could do the following: order the vertices $(0, 0)$ to $(3, 3)$ lexicographically and then compute $p_v^{for}(\lambda)$ in the same order. Because of the directed grid nature of the edit graph, every vertex has incoming neighbors which are lexicographically ahead of itself. Also we initialize $p_{(0,0)}^{for}(\lambda) = 1$. For the example in Fig. 2.4, the forward-potentials are shown in Fig. 2.5. The complexity of this dynamic program is $O(2^t n^{t+1})$ as it goes over $O(n^t)$ vertices and for each vertex it sums $O(2^t)$ polynomials, each of degree $O(n)$.

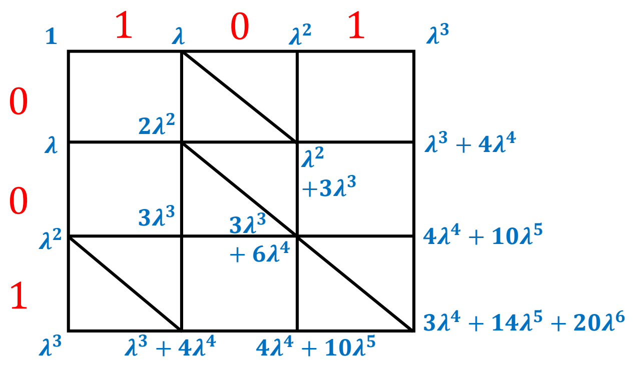


Figure 2.5: The forward-potential $p_v^{for}(\lambda)$ at each vertex.

We compute (b) as follows: pick an edge $(u \to v)$ which corresponds to '1', count the number of $(j-1)$-length paths from origin to u and multiply it with the number of $(k-j)$-length paths from v to the destination – this is exactly the number of paths of length k such that its j^{th} edge is $(u \to v)$. Summing this term for all such edges which correspond to 1 gives us the term in (b). Note that we have already computed the number of k-length paths $(\forall k)$ from origin to every vertex in $p_v^{for}(\lambda)$. We can similarly compute the number of k-length paths $(\forall k)$ from every vertex to the destination as $p_v^{rev}(\lambda)$ – the "reverse potential" polynomial. The dynamic program for $p_v^{rev}(\lambda)$ is:

$$p_v^{rev}(\lambda) = \sum_{u|v \to u} \lambda p_u^{rev}(\lambda), \tag{2.19}$$

with $p_{destination}^{rev}(\lambda) = 1$. The reverse potentials for the example in Fig. 2.4 is shown in Fig. 2.6. Like in the case of forward potential, we first order the vertices reverse lexicographically and then invoke the dynamic program above sequentially to compute the reverse potential polynomial at each vertex.

With this, the term in (b) can be expressed as:

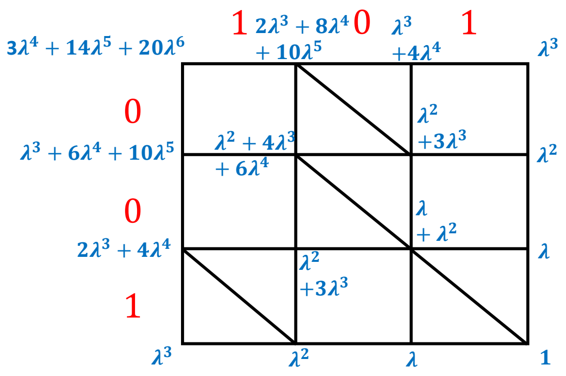
$$\sum_{\substack{w||w|=k,\\ w_j=1}} \langle y^1 \uparrow ... \uparrow y^t, w \rangle = \sum_{\substack{(u,v)|\\ s(u \to v)=1}} \langle p_u^{for}(\lambda), \lambda^{j-1} \rangle \langle p_v^{rev}(\lambda), \lambda^{k-j} \rangle.$$

Algorithm 2.5 Computing the forward-potentials $p_u^{for}(\lambda)$

```
1: Input: Edit graph $\mathcal{G}(y^1, ..., y^t)$
2: Outputs: $p_v^{for}(\lambda)\ \forall\ v$
3: Order the vertices from $(0, 0, ..., 0)$ to $(|y^1|, |y^2|, ..., |y^t|)$ lexicogaphically; let the ordered
   list be $\mathcal{V}$
4: Initialise $p_{(0,...,0)}^{for}(\lambda) \leftarrow 1$
5: for $v \in \mathcal{V}$ do
6:     assign $p_v^{for}(\lambda) \leftarrow \sum_{u|u \to v} \lambda p_u^{for}(\lambda)$
```

Algorithm 2.6 Computing the reverse-potentials $p_u^{rev}(\lambda)$

```
1: Input: Edit graph $\mathcal{G}(y^1, ..., y^t)$
2: Outputs: $p_v^{rev}(\lambda)\ \forall\ v$
3: Order the vertices from $(|y^1|, |y^2|, ..., |y^t|)$ to $(0, 0, ..., 0)$ reverse lexicogaphically; let the
   ordered list be $\mathcal{V}$
4: Initialise $p_{(|y^1|,|y^2|,...,|y^t|)}^{rev}(\lambda) \leftarrow 1$
5: for $v \in \mathcal{V}$ do
6:     assign $p_v^{rev}(\lambda) \leftarrow \sum_{u|v \to u} \lambda p_u^{rev}(\lambda)$
```



$3\lambda^4 + 14\lambda^5 + 20\lambda^6$
1
$2\lambda^3 + 8\lambda^4 + 10\lambda^5$
0
$\lambda^3 + 4\lambda^4$
1
λ^3
0
$\lambda^2 + 3\lambda^3$
$\lambda^3 + 6\lambda^4 + 10\lambda^5$
$\lambda^2 + 4\lambda^3 + 6\lambda^4$
λ^2
0
$\lambda + \lambda^2$
$2\lambda^3 + 4\lambda^4$
λ
$\lambda^2 + 3\lambda^3$
1
λ^3
λ^2
λ
1

Figure 2.6: The reverse-potential $p_v^{rev}(\lambda)$ at each vertex.

Alg. 2.7 now summarizes the computation of the posterior probabilities. This algorithm iterates over all the edges (we have $O((2n)^t)$ of these), and also k, j ($O(n)$ each). The time

Algorithm 2.7 Symbolwise MAP with t traces

```
1: Input: Traces $Y^1 = y^1, ..., Y^t = y^t$, input length $n$
2: Output: $\hat{X} = \hat{X}_1 \hat{X}_2 ... \hat{X}_n$
3: Construct edit graph $\mathcal{G}(y^1, ..., y^t)$
4: Use Alg. 2.5 and Alg. 2.6 on $\mathcal{G}(y^1, ..., y^t)$ to calculate $p_v^{for}(\lambda)$ and $p_v^{rev}(\lambda)$ $\forall$ $v$
5: for $k \in [0:n]$ do
6:     assign $\sum_{w||w|=k} \langle y^1 \uparrow ... \uparrow y^t, w\rangle \leftarrow \langle p^{for}_{destination}(\lambda), \lambda^k\rangle$.
7:     for each $j \in [1:n]$ do
8:         Initialize $temp \leftarrow 0$
9:         for each edge $u \to v \in \mathcal{G}$ do
10:            if $s(u \to v) =$ '1' then
11:                $temp += \langle p_u^{for}(\lambda), \lambda^{j-1}\rangle \langle p_v^{rev}(\lambda), \lambda^{k-j}\rangle$
12:        assign $\sum_{\substack{w||w|=k,\\ w_j=1}} \langle y^1 \uparrow ... \uparrow y^t, w\rangle \leftarrow temp$
13: for $i \in [1:n]$ do
14:     Use (2.14) to compute $\Pr(X_i = 1 | Y^1 = y^1, ..., Y^t = y^t)$
15:     $\hat{X}_i \leftarrow 1$ if $\Pr(X_i = 1 | Y^1 = y^1, ..., Y^t = y^t) > 0.5$ and $\hat{X}_i \leftarrow 0$ otherwise
16: return $\hat{X}_1 \hat{X}_2 ... \hat{X}_n$
```

complexity of Alg. 2.7 hence is $O(2^t n^{t+2})$.

2.5 Numerical results

In this section we show numerics supporting our theoretical results. In all of our experiments, we generate the input sequence uniformly at random (motivated by average case trace reconstruction), and obtain the t traces by passing the input through a deletion channel (with a deletion probability δ) t times. We then reconstruct the input from the obtained traces and measure how *close* the reconstructed sequence is, to the actual input sequence.

We use two metrics to measure the performance of the reconstruction algorithms: 1. *Hamming error rate*, which is defined as the average Hamming distance between the actual input and the estimated sequence divided by the length of the input sequence and 2. *Edit error rate*, which is defined as the average edit distance between the actual input and the estimated sequence divided by the length of the input sequence. The reason for using Hamming error rate is that our goal is to reconstruct a *known-length* sequence, which has been the problem formulation throughout this work. Moreover, the Hamming error rate is also of special interest to us since the symbolwise MAP is an optimal estimator for minimizing the Hamming error rate (see Appendix 2.7.10 for a proof). We also use edit error rate as it is a typical metric used in the context of insertion/deletion channels.

List of trace reconstruction algorithms compared in this work.

Abbreviation	**Description**	**Complexity**
Ind. post. comb.	Independent posterior combination (Alg. 2.8)	$O(n^2t)$
BMA	Bitwise majority alignment of [BKK04] (Alg. 2.9)	$O(nt)$
Trace stats.	Algorithm based on trace symbolwise statistics from [HMP08] (Alg. 2.10)	$O(n^{3.37}+nt)$
Grad asc.	Projected gradient ascent (Alg. 2.2)	$O(n^2t)$
SMAP seq.	Sequential symbolwise MAP heuristic (Alg. 2.4)	$O(n^2t)$
SMAP exact	Exact symbolwise MAP (Alg. 2.7)	$O(n^{t+2}2^t)$

Baseline algorithms:

1. **Independent posterior combination:** As pointed in the background section, computing the posterior probabilities for each deletion channel and combining them as if they came from independent observations does not provide a natural solution for computing the posterior probabilities for the t-trace deletion channel. One could, however, check how such a naive combination of posteriors compares with our reconstruction algorithms

for t-traces. This is detailed as Alg. 2.8. The complexity of this algorithm is $O(n^2t)$ since computing the posteriors takes $O(n^2)$ and we compute posteriors for t traces.

2. **Bitwise Majority Alignment (introduced in [BKK04]):** BMA reconstructs the input sequence by first "aligning" the traces using a pointer for each trace, and then taking the majority of the pointed symbols. BMA is detailed as Alg. 2.9. From an efficiency standpoint, BMA is the most efficient of all the algorithms since it is linear in the blocklength as well as the number of traces ($O(nt)$).

3. **Trace statistics algorithm:** An algorithm based on trace symbol statistics (also called mean-based algorithms and summary statistics algorithms) has been extensively studied for worst-case trace reconstruction (see [HMP08], [DOS], [NP17]). In essence, the algorithm first estimates the "trace symbol statistics" – $\Pr(Y_i = 1)\ \forall\ i$ – from the obtained traces and uses only these estimates to reconstruct X. However, it uses a new set of traces for every position i, thus requiring at least n traces (see (3.6) and the paragraph below (3.8) in [HMP08]). Here we modify the algorithm to adapt them for an arbitrary number of traces; in particular, we reuse the traces while estimating $\Pr(Y_i = 1)\ \forall\ i$. The algorithm is detailed in Alg. 2.10.

   The complexity analysis for this gets tricky since it depends on the algorithm used to solve the set of $2n$ linear programs. The state-of-the-art algorithm for solving a linear program in n variables takes approximately $O(n^{2.37})$ (see [CLS19]); thus the complexity of Trace statistics algorithm is $O(n^{3.37} + nt)$, where the nt term corresponds to the complexity of computing $\hat{p}_j$. However, in our implementation we use the solver from the *SciPy* Python library which uses primal-dual interior point methods for solving linear programs. The complexity of such methods is typically $O(n^3)$ making our implementation $O(n^4 + nt)$. Also note that these are iterative methods and have many hidden constants (such as the number of iterations for convergence).

We note that the state-of-the-art average-case trace reconstruction algorithms in the liter-

Algorithm 2.8 Trace reconstruction via independent posterior combination

```
1: Input: Traces $Y^1 = y^1, ..., Y^t = y^t$, input length $n$
2: Outputs: Estimate of the input $\hat{X}$
3: Initialize priors $p^{old} \leftarrow (0.5, 0.5, ..., 0.5)$
4: for $l = 1 : t$ do
5:     Use Alg. 2.3 with $p^{old}$ and $y^l$ to compute posteriors $p^{l,new}$
6: for $i = 1 : n$ do
7:     if $\prod_{l=1}^{t} p_i^{l,new} \geq \prod_{l=1}^{t} (1 - p_i^{l,new})$ then $\hat{X}_i \leftarrow 1$
8:     else $\hat{X}_i \leftarrow 0$
```

Algorithm 2.9 Bitwise Majority Alignment

```
1: Input: Traces $Y^1 = y^1, ..., Y^t = y^t$, input length $n$
2: Output: estimate of input $\hat{X} = \hat{X}_1 \hat{X}_2 ... \hat{X}_n$.
3: Initialize $c_j = 1$ for $j \in [t]$.
4: Initialize $\hat{X}_i = 1$ for $i \in [n]$.
5: for $i \in [1 : n]$ do
6:     Let $b$ be the majority over all $t$ of $y^j_{c_j}$
7:     $\hat{X}_i \leftarrow b$
8:     Increment $c_j$ for each $j$ such that $y^j_{c_j} = b$
```

ature are applicable in the asymptotic regime where the blocklength n and the number of traces t approach ∞; it is not clear how to adapt such algorithms for a finite blocklength and a small number of traces. It is for this reason that we chose to compare against BMA and Trace statistics algorithm, which can be easily adapted for the finite blocklength regime and for a small number of traces. It should also be noted that the performance of the above two algorithms may not be reliable with a small number of traces (as they are not designed for this regime), yet we include them owing to the lack of better baselines.

Algorithm 2.10 Trace statistics heuristic

```
1: Input: Traces $Y^1 = y^1, ..., Y^t = y^t$, input length $n$
2: Output: estimate of input $\hat{X} = \hat{X}_1\hat{X}_2...\hat{X}_n$.
3: Append each trace $y^j$ with zeros until each of them is of length $n$.
4: Assign $\hat{p}_j \leftarrow \frac{|\{y^l : y^l_j = 1\}|}{t}$.
5: for $i \in [1:n]$ do
6:     Solve the 2 linear programs (3.6) in [HMP08] by fixing $x_i = 0$ and $x_i = 1$: let the optimum value in the two cases be $m_0$ and $m_1$ respectively.
7:     If $m_0 < m_1$, assign $\hat{X}_i = x_i \leftarrow 0$. Else fix $\hat{X}_i = x_i \leftarrow 1$.
```

Algorithms introduced in this paper:

1. **Projected gradient ascent:** Alg. 2.2 used as described, with max iterations $M = 100$ and convergence criteria C set as follows: the percentage difference in $\sum_j \mathbf{F}(p, y^j)$ over two consecutive iterations is less than 0.1%.

2. **Symbolwise MAP sequentially used one trace at a time:** Alg. 2.4 used as described.

3. **Exact symbolwise MAP:** Alg. 2.7 used as described.

Observations: In Fig. 2.7 and Fig. 2.8, we compare the Hamming and edit error rates for the different algorithms described above.

- The 3 algorithms introduced in this work outperform the 3 baselines in most cases. The Hamming error rate of Grad asc. with 2 and 3 traces is a notable exception as it does worse than Ind. post. comb. However, it improves rapidly as we increase the number of traces as seen in Fig. 2.7.

- Both Ind. post. comb. as well as our SMAP seq. struggle with the problem of *diminishing returns* for Hamming error rate as they do not improve much with the number of traces.

This could indicate that considering traces one at a time could fail to accumulate extrinsic information (for instance, it completely neglects the possible alignments given multiple traces); one needs to simultaneously consider multiple traces in order to accomplish this. SMAP seq. however, improves with the number of traces with respect to edit error rate.

- The Grad asc. is the "champion" amongst the algorithms we compare here, when it comes to the edit error rate as illustrated by Fig. 2.8. The Grad asc. was constructed with the aim of maximizing the likelihood of the observed traces, and this in turn seems to have some correlation with minimizing the edit distance – it is not clear why this is the case.

- As seen in Fig. 2.7 (a) and (b), SMAP exact has the minimum Hamming error rate. This supports the fact that symbolwise MAP is the minimizer of the Hamming error rate. However, note that this does not necessarily minimize the edit error rate, as seen from Fig. 2.8 (a) and (b).

2.6 Conclusions and Open Questions

In this chapter we gave, to the best of our knowledge, the first results and techniques to compute posterior distributions over single and multiple deletion channels. We also provided a new perspective on the maximum-likelihood for the deletion channel by showing an equivalence between a discrete optimization problem and a continuous formulation of it. In this process, we introduced a variety of tools (the relaxed binomial coefficient, edit graph and infiltration product) and demonstrated their use for analyzing deletion channels. We also presented numerical evaluations of our algorithms and showed performance improvements over existing trace reconstruction algorithms. An interesting open question is to come up with error rate guarantees for the trace reconstruction algorithms introduced in this chapter. Extending these methods to insertion-deletion-substitutions channels is another open question.

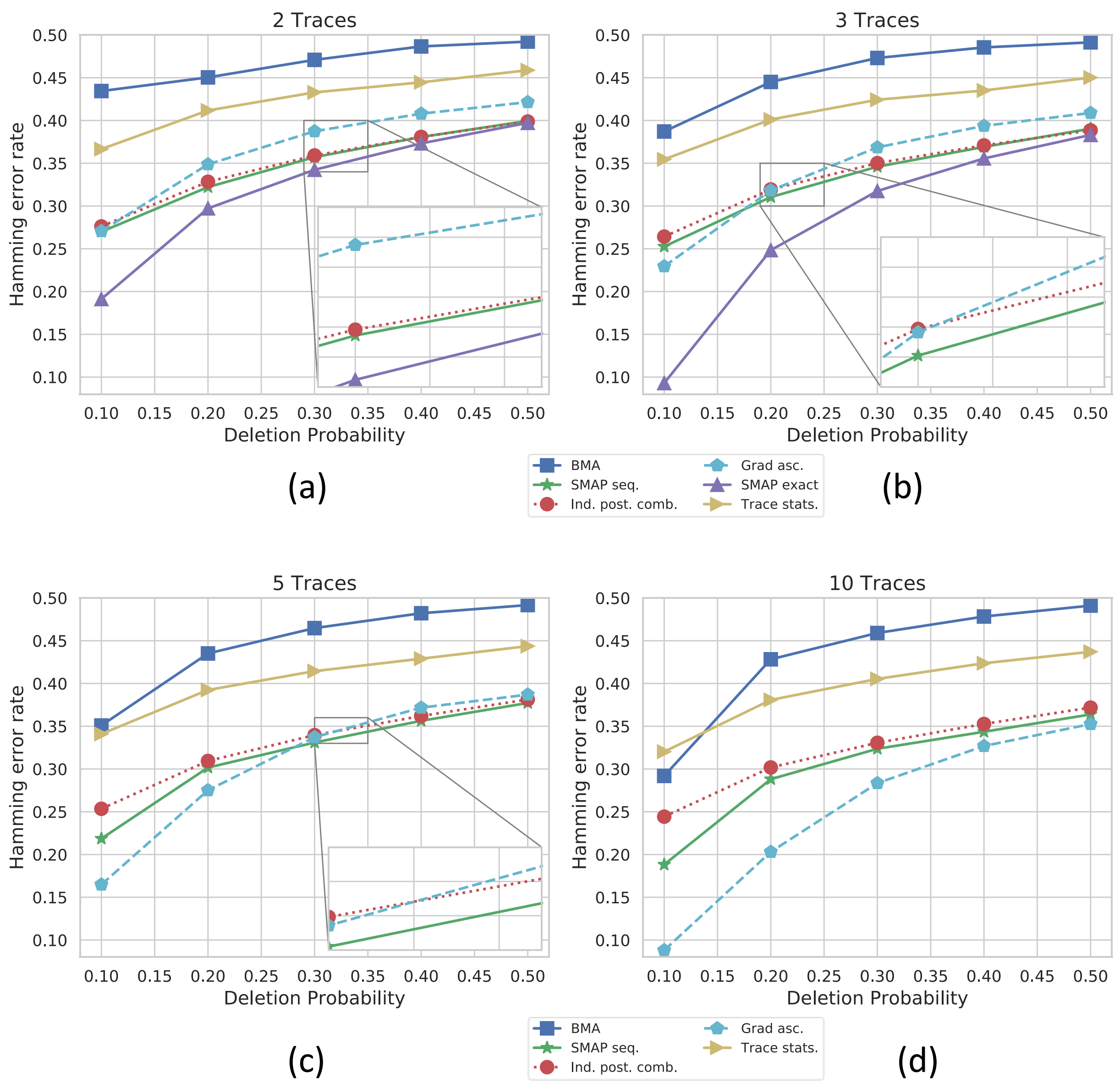


Figure 2.7: Comparison of Hamming error rates for a blocklength $n = 100$ illustrated with 2,3,5 and 10 observed traces. Note that we do not run SMAP exact. for 5 and 10 traces since its complexity grows exponentially with the number of traces. All the subplots are plotted on the same scale to aid comparability across subplots. Few of the subplots which contain algorithms with similar error rates also contain a zoomed-in inset view.

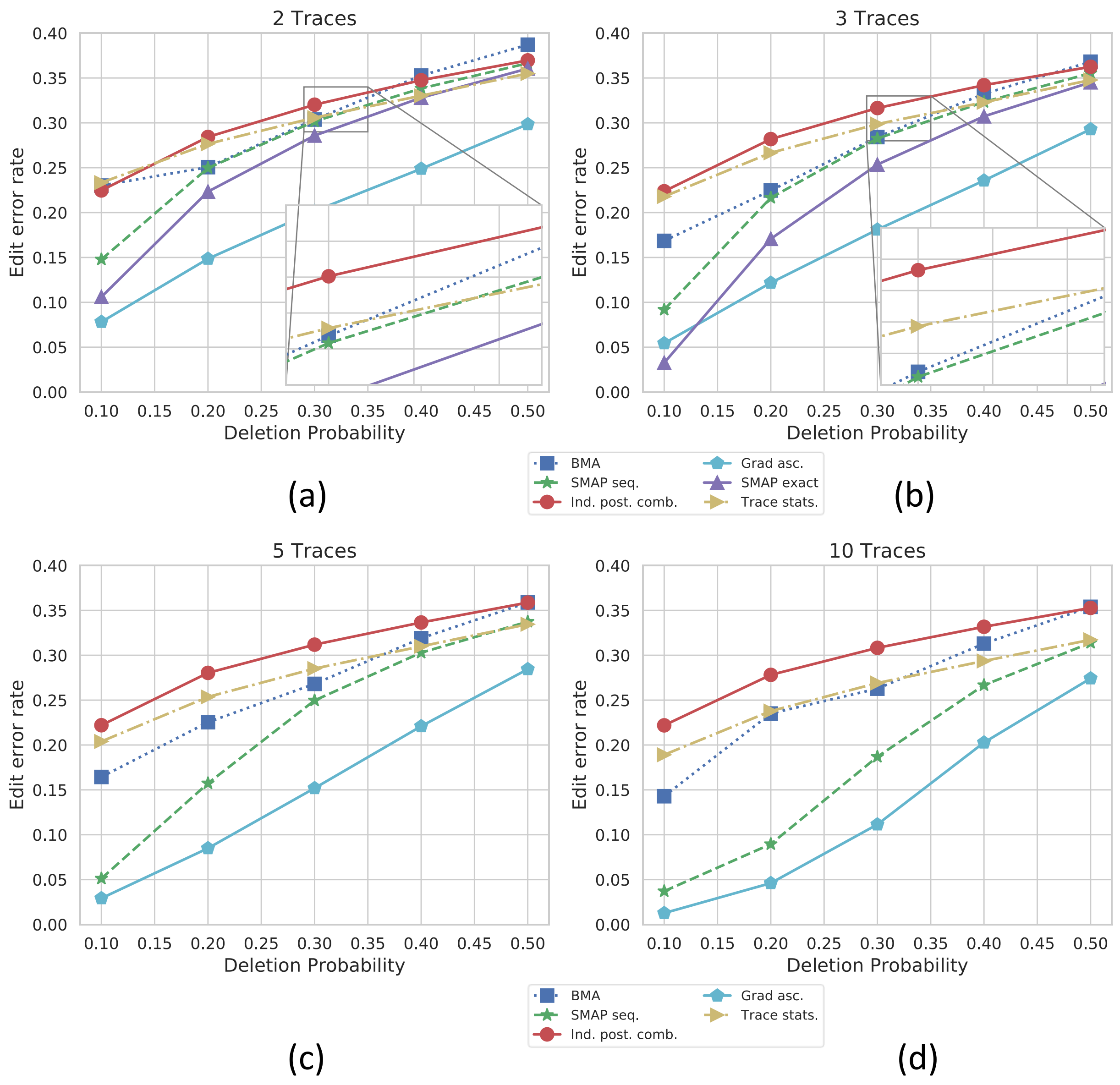


Figure 2.8: Comparison of edit error rates for a blocklength $n = 100$ illustrated with 2,3,5 and 10 observed traces. Note that we do not run SMAP exact. for 5 and 10 traces since its complexity grows exponentially with the number of traces. All the subplots are plotted on the same scale to aid comparability across subplots. Few of the subplots which contain algorithms with similar error rates also contain a zoomed-in inset view.

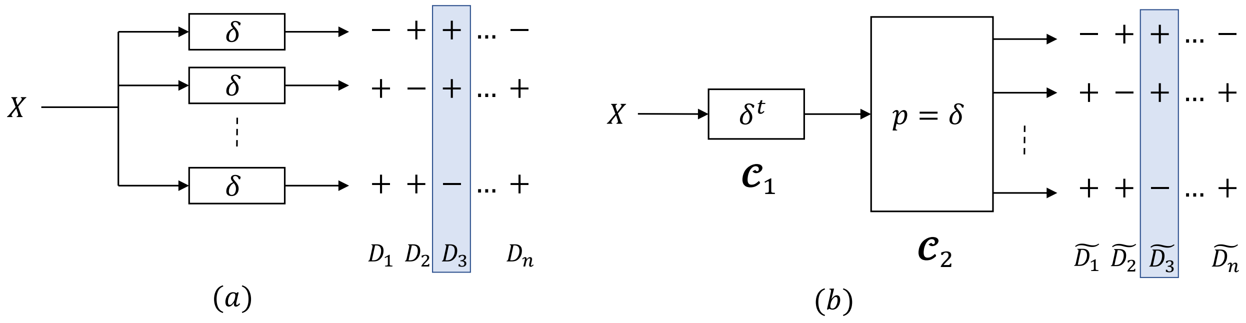
2.7 Appendix

2.7.1 Proof of Theorem 2.3

The intuition behind the theorem is that the cascade model splits the error events in the t-trace deletion channel into 2 parts:

- When an input symbol is deleted in all the traces, which is captured by the deletion channel with parameter δ^t.

- When an input symbol is not deleted in at least one of the traces, captured by the remnant channel.



X
δ
δ
δ
$- \; + \; + \; \ldots \; -$
$+ \; - \; + \; \ldots \; +$
$+ \; + \; - \; \ldots \; +$
D_1 D_2 D_3 D_n
(a)
X
δ^t
$\mathcal{C}_1$
$p = \delta$
$\mathcal{C}_2$
$- \; + \; + \; \ldots \; -$
$+ \; - \; + \; \ldots \; +$
$+ \; + \; - \; \ldots \; +$
$\widetilde{D_1}$ $\widetilde{D_2}$ $\widetilde{D_3}$ $\widetilde{D_n}$
(b)

Figure 2.9: The deletion error events occurring in the two channel models. Here '$-$' corresponds to a symbol being deleted and '$+$' corresponds to a transmission. The deletion pattern D_i corresponds to the input symbol X_i.

In order to prove the theorem, we need to prove that the deletion patterns arising in the t-trace channel model and in the cascade model have the same distribution, i.e.,

$$\Pr(D_1 = d_1, D_2 = d_2, ..., D_n = d_n) = \Pr(\widetilde{D}_1 = d_1, \widetilde{D}_2 = d_2, ..., \widetilde{D}_n = d_n),$$

where $d_i \in \{-, +\}^t$, where a $-$ corresponds to a deletion and a $+$ corresponds to a transmission. Also from the definition of our channel models, the deletions act independently on each input symbol i.e., D_i is independent ofD_j for $i \neq j$. So it is sufficient to prove that the distributions of each D_i and $\widetilde{D}_i$ are the same.

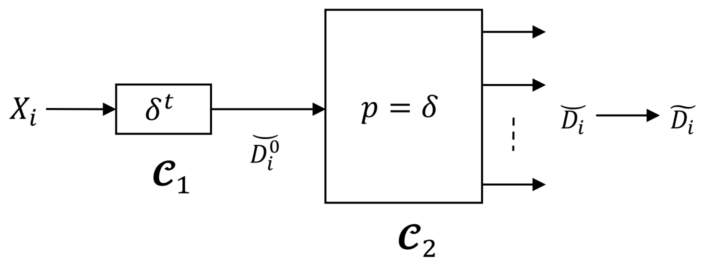


Figure 2.10: The error events of the cascade model, expressed in terms of the error events of its components.

Consider $\widetilde{D}_i$ – this is influenced by $\breve{D}_i^0$ which is the deletion in channel $\mathcal{C}_1$ and by $\breve{D}_i$ which are the deletion in the remnant channel $\mathcal{C}_2$. To prove the equivalence, we consider 2 cases:

- $d_i = (-,-,-,...,-)$, the error event where a symbol is deleted in all the observations. It can be seen that $\Pr(D_i = d_i)$ for this case is δ^t. On the other hand, to compute $\Pr(\widetilde{D}_i = d_i)$, we note that this event is possible if and only if $\breve{D}_i^0 = -$, since by definition, the remnant channel cannot delete the input symbol in all the t observations. Therefore, $\Pr(\widetilde{D}_i = d_i) = \Pr(\breve{D}_i^0 = -) = \delta^t$.

- $d_i \neq (-,-,-,...,-)$, i.e., the input symbol is not deleted in at least one trace. Also let us define k to be the count of $-$ in d_i. In this case, $\Pr(D_i = d_i) = \delta^{\text{Count(-) in } d_i}(1-\delta)^{\text{Count(+) in } d_i} = \delta^k(1-\delta)^{t-k}$. For the cascade model, this event requires that $\breve{D}_i^0 = +$ and $\breve{D}_i = d_i$. Thus,

$$\Pr(\tilde{D}_i = d_i) = \Pr(\breve{D}_i^0 = +) \cdot \Pr(\breve{D}_i = d_i) = (1-\delta^t)\frac{\delta^k(1-\delta)^{t-k}}{1-\delta^t} = \delta^k(1-\delta)^{t-k}.$$

In both cases, the distributions of D_i and $\widetilde{D}_i$ are the same, proving the equivalence.

2.7.2 Proof of Lemma 2.1

Lemma 2.1. *For $p = (p_1, p_2, .., p_i, ..., p_n)$ and $Y = y = y_1 ... y_m$ with $n \geq m > 0$, we have*

$$\mathbf{F}(p, y) = \mathbf{F}(p_{[n]\setminus\{i\}}, y) + p_i \sum_{k|y_k=1} \mathbf{F}(p_{[1:i-1]}, y_{[1:k-1]})\mathbf{F}(p_{[i+1:n]}, y_{[k+1,m]})$$
$$+(1-p_i) \sum_{k|y_k=0} \mathbf{F}(p_{[1:i-1]}, y_{[1:k-1]})\mathbf{F}(p_{[i+1:n]}, y_{[k+1,m]}).$$

Proof. The proof of this lemma uses a similar approach as the proof of Thm. 2.6. First, in the expression for $\mathbf{F}(\cdot)$, we separate out the subsets that contain index i:

$$\begin{aligned}
\mathbf{F}(p, y) &= \sum_{\substack{\mathcal{S}|\mathcal{S}\subseteq[n], \\ |\mathcal{S}|=m}} \prod_{j=1}^{m} p_{\mathcal{S}_j}^{y_j}(1-p_{\mathcal{S}_j})^{1-y_j} \\
&= \sum_{\substack{\mathcal{S}|\mathcal{S}\subseteq[n], \\ |\mathcal{S}|=m, \\ i\notin\mathcal{S}}} \prod_{j=1}^{m} p_{\mathcal{S}_j}^{y_j}(1-p_{\mathcal{S}_j})^{1-y_j} + \sum_{\substack{\mathcal{S}|\mathcal{S}\subseteq[n], \\ |\mathcal{S}|=m, \\ i\in\mathcal{S}}} \prod_{j=1}^{m} p_{\mathcal{S}_j}^{y_j}(1-p_{\mathcal{S}_j})^{1-y_j} \\
&= \mathbf{F}(p_{[n]\setminus\{i\}}, y) + \sum_{\substack{\mathcal{S}|\mathcal{S}\subseteq[n], \\ |\mathcal{S}|=m, \\ i\in\mathcal{S}}} \prod_{j=1}^{m} p_{\mathcal{S}_j}^{y_j}(1-p_{\mathcal{S}_j})^{1-y_j}. \qquad (2.20)
\end{aligned}$$

Now the second term can be further split as,

$$\sum_{\substack{\mathcal{S}|\mathcal{S}\subseteq[n], \\ |\mathcal{S}|=m, \\ i\in\mathcal{S}}} \prod_{j=1}^{m} p_{\mathcal{S}_j}^{y_j}(1-p_{\mathcal{S}_j})^{1-y_j} = \sum_{k=1}^{m} \sum_{\substack{\mathcal{S}|\mathcal{S}\subseteq[n], \\ |\mathcal{S}|=m, \\ \mathcal{S}_k=i}} \prod_{j=1}^{m} p_{\mathcal{S}_j}^{y_j}(1-p_{\mathcal{S}_j})^{1-y_j}.$$

One could express the set $\mathcal{S}$ as the union $\mathcal{S} = \mathcal{S}' \cup \{i\} \cup \mathcal{S}''$ such that $\mathcal{S}' \subseteq [i-1]$ and $\mathcal{S}'' \subseteq [i+1:n]$ to get

$$\begin{aligned}
&\sum_{k=1}^{m} \sum_{\substack{\mathcal{S}|\mathcal{S}\subseteq[n], \\ |\mathcal{S}|=m, \\ \mathcal{S}_k=i}} \prod_{j=1}^{m} p_{\mathcal{S}_j}^{y_j}(1-p_{\mathcal{S}_j})^{1-y_j} \\
&= \sum_{k=1}^{m} \sum_{\substack{\mathcal{S}'| \\ \mathcal{S}'\subseteq[i-1] \\ |\mathcal{S}'|=k-1}} \sum_{\substack{\mathcal{S}''| \\ \mathcal{S}''\subseteq[i+1:n] \\ |\mathcal{S}''|=m-k}} \left(\prod_{j=1}^{k-1} p_{\mathcal{S}'_j}^{y_j}(1-p_{\mathcal{S}'_j})^{1-y_j} \right) \left(p_i^{y_k}(1-p_i)^{1-y_k} \right) \left(\prod_{j=1}^{m-k} p_{\mathcal{S}''_j}^{y_{j+k}}(1-p_{\mathcal{S}''_j})^{1-y_{j+k}} \right)
\end{aligned}$$

$$= \sum_{k=1}^{m} p_i^{y_k}(1-p_i)^{1-y_k} \left(\sum_{\substack{\mathcal{S}'| \\ \mathcal{S}'\subseteq[i-1] \\ |\mathcal{S}'|=k-1}} \prod_{j=1}^{k-1} p_{\mathcal{S}'_j}^{y_j}(1-p_{\mathcal{S}'_j})^{1-y_j} \right) \left(\sum_{\substack{\mathcal{S}''| \\ \mathcal{S}''\subseteq[i+1:n] \\ |\mathcal{S}''|=m-k}} \prod_{j=1}^{m-k} p_{\mathcal{S}''_j}^{y_{j+k}}(1-p_{\mathcal{S}''_j})^{1-y_{j+k}} \right)$$

$$= \sum_{k=1}^{m} p_i^{y_k}(1-p_i)^{1-y_k} \mathbf{F}(p_{[i-1]}, y_{[k-1]}) \mathbf{F}(p_{[i+1:n]}, y_{[k+1:m]}).$$

The $\sum_{k=1}^{m}$ summation in the above expression could further be split into the two cases depending on whether $y_k = 0$ or $y_k = 1$, which simplifies the term $p_i^{y_k}(1-p_i)^{1-y_k}$ to either $1-p_i$ or p_i respectively. Thus,

$$\sum_{\substack{\mathcal{S}|\mathcal{S}\subseteq[n], \\ |\mathcal{S}|=m, \\ i\in\mathcal{S}}} \prod_{j=1}^{m} p_{\mathcal{S}_j}^{y_j}(1-p_{\mathcal{S}_j})^{1-y_j}$$

$$= (1-p_i) \sum_{k|y_k=0} \mathbf{F}(p_{[i-1]}, y_{[k-1]}) \mathbf{F}(p_{[i+1:n]}, y_{[k+1:m]}) + p_i \sum_{k|y_k=1} \mathbf{F}(p_{[i-1]}, y_{[k-1]}) \mathbf{F}(p_{[i+1:n]}, y_{[k+1:m]}). \tag{2.21}$$

Plugging (2.21) in (2.20) concludes the proof of the Lemma. □

2.7.3 Proof of Lemma 2.2

The following Lemma forms the backbone of the analyses for multiple traces. This lemma is also closely related to the channel equivalence in Theorem 2.3.

Lemma 2.2. *For $h, f_1, f_2, ..., f_m \in \mathcal{A}^*$,*

$$\binom{h}{f_1}\binom{h}{f_2}...\binom{h}{f_m} = \sum_{w\in\mathcal{A}^*} \langle f_1 \uparrow f_2 \uparrow ... \uparrow f_m, w\rangle \binom{h}{w}.$$

Proof. The channel equivalence can essentially be tied to this lemma as follows: consider the two channel models in Fig. 2.3. The probability of observations given the input in both cases is proportional to the number of ways of obtaining the observations given the input.

- For the t-trace deletion channel model in Fig. 2.3 (a), the number of ways to obtain the traces given the input is equal to $\binom{X}{Y^1}\binom{X}{Y^2}...\binom{X}{Y^t}$.
- For the cascade model in Fig. 2.3 (b), the number of ways to obtain the traces given the input is equal to $\sum_z \binom{X}{z}\langle \tilde{Y}^1 \uparrow \tilde{Y}^2 \uparrow ... \uparrow \tilde{Y}^t, z\rangle$, which we show below.

The above two are expression must be equal since the two channel models are equivalent.

We now first compute the probability of a given set of output sequences given an input sequence for the remnant channel, namely $\Pr(\tilde{Y}^1, \tilde{Y}^2, ..., \tilde{Y}^t|Z)$. First, note that there can be multiple deletion patterns corresponding to outputs $\tilde{Y}^1, \tilde{Y}^2, ..., \tilde{Y}^t$ resulting from a given input Z. The number of such patterns is equal to $\langle \tilde{Y}^1 \uparrow \tilde{Y}^2 \uparrow ... \uparrow \tilde{Y}^t, Z\rangle$, which essentially follows from the definition of the infiltration product. Consider one such valid deletion pattern, i.e., a deletion pattern $\mathcal{D}$ that is a mapping of the symbols in Z onto the symbols in $\tilde{Y}^1, \tilde{Y}^2, ..., \tilde{Y}^t$: $\mathcal{D} = \{(1, S_1), (2, S_2), ..., (|Z|, S_{|Z|})\}$. Here (i, S_i) represents the fact that Z_i is not deleted in the output set $\tilde{Y}^{S_i}$ and is deleted in the rest. From the definition of the remnant channel, we have $|S_i| > 0$. Also $\sum_{i=1}^{|Z|} |S_i| = \sum_{j=1}^{t} |\tilde{Y}^j|$ since every symbol of each output is associated with exactly one input symbol and hence corresponds to one particular S_i. Thus,

$$\begin{aligned}
\Pr(\tilde{Y}^1, \tilde{Y}^2, ..., \tilde{Y}^t|Z) &= \langle \tilde{Y}^1 \uparrow \tilde{Y}^2 \uparrow ... \uparrow \tilde{Y}^t, Z\rangle \Pr(\tilde{Y}^1, \tilde{Y}^2, ..., \tilde{Y}^t|Z, \mathcal{D}) \\
&= \langle \tilde{Y}^1 \uparrow \tilde{Y}^2 \uparrow ... \uparrow \tilde{Y}^t, Z\rangle \prod_{i=1}^{|Z|} \frac{(1-\delta)^{|S_i|}\delta^{t-|S_i|}}{1-\delta^t} \\
&= \langle \tilde{Y}^1 \uparrow \tilde{Y}^2 \uparrow ... \uparrow \tilde{Y}^t, Z\rangle \frac{(1-\delta)^{\sum |S_i|}\delta^{|Z|t-\sum |S_i|}}{(1-\delta^t)^{|Z|}} \\
&= \langle \tilde{Y}^1 \uparrow \tilde{Y}^2 \uparrow ... \uparrow \tilde{Y}^t, Z\rangle \frac{(1-\delta)^{\sum |\tilde{Y}^j|}\delta^{|Z|t-\sum |\tilde{Y}^j|}}{(1-\delta^t)^{|Z|}}.
\end{aligned}$$

We can then compute the probability of the output given the input for the cascade channel as

$$\Pr(\tilde{Y}^1, \tilde{Y}^2, ..., \tilde{Y}^t|X)$$

$$
\begin{aligned}
&= \sum_z \Pr(\tilde{Y}^1, \tilde{Y}^2, ..., \tilde{Y}^t, Z = z|X) \\
&= \sum_z \Pr(Z = z|X) \Pr(\tilde{Y}^1, \tilde{Y}^2, ..., \tilde{Y}^t|Z = z) \\
&= \sum_z \left[\binom{X}{z} \delta^{t(|X|-|z|)}(1-\delta^t)^{|z|} \langle \tilde{Y}^1 \uparrow \tilde{Y}^2 \uparrow ... \uparrow \tilde{Y}^t, z \rangle \frac{(1-\delta)^{\sum |\tilde{Y}^j|}\delta^{|z|t-\sum |\tilde{Y}^j|}}{(1-\delta^t)^{|z|}} \right] \\
&= \left[\sum_z \binom{X}{z} \langle \tilde{Y}^1 \uparrow \tilde{Y}^2 \uparrow ... \uparrow \tilde{Y}^t, z \rangle \right] \delta^{t|X|-\sum |\tilde{Y}^j|}(1-\delta)^{\sum |\tilde{Y}^j|}. \qquad (2.22)
\end{aligned}
$$

For the t-trace deletion channel model, we have:

$$
\begin{aligned}
\Pr(Y^1, Y^2, ..., Y^t|X) &= \prod_{j=1}^t \binom{X}{Y^j} \delta^{|X|-|Y^j|}(1-\delta)^{|Y_j|} \\
&= \binom{X}{Y^1}\binom{X}{Y^2}...\binom{X}{Y^t} \delta^{t|X|-\sum |Y^j|}(1-\delta)^{\sum |Y^j|}. \qquad (2.23)
\end{aligned}
$$

Equating (2.22) and (2.23) with $X = h$ and traces as $Y^j = \tilde{Y}^j = f_j$ proves the Lemma.

Alternatively, we use also induction to prove the statement as we do below. The statement is trivially true when $m = 1$ since, $\sum_w \binom{h}{w}\langle f_1, w\rangle = \binom{h}{f_1}$ as $\langle f, w\rangle = \mathbb{1}_{f=w}$. We refer the reader to equation 6.3.25 in [Lot97] for the proof of the lemma for the case $m = 2$. Assume that the statement is true for $m = k \in \mathbb{Z}, k \geq 2$. We next prove the validity when $m = k+1$. Consider

$$
\begin{aligned}
\binom{h}{f_1}\binom{h}{f_2}...\binom{h}{f_k}\binom{h}{f_{k+1}} &= \sum_w \binom{h}{w} \langle f_1 \uparrow f_2 \uparrow ... \uparrow f_k, w\rangle \binom{h}{f_{k+1}} \\
&= \sum_w \left[\binom{h}{w}\binom{h}{f_{k+1}} \right] \langle f_1 \uparrow f_2 \uparrow ... \uparrow f_k, w\rangle \\
&= \sum_w \left[\sum_v \langle w \uparrow f_{k+1}, v\rangle \binom{h}{v} \right] \langle f_1 \uparrow f_2 \uparrow ... \uparrow f_k, w\rangle \\
&= \sum_v \binom{h}{v} \left[\sum_w \langle w \uparrow f_{k+1}, v\rangle \langle f_1 \uparrow f_2 \uparrow ... \uparrow f_k, w\rangle \right]. \qquad (2.24)
\end{aligned}
$$

To evaluate the term in the square bracket, we use (2.34). For the case where $\tau \in \mathcal{A}^*, \sigma \in$

$\mathbb{Z}\langle\mathcal{A}\rangle$ in (2.34), we have

$$\sigma \uparrow \tau = \sum_{f\in\mathcal{A}^*} \langle \sigma, f\rangle (f \uparrow \tau),$$

and thus

$$\langle \sigma \uparrow \tau, u\rangle = \sum_{f\in\mathcal{A}^*} \langle \sigma, f\rangle \langle f \uparrow \tau, u\rangle. \tag{2.25}$$

We use (2.25) to replace the term in the square bracket in (2.24), i.e.,

$$\begin{aligned}&\binom{h}{f_1}\binom{h}{f_2}\cdots\binom{h}{f_k}\binom{h}{f_{k+1}}\\ &= \sum_v \binom{h}{v} \langle (f_1 \uparrow f_2 \uparrow \ldots \uparrow f_k) \uparrow f_{k+1}, v\rangle,\end{aligned} \tag{2.26}$$

and the lemma follows from the associativity property of the infiltration product. □

2.7.4 Proof of Lemma 2.3

Lemma 2.3.

$$\sum_{\substack{f||f|=n\\ f_i=a}} \binom{f}{g} = 2^{n-|g|}\left(\frac{1}{2}\binom{n-1}{|g|} + \sum_{j|g_j=a} \binom{i-1}{j-1}\binom{n-i}{|g|-j}\right),$$

where $j \in \left[\max\{1, |g|+i-n\} : \min\{i, |g|\}\right]$.

Proof. First, observe that

$$\binom{f}{g} = \sum_{\substack{S\subseteq[n]:\\ |S|=|g|}} \mathbb{1}_{f_S=g},$$

where the summation is over all ordered subsets of $[n] = \{1, 2, \ldots, n\}$ of size $|g|$ and f_S corresponds to the subsequence of f indexed by S. Thus,

$$\sum_{\substack{f\in\mathcal{A}^n|\\ f_i=a}} \binom{f}{g} = \sum_{\substack{f\in\mathcal{A}^n|\\ f_i=a}} \sum_{\substack{S\subseteq[n]|\\ |S|=|g|}} \mathbb{1}_{f_S=g} = \sum_{\substack{S\subseteq[n]|\\ |S|=|g|}} \sum_{\substack{f\in\mathcal{A}^n|\\ f_i=a}} \mathbb{1}_{f_S=g}$$

$$= \sum_{\substack{S\subseteq[n]| \\ |S|=|g| \\ i\notin S}} \sum_{\substack{f\in\mathcal{A}^n| \\ f_i=a}} \mathbb{1}_{f_S=g} + \sum_{\substack{S\subseteq[n]| \\ |S|=|g| \\ i\in S}} \sum_{\substack{f\in\mathcal{A}^n| \\ f_i=a}} \mathbb{1}_{f_S=g}$$

$$= \sum_{\substack{S\subseteq[n]| \\ |S|=|g| \\ i\notin S}} \sum_{\substack{f\in\mathcal{A}^n| \\ f_i=a}} \mathbb{1}_{f_S=g} + \sum_{j=1}^{m} \sum_{\substack{S\subseteq[n]| \\ |S|=|g| \\ S_j=i}} \sum_{\substack{f\in\mathcal{A}^n| \\ f_i=a}} \mathbb{1}_{f_S=g}. \tag{2.27}$$

The two terms in (2.27) can be visualized as the number of ways to fill up the blank spaces

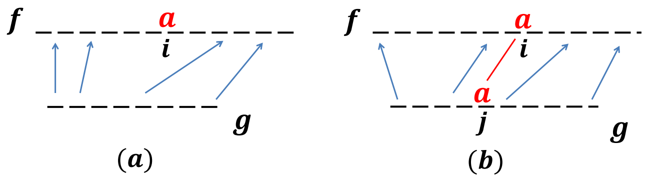


Figure 2.11: Figure illustrating proof of Lemma 2.3.

(spaces without arrows pointing to it in f) in Fig. 2.11(a) and (b) respectively. Solving this counting problem, we get

$$\sum_{\substack{f\in\mathcal{A}^n| \\ f_i=a}} \binom{f}{g} = 2^{n-|g|}\left(\frac{1}{2}\binom{n-1}{|g|} + \sum_{j|g_j=a} \binom{i-1}{j-1}\binom{n-i}{|g|-j}\right).$$

$\square$

2.7.5 Computation of $\mathbf{F}(p,v)$

We here describe how to compute $\mathbf{F}(p,v)$ in $O(mn)$ time and space complexity, where $p = (p_1,...,p_n)$ and $v = v_1...v_m$, via a dynamic programming approach. Note that $m \le n$ otherwise $\mathbf{F}(p,v) = 0$. We first define

$$\mathbf{G}^{for}(k,j) \triangleq \mathbf{F}(p_{[1:k]}, v_{[1:j]}). \tag{2.28}$$

Using Lemma 2.1 with $i = n$, we get

$$\mathbf{F}(p,v) = \mathbf{F}(p_{[n-1]}, v) + p_n^{v_m}(1-p_n)^{(1-v_m)}\mathbf{F}(p_{[n-1]}, v_{[m-1]}).$$

This translates to the following dynamic program for $\mathbf{G}^{for}$:

$$\mathbf{G}^{for}(k,j) = \mathbf{G}^{for}(k-1,j) + p_k^{v_j}(1-p_k)^{1-v_j}\mathbf{G}^{for}(k-1,j-1), \tag{2.29}$$

with the boundary conditions $\mathbf{G}^{for}(k,0) = 1 \; \forall \; k \geq 0$ and $\mathbf{G}^{for}(k,j) = 0 \; \forall \; k < j$. The algorithm is now summarized as Alg. 2.11.

Algorithm 2.11 Computing $\mathbf{F}(p,v)$

```
1: Inputs: $p \in [0,1]^n$, $v \in \{0,1\}^m$
2: Outputs: $\mathbf{F}(p_{[1:k]}, v_{[1:j]})$ for all $k \in [n]$ and $j \in [m]$
3: Initialize $\mathbf{G}^{for}(k,0) = 1 \; \forall \; k$ and $\mathbf{G}^{for}(k,j) = 0 \; \forall \; k < j$
4: for $k = 1 : n$ and $j = 1 : m$ do
5:     Use (2.29) to update $\mathbf{G}^{for}(k,j)$
6: return $\mathbf{G}^{for}(k,j) \; \forall \; k,j$
```

We note that a similar dynamic programming approach yields $\mathbf{F}(p_{[k+1:n]}, v_{[j+1:m]})$ for all $k \in [n]$ and $j \in [m]$ in $O(mn)$ time and space complexity by defining

$$\mathbf{G}^{rev}(k,j) \triangleq \mathbf{F}(p_{[k+1:n]}, v_{[j+1:m]}).$$

The following dynamic program can be used for $\mathbf{G}^{rev}$:

$$\mathbf{G}^{rev}(k,j) = \mathbf{G}^{rev}(k+1,j) + p_{k+1}^{v_{j+1}}(1-p_{k+1})^{1-v_{j+1}}\mathbf{G}^{rev}(k+1,j+1), \tag{2.30}$$

with the boundary conditions $\mathbf{G}^{rev}(k,m) = 1 \, \forall \, k \geq 0$ and $\mathbf{G}^{rev}(k,j) = 0 \, \forall \, k,j : n-k < m-j$.

2.7.6 Computation of $\nabla_p \mathbf{F}(p,v)$

First, from Lemma 2.1, we have

$$\mathbf{F}(p,v) = \mathbf{F}(p_{[n]\setminus\{i\}}, v) + (1-p_i) \sum_{k|v_k=0} \mathbf{F}(p_{[i-1]}, v_{[k-1]})\mathbf{F}(p_{[i+1:n]}, v_{[k+1:m]})$$

$$+ p_i \sum_{k|v_k=1} \mathbf{F}(p_{[i-1]}, v_{[k-1]})\mathbf{F}(p_{[i+1:n]}, v_{[k+1:m]}).$$

Differentiating with respect to p_i, we get

$$\begin{aligned}\frac{\partial \mathbf{F}(p,v)}{\partial p_i} &= \sum_{k|v_k=1} \mathbf{F}(p_{[i-1]}, v_{[k-1]})\mathbf{F}(p_{[i+1:n]}, v_{[k+1:m]}) - \sum_{k|v_k=0} \mathbf{F}(p_{[i-1]}, v_{[k-1]})\mathbf{F}(p_{[i+1:n]}, v_{[k+1:m]}) \\ &= \sum_{k|v_k=1} \mathbf{G}^{for}(i-1,k-1)\mathbf{G}^{rev}(i,k) - \sum_{k|v_k=0} \mathbf{G}^{for}(i-1,k-1)\mathbf{G}^{rev}(i,k). \qquad (2.31)\end{aligned}$$

Thus, computing the $\mathbf{G}^{for}$ and $\mathbf{G}^{rev}$ terms is sufficient to compute the gradient. As discussed above, this computation requires $O(nm)$ operations. Given $\mathbf{G}^{for}$ and $\mathbf{G}^{rev}$, the computation of each partial derivative $\frac{\partial \mathbf{F}(p,v)}{\partial p_i}$ requires $O(m)$ operations, and we need to compute n such partial derivatives. Thus, the complexity of computing $\nabla_p \mathbf{F}(p,v)$ can be done in $O(nm)$ time and space complexity.

Algorithm 2.12 Computing $\nabla_p \mathbf{F}(p,v)$

```
1: Inputs: $p \in [0,1]^n$, $v \in \{0,1\}^m$
2: Outputs: $\nabla_p \mathbf{F}(p,v)$
3: Initialize $\mathbf{G}^{for}(k,0) = 1 \ \forall\ k$ and $\mathbf{G}^{for}(k,j) = 0 \ \forall\ k < j$
4: Initialize $\mathbf{G}^{rev}(k,m) = 1 \ \forall\ k$ and $\mathbf{G}^{rev}(k,j) = 0 \ \forall\ k,j : n-k < m-j$
5: for $k = 1 : n$ and $j = 1 : m$ do
6:     Use (2.29) and (2.30) to compute $\mathbf{G}^{for}(k,j)$ and $\mathbf{G}^{rev}(k,j)$
7: for $i = 1 : n$ do
8:     Use (2.31) to compute $\frac{\partial \mathbf{F}(p,v)}{\partial p_i}$
9: return $\nabla_p \mathbf{F}(p,v)$
```

2.7.7 An algebraic definition of the infiltration product.

For completeness, we reproduce the formal definition of the infiltration product from Section 6.3 of [Lot97] (also see there for the equivalence of the two definitions). A *formal series* with indeterminates (or variables) in a set $\mathcal{A}$ and coefficients in a commutative ring $\mathcal{R}$, is a mapping of $\mathcal{A}^*$ onto $\mathcal{R}$. Recall that a commutative ring is a set which forms an abelian group under an *addition* operation, is a monoid under a *multiplication* operation which commutes, and the multiplication operation distributes over addition. Here we consider $\mathbb{Z}$, the set of integers as the commutative ring $\mathcal{R}$. A formal series is called a *polynomial* if only a finite number of sequences are mapped to non-zero values, the rest of the sequences map to zero. Consider two polynomials $\sigma, \tau : \mathcal{A}^* \to \mathbb{Z}$. The value taken by a sequence $w \in \mathcal{A}^*$ on σ (or the coefficient of w in σ) is denoted by $\langle \sigma, w \rangle \in \mathbb{R}$. We also define binary addition ($\oplus$) and multiplication operations ($\times$) on the set of polynomials as follows:

$$\langle \sigma \oplus \tau, w \rangle \triangleq \langle \sigma, w \rangle + \langle \tau, w \rangle \quad \forall w \in \mathcal{A}^*, \tag{2.32}$$

$$\langle \sigma \times \tau, w \rangle \triangleq \sum_{\substack{f,g \in \mathcal{A}^*: \\ f.g = w}} \langle \sigma, f \rangle \langle \tau, g \rangle \quad \forall w \in \mathcal{A}^*. \tag{2.33}$$

We will use the usual symbols + and . in place of $\oplus$ and $\times$ in this work for convenience. The meaning of the operation would be clear depending on the operands. With these operations the set of polynomials form a non-commutative ring, and is denoted by $\mathbb{Z}\langle \mathcal{A} \rangle$, also called the free $\mathbb{Z}$-algebra on $\mathcal{A}$ in ring theory. Note that the addition and multiplication operations defined in (2.32) and (2.33) are similar to the operations defined on commutative polynomials, except that the multiplication operation under the summation in (2.33) ($f.g = w$) is actually concatenation and is non-commutative. The multiplication inside the summation in (2.33) is multiplication in the real field and hence commutative. The multiplication defined in (2.33) distributes over addition defined in (2.32). Thus, a polynomial in $\mathbb{Z}\langle \mathcal{A} \rangle$ can be represented as a sum of monomials in $\mathcal{A}^*$ each with an associated coefficient in $\mathbb{Z}$, i.e., $\sigma = \sum_{w \in \mathcal{A}^*} \langle \sigma, w \rangle w$. Define the *degree* of a polynomial to be equal to the length of a longest sequence with a non-zero coefficient in the polynomial and the *number of terms* of a polynomial as the number of sequences with non-zero coefficients in the polynomial. Note that a degree d polynomial could have a number of terms upto $2^{d+1} - 1$.

With this, the *infiltration product* (in general, for two polynomials) is defined as follows:

$$\forall f \in \mathcal{A}^*, \quad f \uparrow e = e \uparrow f = f.$$

$$\forall f, g \in \mathcal{A}^*, \quad \forall a, b \in \mathcal{A},$$

$$fa \uparrow gb = (f \uparrow gb)a + (fa \uparrow g)b + \mathbb{1}_{a=b}(f \uparrow g)a.$$

$$\forall \sigma, \tau \in \mathbb{Z}\langle\mathcal{A}\rangle, \quad \sigma \uparrow \tau = \sum_{f,g \in \mathcal{A}^*} \langle\sigma, f\rangle\langle\tau, g\rangle(f \uparrow g). \tag{2.34}$$

2.7.8 Symbolwise posterior probabilities for the remnant channel

Consider the remnant channel shown below, and let $Z = Z_1Z_2...Z_n$. Also let $Z_i \sim \text{Ber}(0.5)$. We aim to compute $\Pr(Z_i = 1|\tilde{Y}^1 = y^1, \tilde{Y}^2 = y^2, ..., \tilde{Y}^t = y^t)$. From the definition of the

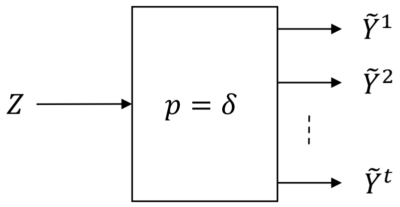


Figure 2.12: The remnant channel

infiltration product, the input-output relation for this channel can be derived to be:

$$\Pr(\tilde{Y}^1 = y^1, \tilde{Y}^2 = y^2, ..., \tilde{Y}^t = y^t|Z) = \langle y^1 \uparrow y^2 \uparrow ... \uparrow y^t, Z\rangle \frac{(1-\delta)^{\sum|y^j|}\delta^{nt-\sum|y^j|}}{(1-\delta^t)^n}.$$

Now, one could write the symbolwise posterior probabilities for Z as:

$$\Pr(Z_i = 1|\tilde{Y}^1 = y^1, \tilde{Y}^2 = y^2, ..., \tilde{Y}^t = y^t) = \sum_{\substack{z||z|=n,\\ z_i=1}} \Pr(z|\tilde{Y}^1 = y^1, \tilde{Y}^2 = y^2, ..., \tilde{Y}^t = y^t)$$

$$= \frac{1}{2^n \Pr(\tilde{Y}^1 = y^1, \tilde{Y}^2 = y^2, ..., \tilde{Y}^t = y^t)} \sum_{\substack{z||z|=n,\\ z_i=1}} \Pr(\tilde{Y}^1 = y^1, \tilde{Y}^2 = y^2, ..., \tilde{Y}^t = y^t|z)$$

$$= \frac{(1-\delta)^{\sum|y^j|}\delta^{nt-\sum|y^j|}}{(1-\delta^t)^n 2^n \Pr(\tilde{Y}^1 = y^1, \tilde{Y}^2 = y^2, ..., \tilde{Y}^t = y^t)} \sum_{\substack{z||z|=n,\\ z_i=1}} \langle y^1 \uparrow y^2 \uparrow ... \uparrow y^t, z\rangle. \tag{2.35}$$

A similar expression can be obtained for the case when $Z_i = 0$ as

$$\Pr(Z_i = 0|\tilde{Y}^1 = y^1, \tilde{Y}^2 = y^2, ..., \tilde{Y}^t = y^t)$$
$$= \frac{(1-\delta)^{\sum |y^j|}\delta^{nt-\sum |y^j|}}{(1-\delta^t)^n 2^n \Pr(\tilde{Y}^1 = y^1, \tilde{Y}^2 = y^2, ..., \tilde{Y}^t = y^t)} \sum_{\substack{z||z|=n,\\ z_i=0}} \langle y^1 \uparrow y^2 \uparrow ... \uparrow y^t, z\rangle. \quad (2.36)$$

We could further simplify (2.35) and (2.36) using the fact that the expressions in (2.35) and (2.36) must sum to 1, leading us to

$$\Pr(Z_i = 1|\tilde{Y}^1 = y^1, \tilde{Y}^2 = y^2, ..., \tilde{Y}^t = y^t) = \frac{\sum\limits_{\substack{z||z|=n,\\ z_i=1}} \langle y^1 \uparrow y^2 \uparrow ... \uparrow y^t, z\rangle}{\sum\limits_{z||z|=n} \langle y^1 \uparrow y^2 \uparrow ... \uparrow y^t, z\rangle}. \quad (2.37)$$

We precisely describe the algorithm which computes the terms in (2.37) in section 2.4, by exploiting the edit graph interpretation of the infiltration product, but give a high level idea below. The complexity of such an algorithm is $O((2n)^t)$ which is equal to the number of edges in the edit graph. Note that for a fixed number of traces, this algorithm is polynomial in the blocklength as opposed to a naive approach of iterating through all the n-length sequences.

Recall that $\langle y^1 \uparrow y^2 \uparrow ... \uparrow y^t, z\rangle$ is the number of paths from origin to destination of the edit graph $\mathcal{G}(y^1, y^2, ..., y^t)$ which correspond to z. Therefore, $\sum_{z||z|=n}\langle y^1 \uparrow y^2 \uparrow ... \uparrow y^t, z\rangle$ is equal to the number of n-length paths in $\mathcal{G}(y^1, y^2, ..., y^t)$ from the origin to the destination. Note that the edit graph has no cycles, so this quantity can be efficiently computed via the following dynamic program – the number of n length paths from the origin to a vertex v is equal to the sum of the number of $n-1$ length paths from the origin to the in-neighbors of v. Such a procedure iterates over the vertex set of $\mathcal{G}(y^1, y^2, ..., y^t)$ exactly once.

The numerator term $\sum_{\substack{z||z|=n\\ z_i=1}}\langle y^1 \uparrow y^2 \uparrow ... \uparrow y^t, z\rangle$ can be interpreted in a similar way: it is equal to the number of n-length paths in $\mathcal{G}(y^1, y^2, ..., y^t)$ from the origin to the destination such that the i^{th} edge of the path corresponds to a '1'. The algorithm for this, therefore, follows a similar principle but has an extra step. For each vertex v, we compute

- the number of paths from the origin to v of length $0, 1, ..., n$,

- the number of paths from v to the destination of length $0, 1, ..., n$.

Next we iterate over all edges in $\mathcal{G}(y^1, y^2, ..., y^t)$ corresponding to a '1' and accumulate the number of n length paths which have this particular edge as its i^{th} edge. Thus, this algorithm iterates over the vertex set twice and the edge set of $\mathcal{G}(y^1, y^2, ..., y^t)$ once.

2.7.9 A heuristic for ML optimization with a single trace.

The proof of Theorem 2.4 inspires a heuristic for sequence reconstruction (see Alg. 2.13):

- Start from a given point $p = (p_1, ..., p_n) \in [0, 1]^n$.
- One round of iteration is defined as follows: fix a traversal order for the indices $\{1, 2, ..., n\}$. Traverse through the indices i in order and make p_i either 0 or 1 depending on whether $\mathbf{F}(p^{(i\to 0)}, y)$ or $\mathbf{F}(p^{(i\to 1)}, y)$ is larger. This ensures that $\mathbf{F}(p, y)$ never decreases.
- At the end of the round, check if the resultant p was already obtained at the end of a previous round: if so, end the algorithm (to prevent it from going into an endless cycle). Otherwise, start a new round from the resultant p.

The resultant p at the end of a round is a lattice point since we make each p_i to be 0 or 1. Therefore, the algorithm will end after a finite number of steps; in the worst case it will iterate through all 2^n sequences, although in practice we observe that it ends in 4-5 rounds (tested up to a blocklength of 100). We also note that the complexity of each round is $O(n^3)$ since it iterates through n coordinates and for each coordinate computes $\mathbf{F}(\cdot)$, which is $O(n^2)$.

A natural question is whether it makes a difference if Alg. 2.13 starts from an interior point ($p = (p_1, ..., p_n) \in [0, 1]^n$ where $\exists\ p_i \in (0, 1)$) as compared to starting from a lattice point (for instance, we could start from $p = (y, 0, ..., 0) \in \{0, 1\}^n$) which is the n-length sequence

Algorithm 2.13 Coordinate switch ML heuristic

```
1: Input: Blocklength $n$, Trace $Y = y$, Initial point $p = (p_1, p_2, ..., p_n)$
2: Outputs: Estimated sequence $\hat{X}$
3: Initialize visited set $\mathcal{V} = \varnothing$
4: while True do
5:     Compute $\mathcal{F}_i = |\mathbf{F}(p^{(i\to 1)}, y) - \mathbf{F}(p^{(i\to 0)}, y)|\ \forall\ i$ and let $\mathcal{F} = (\mathcal{F}_1, \mathcal{F}_2, ..., \mathcal{F}_n)$.
6:     Define the ordered list $\mathcal{S} = \texttt{argsort}(\mathcal{F})$ where $\texttt{argsort}(\mathcal{F})$ returns the index set $[n]$
   sorted by descending order of $\mathcal{F}$, i.e., $\mathcal{F}_{\mathcal{S}_1} \geq \mathcal{F}_{\mathcal{S}_2} \geq ... \geq \mathcal{F}_{\mathcal{S}_n}$.
7:     for $i \in \mathcal{S}$ (ordered traversal) do
8:         if $\mathbf{F}(p^{(i\to 1)}, y) - \mathbf{F}(p^{(i\to 0)}, y) \geq 0$ then
9:             update $p \leftarrow p^{(i\to 1)}$
10:        else
11:            update $p \leftarrow p^{(i\to 0)}$
12:    if $p \in \mathcal{V}$ then break
13:    $\mathcal{V} = \mathcal{V} \cup \{p\}$
14: return $\hat{X} = p$
```

obtained via appending y with zeros. It turns out that starting from an interior point results in better accuracy on both Hamming and edit error rate metrics, thus supporting the usefulness of our ML relaxation result.

In Fig. 2.13, we compare the performance of Coordinate switch heuristic with the other trace reconstruction heuristics in Section 2.5. We see that the coordinate switch with interior point initialization performs very similar to the true ML sequence (obtained via exhaustive search), in terms of both the Hamming error rate as well as the edit error rate. This intuitively supports the idea that this is a good heuristic for the ML optimization problem. However, at this point the heuristic is applicable for reconstruction using just a single trace and it is unclear on how to extend it to multiple traces.

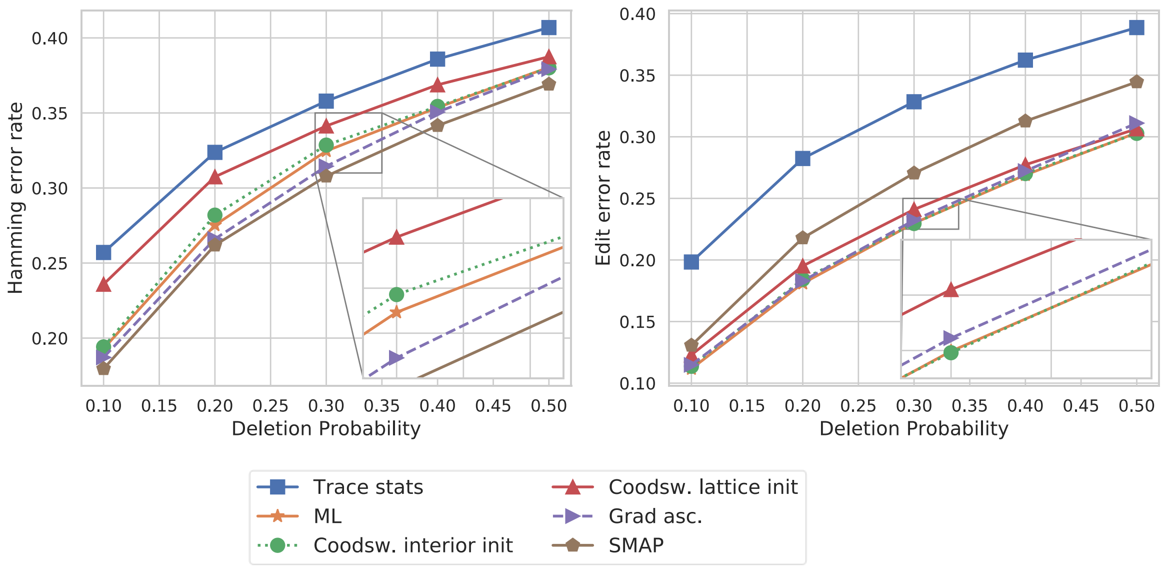


Figure 2.13: Numerics for reconstruction from a single trace for a blocklength $n = 20$. This plot compares the performance of coordinate switch heuristic (abbreviated "Coodsw. interior init." and "Coodsw. lattice init.") with other trace reconstruction algorithms from Section 2.5. "ML" refers to the true ML sequence obtained via an exhaustive search on all 20 length binary sequences. The interior point initialization initializes $p = (0.5, 0.5, ..., 0.5)$ while the lattice point initialization appends the trace y with zeros to obtain an n-length vector $p = (y, 0, ..., 0)$.

2.7.10 Symbolwise MAP as the minimizer of Hamming error rate

Symbolwise MAP is an optimal estimator for minimizing the Hamming error rate for any channel, regardless of whether it is memoryless or not. This fact can be seen from the following argument: Consider a fixed observation y (note that y here can also be a collection of multiple observations, our arguments which follow remain unchanged) and that we aim to estimate a binary input sequence X; let the estimate of the input be $\hat{X}(y)$. Note that the estimate is a function of observation y alone. Now the Hamming error rate of any estimator given y is the expectation (over all inputs) of number of symbol mismatches divided by the blocklength, i.e.,

$$\begin{aligned}\frac{1}{n}\mathbb{E}\left[\sum_{i=1}^{n}\mathbb{1}\{X_i \neq \hat{X}_i(y)\}\middle| Y=y\right] &= \frac{1}{n}\sum_{i=1}^{n}\mathbb{E}\left[\mathbb{1}\{X_i \neq \hat{X}_i(y)\}\middle| Y=y\right]\\ &= \frac{1}{n}\sum_{i=1}^{n}\Pr\left(X_i \neq \hat{X}_i(y)\middle| Y=y\right)\\ &= \frac{1}{n}\sum_{i=1}^{n}\Bigg(\Pr(X_i=0|Y=y)\Pr(\hat{X}_i(y)=1|X_i=0,Y=y)\\ &\qquad + \Pr(X_i=1|Y=y)\Pr(\hat{X}_i(y)=0|X_i=1,Y=y)\Bigg).\end{aligned}$$

But, $\hat{X}_i$ is a function of only y and hence is conditionally independent of X_i given y, which implies the following:

$$\begin{aligned}\frac{1}{n}\mathbb{E}\left[\sum_{i=1}^{n}\mathbb{1}\{X_i \neq \hat{X}_i(y)\}\middle| Y=y\right] &= \frac{1}{n}\sum_{i=1}^{n}\Bigg(\Pr(X_i=0|Y=y)\Pr(\hat{X}_i(y)=1|Y=y)\\ &\qquad + \Pr(X_i=1|Y=y)\Pr(\hat{X}_i(y)=0|Y=y)\Bigg).\end{aligned}$$

To simplify notation, let the posterior probabilities be $q_i(y) \triangleq \Pr(X_i=1|Y=y)$ and let $\alpha_i(y) \triangleq \Pr(\hat{X}_i(y)=1|Y=y)$. Note that $q_i(y)$ is a property of the channel and is fixed given y, while $\alpha_i(y)$ depends on the design of our estimator. With this, the above expression can be re-written as

$$\frac{1}{n}\mathbb{E}\left[\sum_{i=1}^{n}\mathbb{1}\{X_i \neq \hat{X}_i(y)\}\middle| Y=y\right] = \frac{1}{n}\sum_{i=1}^{n}\Bigg((1-q_i(y))\alpha_i(y) + q_i(y)(1-\alpha_i(y))\Bigg).$$

The optimal assignment of $\alpha_i(y)$ to minimize this expression is $\alpha_i(y)=1$ if $q_i(y) \geq 0.5$ and $\alpha_i(y)=0$ otherwise, which coincides with the symbolwise MAP estimate. This proves the optimality of symbolwise MAP for minimizing the Hamming error rate given any observation y, for any channel.

CHAPTER 3

ML inference as continuous optimization: deletion channels and beyond

Summary: *In this chapter, we generalize the continuous-optimization equivalence idea proved in the previous chapter to fit more general channel models. Further, we use this perspective to propose other gradient-based heuristics for trace reconstruction which improve upon algorithms in Chapter 2 in certain situations.*

The problem of estimating an unknown discrete input sequence from its noisy observation arises in many disciplines, including communications, information and coding theory. Fig. 3.1 represents a typical decoding problem – an input message sequence X must be estimated from the observation or output sequence Y. The input-output relation depends on a multitude of factors such as the exact choice and properties of the encoder, the model and parameters of the noisy channel, and initializations. We capture all these factors together by what we call the *system channel* $\mathcal{C}$. In this work, we are agnostic on what exactly happens inside $\mathcal{C}$, and instead, only assume the knowledge of the input-output relation $\Pr(Y|X, \mathcal{C})$.

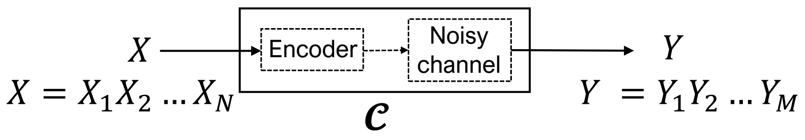


Figure 3.1: A generic model of a probabilistic system channel where each $X_i \in \mathcal{A} = \{1, ..., A\}$. The goal is to estimate X given Y and we assume the knowledge of $\Pr(Y|X, \mathcal{C})$.

Central to the above discussion lie two algorithmic problems on optimal decoding – computation of the maximum-likelihood (ML) and the symbolwise maximum-aposteriori (MAP) estimates. The ML problem, in words, is the integer program that maximizes the likelihood $\Pr(Y|X,\mathcal{C})$ over all N-length input sequences X, and the symbolwise MAP problem involves computing the symbolwise posterior probabilities (SPs) $\Pr(X_i{=}a|Y,\mathcal{C})$ for a pre-defined prior distribution on X. Many of these problems are proven to be intractable, for example through reduction to NP-complete integer optimization problems (see [BMT06], [GV05] for instance).

In this chapter, we provide continuous variable formulations for the ML and MAP decoding problems for an arbitrary system channel $\mathcal{C}$. In particular, we posit the ML problem as maximization over all product distributions for X. Rather surprisingly, this formulation closely relates to an expression for computing the MAP SPs. Our formulation is particularly useful for system channels where a function termed *expected likelihood function* (that we will define later) can be computed efficiently. Although in full generality this function would be hard to compute, it could still lead to new efficient optimal or heuristic algorithms over new classes of system channels. We believe that at the very least, our observations give a new theoretical perspective on ML and MAP decoding.

Contributions. The main result of this chapter is the formulation of the ML estimate of an arbitrary system channel as a continuous optimization problem; in particular, we optimize the *expected likelihood function* over the space of product distributions for X, instead of optimizing the actual likelihood. This opens the door to the use of first-order heuristics like gradient ascent. Moreover, we propose an alternate heuristic called *coordinate refinement* for the ML estimate. For the SPs, we give an expression in terms of the expected likelihood and its gradient. As an application, we illustrate performance benefits of our formulations via numerics for the deletion channel.

Related work. Over the past few decades, there has been significant progress towards understanding the complexity of computing optimal ML and MAP estimates (see [BMT06], [Var97], [GV05], [HB98]) as well as towards coming up with efficient algorithms/heuristics (such as Viterbi [For73], forward-backward [BCJ74], message-passing [Gal62], sphere decoding [FP85], [HV02]). These algorithms are tailored to specific classes of system channels; in contrast, our approach applies for all system channels (is agnostic to the encoder and system model).

Decoding a discrete sequence via continuous optimization methods has also been explored in [FRH83] which formulates the ML decoding problem for a linear code over a discrete memoryless channel as a continuous optimization problem, and proposes a gradient ascent heuristic to solve it. In contrast, our work applies to an arbitrary system channel, and we further we propose a decoding heuristic which empirically performs better than gradient ascent.

Reconstruction over deletion channels without the use of a codebook is closely related to the problem of *trace reconstruction* (see for example [BKK04], [HMP08], [DOS], [HPP18]). The symbolwise MAP estimate for this case has been solved (see [SDDb]). However little is known about ML estimate in this case (see [Mit09], [SDDa]).

3.1 Notation and Tools

Basic notation: Calligraphic letters refer to sets, capitalized letters correspond to either random variables or integer constants (usage will be clear with context), bold letters are used for matrices and greek letters are used to denote functions.

ML estimate. For a system channel $\mathcal{C}$ as in Fig. 3.1 where $X_i \in \mathcal{A} = \{1, 2, ..., A\}$, the ML estimate of X given observation Y is the integer program:

$$x^*_{ml} \triangleq \underset{x \in \mathcal{A}^N}{\arg\max} \; \Pr(Y|X = x, \mathcal{C}). \tag{3.1}$$

Note that there could be multiple optimal solutions to (3.1) and in such cases, it suffices to obtain just one such solution.

Notation	**Definition**
$[i:j]$	$\{i, i+1, ..., j\}$ if $j \geq i$ and $[i:j] \triangleq \varnothing$ otherwise
$[i]$	$[1:i]$
$x_{[i:j]}$	$x_i x_{i+1} ... x_j$
$\mathbf{P} \in [0,1]^{N \times A}$	matrix that parametrizes the distribution of an N-length random vector
$\mathbf{P}_i$ for matrix $\mathbf{P}$	i^{th} row of the matrix
$\mathbf{P}_{ij}$ for matrix $\mathbf{P}$	$(i,j)^{th}$ entry of the matrix
$X \sim (\mathbf{P})$	$X = X_{[1:N]}$ and X_i is *independently* distributed and $\Pr(X_i{=}a) = \mathbf{P}_{ia}$
$\mathbf{P} \odot \mathbf{Q}$ for matrices $\mathbf{P}$ and $\mathbf{Q}$	Hadamard product (element wise product)
$\mathbf{cat}(x)$ for sequence x	$\mathbf{cat}(x)$ is an $N \times A$ matrix where $\mathbf{cat}(x)_{ia}{=}1$ if $x_i{=}a$ and $\mathbf{cat}(x)_{ia}{=}0$ otherwise, i.e., $\mathbf{cat}(x)$ is the categorical representation of x.
Matrix $\mathbf{P}$ is a lattice point	$\exists\ x$ such that $\mathbf{cat}(x) = \mathbf{P}$
$\mathbf{cat}^{-1}(\mathbf{P})$	$\mathbf{cat}^{-1}(\mathbf{P}) = x$ if $\mathbf{cat}(x) = \mathbf{P}$.
$\mathbf{P}^{(j \to b)}$	$\mathbf{P}^{(j \to b)}_{ia} = \begin{cases} \mathbf{P}_{ia} & i \neq j \\ \mathbb{1}_{\{a=b\}} & i = j. \end{cases}$ $\mathbf{P}^{(j \to b)}$ modifies only the j^{th} row of $\mathbf{P}$ to be a unit vector where $\mathbf{P}_{jb} = 1$.

Table 3.1: Table of common notation.

SPs and symbolwise MAP. For the system channel $\mathcal{C}$ in Fig. 3.1 where $X_i \in \mathcal{A} = \{1, 2, ..., A\}$, let the prior input distribution be $X \sim (\mathbf{P})$. The SPs can be collected in the matrix $\mathbf{P}^{\text{post}}$:

$$\mathbf{P}^{\text{post}}_{ia} = \Pr(X_i = a | Y, \mathcal{C}).$$

Note that $\mathbf{P}^{\text{post}}$ varies with both $\mathbf{P}$ as well as Y. The SPs give a convenient way of estimating X by picking the most likely symbol at each position (the symbolwise MAP estimate).

Definition 3.1. Expected likelihood function. For the system channel model in Fig. 3.1, given an observation Y and a distribution matrix $\mathbf{P}$, we define the expected likelihood function as the expectation of the likelihood of observing Y w.r.t the distribution $X \sim (\mathbf{P})$, i.e.,

$$\lambda(\mathbf{P}, Y; \mathcal{C}) \triangleq \mathop{\mathbb{E}}_{X \sim (\mathbf{P})} \Pr(Y|X, \mathcal{C}). \tag{3.2}$$

Some properties of the expected likelihood function are:

- $0 \leq \lambda(\mathbf{P}, Y; \mathcal{C}) \leq 1$, since it is an expectation of the likelihood.
- For a lattice point $\mathbf{P}$, $\lambda(\mathbf{P}, Y; \mathcal{C}) = \Pr(Y|X = \mathbf{cat}^{-1}(\mathbf{P}), \mathcal{C})$.

3.2 ML and SPs via expected likelihood function

In this section, we discuss obtaining ML and SPs via the expected likelihood function.

3.2.1 ML via expected likelihood

Theorem 3.1. *Consider a system channel $\mathcal{C}$ as in Fig. 3.1. Assume that $X_i \in \mathcal{A} = \{1, 2, ..., A\}$. The ML estimate in* (3.1) *is equivalent to solving the following continuous optimization:*

$$\begin{aligned} \mathop{\arg\max}_{\mathbf{P} \in \mathbb{R}^{N \times A}} \quad & \lambda(\mathbf{P}, Y; \mathcal{C}) \\ s.t. \quad & \mathbf{P} \cdot \mathbf{1} = \mathbf{1} \\ & \mathbf{0} \leq \mathbf{P}. \end{aligned} \tag{3.3}$$

$\mathbf{P} \cdot \mathbf{1}$ *represents the matrix product of* $\mathbf{P}$ *with the all ones vector* $\mathbf{1}$*, and "$\leq$" represents component-wise inequality.*

Proof. The idea behind the proof is that instead of optimizing over all possible choices for X, we optimize over all possible product distributions for X. Recall that $\lambda(\mathbf{P}, Y; \mathcal{C}) \triangleq \mathbb{E}_{X\sim(\mathbf{P})} \Pr(Y|X, \mathcal{C})$. We prove the theorem by proving the following three claims:

1. For every feasible $\mathbf{P}$,
$$\lambda(\mathbf{P}, Y; \mathcal{C}) \leq \max_{x\in\mathcal{A}^N} \Pr(Y|X = x, \mathcal{C}).$$
2. Given a solution x^*_{ml} of (3.1), there exists $\mathbf{P}^*$ such that
$$\begin{aligned} \lambda(\mathbf{P}^*, Y; \mathcal{C}) &= \max_{x\in\mathcal{A}^N} \quad \Pr(Y|X = x, \mathcal{C}) \\ &= \max_{\mathbf{P}\in\mathbb{R}^{N\times A}} \quad \lambda(\mathbf{P}, Y; \mathcal{C}). \end{aligned}$$
3. Consider a $\mathbf{P}^*$ which maximizes $\lambda(\mathbf{P}, Y; \mathcal{C})$. Sample an X from $X \sim (\mathbf{P}^*)$, then X is a solution of (3.1).

Claims 1. and 2. together prove that the maximum objective values of (3.1) and (3.3) are equal, claim 2. also gives a way of obtaining a solution of (3.3) from a solution of (3.1), and claim 3. gives a way of obtaining a solution of (3.1) from (3.3).

Claim 1. is easily seen by observing that $\lambda(\mathbf{P}, Y; \mathcal{C})$ is the expectation of $\Pr(Y|X, \mathcal{C})$ w.r.t to a distribution on X defined over the set $\mathcal{A}^N$. Clearly $\lambda(\mathbf{P}, Y; \mathcal{C})$ must not exceed the maximum value taken by $\Pr(Y|X, \mathcal{C})$ over $\mathcal{A}^N$.

Claim 2. can be seen by taking $\mathbf{P}^* = \mathbf{cat}(x^*_{ml})$ i.e.,
$$\lambda(\mathbf{cat}(x^*_{ml}), Y; \mathcal{C}) = \Pr(Y|X = x^*_{ml}, \mathcal{C}).$$

To prove claim 3., we first note that $\lambda(\mathbf{P}^*, Y; \mathcal{C}) = \Pr(Y|X{=}x^*_{ml})$ from claims 1) and 2). But $\lambda(\mathbf{P}^*, Y; \mathcal{C})$ is also the expectation of $\Pr(Y|X, \mathcal{C})$ over $X \sim (\mathbf{P}^*)$. Since $\Pr(Y|X{=}x, \mathcal{C}) \leq \Pr(Y|X{=}x^*_{ml}, \mathcal{C})\ \forall x$, we have that for every x such that $\Pr(Y|X{=}x, \mathcal{C}) > 0$ w.r.t to $X \sim (\mathbf{P}^*)$, $\Pr(Y|X{=}x, \mathcal{C}){=}\Pr(Y|X{=}x^*_{ml}, \mathcal{C})$. □

We remark that the formulation in (3.3) falls under the umbrella of signomial optimization problems (see [Xu14], [CS16] and references therein) which are, in general, hard to solve. Typical heuristic approaches to such problems involve convexification strategies that instead solve a series of related convex programs. However, such strategies would in general fail for (3.3) since, there could be exponentially many terms in (3.3) and with a change of variables, (3.3) can be written as minimization of a concave function over a convex set.

3.2.2 SPs via expected likelihood

Recall that the SPs for the model in Fig. 3.1 can be collected in the matrix $\mathbf{P}^{\text{post}}$ where $\mathbf{P}^{\text{post}}_{ia} = \Pr(X_i = a|Y, \mathcal{C})$. We first state some results that will be used to prove Theorem 3.2 and for the heuristic in Section 3.3.

Lemma 3.1. *Consider Fig. 3.1 and let the prior input distribution be $X \sim (\mathbf{P})$. Then,*

$$\sum_{x:x_i=a} \Pr(x)\Pr(Y|X = x, \mathcal{C}) = \mathbf{P}_{ia}\lambda(\mathbf{P}^{(i\to a)}, Y; \mathcal{C}).$$

The proof follows from the definition of expected likelihood and can be found in the appendix. The following two corollaries are easily seen from the definition of $\lambda(\cdot)$ and Lemma 3.1.

Corollary 1. $$\lambda(\mathbf{P}, Y; \mathcal{C}) = \sum_{a=1}^{A} \mathbf{P}_{ia}\lambda(\mathbf{P}^{(i\to a)}, Y; \mathcal{C}). \tag{3.4}$$

Corollary 2. $$\frac{\partial}{\partial \mathbf{P}_{ia}}\lambda(\mathbf{P}, Y; \mathcal{C}) = \lambda(\mathbf{P}^{(i\to a)}, Y; \mathcal{C}). \tag{3.5}$$

(3.4) indicates an important property about the geometry of the expected likelihood function – it is linear in each $\mathbf{P}_i$ (however it is not linear in $\mathbf{P}$) and (3.5) relates $\lambda(\cdot)$ and its gradient.

Theorem 3.2. *In Fig. 3.1, let the prior distribution be $X \sim (\mathbf{P})$. Then the SPs $\mathbf{P}^{post}$ can be written as:*

$$\mathbf{P}^{post}_{ia} = \mathbf{P}_{ia}\frac{\lambda(\mathbf{P}^{(i\to a)}, Y; \mathcal{C})}{\lambda(\mathbf{P}, Y; \mathcal{C})} = \mathbf{P}_{ia}\frac{\frac{\partial}{\partial \mathbf{P}_{ia}}\lambda(\mathbf{P}, Y; \mathcal{C})}{\lambda(\mathbf{P}, Y; \mathcal{C})}. \tag{3.6}$$

Alternatively a matrix formulation for the SPs is,

$$\mathbf{P}^{post} = \frac{\mathbf{P} \odot \nabla_{\mathbf{P}} \lambda(\mathbf{P}, Y; \mathcal{C})}{\lambda(\mathbf{P}, Y; \mathcal{C})}. \tag{3.7}$$

Proof. First we note that the SPs can be written as,

$$\begin{aligned} \mathbf{P}_{ia}^{\text{post}} &= \frac{\Pr(X_i = a, Y|\mathcal{C})}{\Pr(Y|\mathcal{C})} \\ &= \frac{1}{\lambda(\mathbf{P}, Y; \mathcal{C})} \sum_{x:x_i=a} \Pr(x) \Pr(Y|X = x, \mathcal{C}). \end{aligned} \tag{3.8}$$

Using Lemma 3.1 and (3.5) with (3.8) concludes the proof. □

3.3 Coordinate refinement: A global ML heuristic based on expected likelihood

In this section, we propose a heuristic for ML based on our theoretical observations in Section 3.2. But first, we discuss some classes of system channels where our expected-likelihood formulation of ML and SPs can be useful.

3.3.1 Algorithmic aspects of expected likelihood

Clearly, Theorem 3.1 and Theorem 3.2 are directly applicable for system channels where the expected likelihood can be computed efficiently. For such cases, we observe the following:

- First we observe that (3.5) implies that computing the gradient of $\lambda(\cdot)$ amounts to NA computations of $\lambda(\cdot)$. However, in many cases it might be possible to compute the gradients directly and faster (we do this for deletion channels). Moreover, Theorem 3.2 signifies that in such cases, the MAP SPs can be computed in polynomial time.
- Existence of a polynomial time algorithm to compute $\lambda(\cdot)$ does not necessarily imply that the ML problem in (3.3) is solvable in polynomial time. However, what it indicates is that heuristics for continuous optimization can be employed for (3.3).

- A natural first-order heuristic for (3.3) is *projected gradient ascent*, which is a variant of gradient ascent for maximization with constraints. In our case the constraint is that $\mathbf{P}$ must be a valid distribution matrix, i.e., $\mathbf{P}$ lies in the polytope

  $$\mathcal{D} \triangleq \left\{\mathbf{Q} : \mathbf{Q} \in [0,1]^{N\times A} \text{ and } \mathbf{Q}\cdot\mathbf{1} = \mathbf{1}\right\}.$$

  In projected gradient ascent, at each update step, the updated point is projected back onto $\mathcal{D}$ by finding a point in $\mathcal{D}$ closest to $\mathbf{P}$, i.e, the update step is

  $$\mathbf{P} \leftarrow \underset{\mathbf{Q}\in\mathcal{D}}{\arg\min} \left|\left|\mathbf{Q} - (\mathbf{P} + \epsilon.\nabla_{\mathbf{P}}\lambda(\mathbf{P}, Y; \mathcal{C}))\right|\right|^2. \tag{3.9}$$

We now comment on the complexity of computing the expected likelihood for a few examples of system channel $\mathcal{C}$ in Fig. 3.1. We refer the reader to Appendix 3.6.2 of this chapter for a more detailed discussion.

Discrete memoryless channel (DMC): When $\mathcal{C}$ is a DMC, $\lambda(\mathbf{P}, Y; \mathcal{C})$ breaks down into a product of N terms and can be computed in $O(NA)$. We clarify here that this case does not subsume the situation of having an encoder before a DMC. Note that the ML formulation in Theorem 3.1 also breaks down into N smaller problems each of which can be solved efficiently. This can also be proved to be equivalent to the symbolwise MAP estimate.

Probabilistic finite state machine (FSM): Say $\mathcal{C}$ is an FSM with states in $\mathcal{S}$ which outputs exactly K symbols corresponding to each input symbol. Then the expected likelihood can be computed using a dynamic programming approach in $O(A|\mathcal{S}|^2N)$. We note that this complexity is of the same order as the complexity of computing the ML via Viterbi algorithm or the symbolwise MAP via Forward-Backward algorithm. The class of probabilistic FSM system channels encompasses a variety of situations. For example, a convolutional encoder followed by a DMC can be represented by such a model.

Deletion channel: Assume that $\mathcal{C}$ deletes each input symbol with a constant probability of deletion δ (the encoder is an identity encoder). We prove that a dynamic program can be used to compute $\lambda(\cdot)$ in $O(NM)$ (see Appendix 3.6.2).

Sticky channel: A sticky channel with parameter p repeats each input symbol $K \in \{1, 2, ...\}$ times with a probability $(1-p)^{K-1}p$. Recent works have looked at the capacity of sticky channels [CR19]. We can prove that a dynamic program can compute $\lambda(\cdot)$ in $O(NM^2)$ time complexity (see Appendix 3.6.2).

3.3.2 Coordinate refinement

In situations where the expected likelihood function and its gradient are efficiently computable, we give a heuristic for the ML formulation in Theorem 3.1. This algorithm exploits the linearity of $\lambda(\mathbf{P}, Y; \mathcal{C})$ when projected on to the coordinates $\mathbf{P}_i$ (3.4). The basic underlying idea is as follows (the exact algorithm is detailed in Alg. 3.1):

- Say we start with a distribution matrix $\mathbf{P} \in [0,1]^{N \times A}$.
- We iterate over the indices [1:N] (the rows of $\mathbf{P}$) in a specified order (here we do so greedily). In the iteration corresponding to index i, we update row $\mathbf{P}_i$ such that the value of $\lambda(\cdot)$ never decreases.
- This update is done by comparing $\lambda(\mathbf{P}^{(i\to a)}, Y; \mathcal{C})\ \forall\ a$ and picking the a which maximizes $\lambda(\mathbf{P}^{(i\to a)}, Y; \mathcal{C})$, i.e.,
$$\mathbf{P} \leftarrow \mathbf{P}^{(i \to \arg\max_a \lambda(\mathbf{P}^{(i\to a)}, Y; \mathcal{C}))}. \tag{3.10}$$
  Note that $\forall\ i,\ \exists\ a$ such that $\lambda(\mathbf{P}^{(i\to a)}, Y; \mathcal{C}) \geq \lambda(\mathbf{P}, Y; \mathcal{C})$ due to (3.4), thus ensuring that the update step never decreases $\lambda(\cdot)$. Further, (3.5) signifies that $\nabla_{\mathbf{P}} \lambda(\mathbf{P}, Y; \mathcal{C})$ computes $\lambda(\mathbf{P}^{(i\to a)}, Y; \mathcal{C})\ \forall\ i, a$.

Iterating over the indices [1:N] once amounts to one *refinement iteration*. At the end of a refinement iteration, the final $\mathbf{P}$ is a lattice point (since every row has been updated to a unit vector). A new refinement iteration can now be started using current distribution $\mathbf{P}$ to further improve $\lambda(\cdot)$. Note that once we reach a lattice point, every update step results in a distribution which is also lattice point. Since the number of lattice points are finite, there will arise a situation where the update stagnates (does not strictly increase $\lambda(\cdot)$). In that case, we have arrived at a *fixed point* of this algorithm and we stop.

Before moving further we first define a *fixed point* of an update algorithm. An update algorithm takes as input a distribution $\mathbf{P}$ and updates it iteratively. The projected gradient ascent and coordinate refinement are both update algorithms.

Definition 3.2. $\mathbf{P}^{\text{fixed}}$ is a *fixed point* of an update algorithm if the update step applied on $\mathbf{P}^{\text{fixed}}$ does not change $\mathbf{P}^{\text{fixed}}$.

$\mathbf{P}^{\text{fixed}}$ is a fixed point of the projected gradient ascent if the right-hand side of (3.9) is equal to $\mathbf{P}^{\text{fixed}}$ itself and it is a fixed point of coordinate refinement if the right-hand side of (3.10) is $\mathbf{P}^{\text{fixed}}$ itself. More precisely,

- $\mathbf{P}$ is a fixed point for project gradient descent iff

$$\mathbf{P} = \arg\min_{\mathbf{Q}\in\mathcal{D}} \left|\left|\mathbf{Q} - (\mathbf{P} + \epsilon\ \nabla_{\mathbf{P}}\lambda(\mathbf{P}, Y; \mathcal{C}))\right|\right|^2 \ \forall\ \epsilon > 0, \tag{3.11}$$

  where $\mathcal{D} = \left\{\mathbf{Q} : \mathbf{Q} \in [0,1]^{N\times A} \text{ and } \mathbf{Q}\cdot\mathbf{1} = \mathbf{1}\right\}$.

- $\mathbf{P}$ is a fixed point for coordinate refinement iff

$$\mathbf{cat}^{-1}(\mathbf{P})_i = \arg\max_{a\in\mathcal{A}}\ \lambda(\mathbf{P}^{(i\to a)}, Y; \mathcal{C}) \quad \forall\ i. \tag{3.12}$$

We do note that there could be multiple solutions to $\arg\max_{a\in\mathcal{A}}\ \lambda(\mathbf{P}^{(i\to a)}, Y; \mathcal{C})$, but we choose to stop coordinate refinement at $\mathbf{P}^{\text{fixed}}$ instead. Although coordinate refinement reaches a fixed point after a finite number of refinement iterations, this number could potentially be exponential in N. However in practice, for the deletion channel, the coordinate refinement reached a fixed point mostly within 3 refinement iterations even for N=100. Further, we give an interesting result about such fixed points.

Theorem 3.3. *If the distribution* $\mathbf{P}$ *is a fixed point for coordinate refinement (given* Y*), then* $\mathbf{P}$ *is also a fixed point for projected gradient ascent.*

Algorithm 3.1 Greedy coordinate refinement

```
1: Inputs: Distribution $\mathbf{P}^{\text{init}}$, Observation $Y$, Algorithms to query $\lambda(\mathbf{P}, Y; \mathcal{C})$ and
   $\nabla_{\mathbf{P}}\lambda(\mathbf{P}, Y; \mathcal{C})$, Max refinement iterations $RF_{max}$
2: Outputs: Estimate $\hat{X}$
3: Initialize $\mathbf{P} = \mathbf{P}^{\text{init}}$
4: for $iter$ in $[1 : RF_{max}]$ do
5:     Initialize visited indices $\mathcal{I} = \varnothing$
6:     while $|\mathcal{I}| < N$ do
7:         Compute gradient matrix $\mathbf{G} = \nabla_{\mathbf{P}}\lambda(\mathbf{P}, Y; \mathcal{C})$
8:         if $\mathbf{P}$ is a lattice point and satisfies (3.12) then
9:             return $\mathbf{cat}^{-1}(\mathbf{P})$ and exit
10:        $(i^*, a^*) = \underset{i\in[N]\setminus\mathcal{I},\ a\in\mathcal{A}}{\arg\max} \mathbf{G}_{ia}$
11:        Update $\mathbf{P} \leftarrow \mathbf{P}^{(i^*\to a^*)}$
12:        Update $\mathcal{I} \leftarrow \mathcal{I} \cup \{i^*\}$
13: return $\mathbf{cat}^{-1}(\mathbf{P})$
```

The proof of the theorem can be found in Appendix 3.6.3. The idea behind the proof is that the gradient at a fixed point $\mathbf{P}$ extends outwardly from $\mathbf{P}$ such that any point lying outside $\mathcal{D}$ in the direction of the gradient is closer to $\mathbf{P}$ than every other point in $\mathcal{D}$ (see Fig. 3.2). Thus the result of the projection onto $\mathcal{D}$ is again $\mathbf{P}$.

Note on initializations for coordinate refinement: A natural question is if it makes a difference initializing $\mathbf{P}$ as an interior point ($\mathbf{P}_{ia}\in(0,1)$) or a lattice point. For an interior

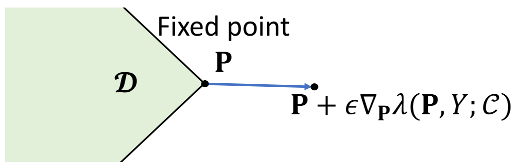

Fixed point
$\mathbf{P}$
$\mathcal{D}$
$\mathbf{P} + \epsilon\nabla_{\mathbf{P}}\lambda(\mathbf{P}, Y; \mathcal{C})$

Figure 3.2: Figure illustrating the idea behind Theorem 3.3.

point, the first refinement iteration updates $\mathbf{P}$ to a lattice point and subsequent refinement iterations deal with $\mathbf{P}$ in the set of lattice points thereon, in which case we are optimizing $\Pr(Y|X,\mathcal{C})$ directly. One could have initialized $\mathbf{P}$ to be a lattice point to begin with and optimize $\Pr(Y|X,\mathcal{C})$, circumventing the use of expected likelihood: numerical evaluation in the next section indicates that such an initialization can significantly deteriorate the performance of coordinate refinement.

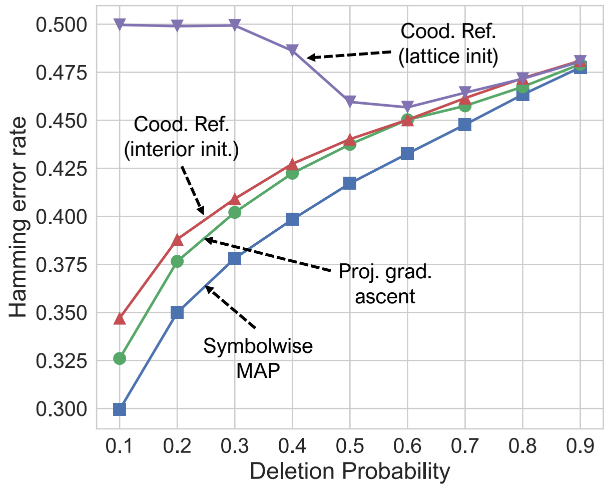
3.4 Example application: numerical results for the deletion channel

As an application, we focus on the deletion channel and show numerical results for the various algorithms which exploit our ML and SPs formulation. We restrict ourselves to the binary alphabet $\mathcal{A} = \{0, 1\}$ for simplicity. As mentioned in Section 3.3, the expected likelihood for the deletion channel can be computed in $O(NM)$. A similar dynamic programming approach can be employed to compute its gradient in $O(NM)$ as well, as we did in chapter 2. Our comparisons are based on two metrics:

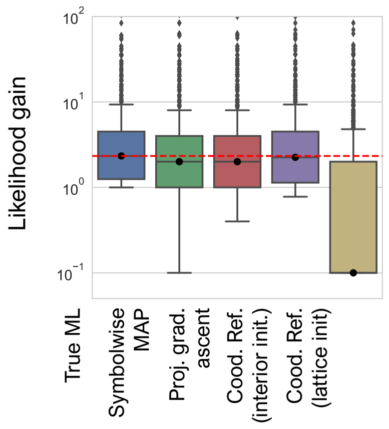
- The likelihood gain $\gamma(x,\hat{X}) = \frac{\Pr(Y|X=\hat{X},\mathcal{C})}{\Pr(Y|X=x,\mathcal{C})}$ where x is the actual input and $\hat{X}$ is the estimate. The true ML sequence gives the optimal (largest) likelihood gain.
- The hamming error rate $\psi(x,\hat{X})$ which is defined to be number of bit errors between the actual input x and estimate $\hat{X}$ divided by its blocklength. The symbolwise MAP is an optimal estimator for the hamming error rate. Note that, in general, optimizing for hamming error rate is not equivalent to optimizing for the likelihood gain and vice-versa.

We compare the performance of the following algorithms:

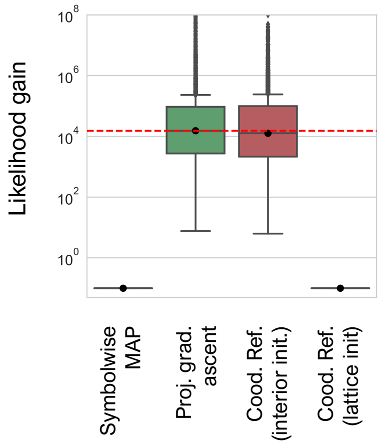
- *Symbolwise MAP*: we first compute the SPs via Theorem 3.2 and then pick the most



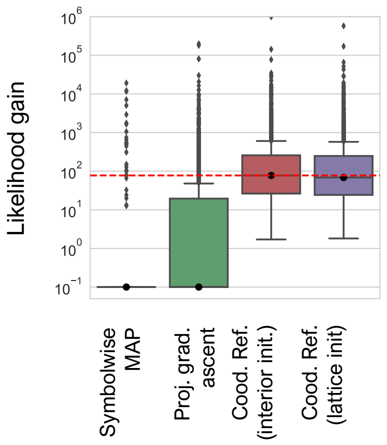
(a) Hamming error rates for blocklength $N = 100$.



(b) Likelihood gains for $N = 10, \delta = 0.2$.



(c) Likelihood gains for $N = 100, \delta = 0.2$.



(d) Likelihood gains for $N = 100, \delta = 0.8$.

Figure 3.3: Hamming error rates and likelihood gains for coordinate refinement (with both vertex and interior point initializations), symbolwise MAP, and projected gradient ascent. We compare for various blocklengths and deletion probabilities. We use box plots to visualize the sample distribution of the likelihood gains. The ends of the boxes indicate the upper and lower quartiles, the dot in each box is the median of the samples, the whiskers indicate the extrema and the diamonds are deemed as outlier samples. We note here that we enforce a lower cap for the likelihood gain at 0.1 to aid log domain visualization.

likely symbol for each position.

- *Projected gradient ascent:* as defined by (3.9). At distribution $\mathbf{P}$, we use an adaptive step size $\epsilon = \frac{0.1}{\lambda(\mathbf{P}, Y; \mathcal{C})}$ and allow a maximum of 200 update steps.

- *Coordinate refinement with interior point initialization:* We use Alg. 3.1 with $\mathbf{P}^{\text{init}}$ whose entries are all 0.5, i.e., they correspond to the uniform distribution.

- *Coordinate refinement with lattice point initialization:* We use Alg. 3.1 and initialize $\mathbf{P}^{\text{init}}$ as a random lattice point.

Observations in Fig. 6.4.

- Symbolwise MAP has the least hamming error rate as it is an optimal estimator for this error metric. However, it has poor likelihood gains. The reasoning is very specific to the nature of deletion channels – changing just a few bits could vastly affect the likelihoods of the corresponding sequences. For instance, consider N=5 and an observation Y=001. It is easily seen that input sequence X=00001 corresponds to a high likelihood while X=00000 corresponds to 0 likelihood although it differs by only one bit.

- Coordinate refinement with lattice point initialization is seen to perform much worse than coordinate refinement with interior point initialization in all cases, which supports the usefulness of the equivalence provided by Theorem 3.1.

- Coordinate refinement with interior point initialization has consistently good likelihood gain performance across deletion probabilities unlike the other algorithms. One intuitive explanation is that it can be envisioned as a two step process: 1) in the first refinement iteration, the algorithm performs a coarse search (via the gradient values) and finds a "good" initial lattice point distribution for subsequent refinement iterations 2) subsequent refinement iterations finely "refine" the symbols to further improve the quality of the solution. The projected gradient ascent is lacking of step 2) while coordinate refinement with lattice point initialization lacks step 1).

3.5 Conclusions and Open Questions

In this chapter, we formulated the ML estimate of an arbitrary system channel as a continuous optimization problem; in particular, we optimize the *expected likelihood function* over the space of product distributions for X, instead of optimizing the actual likelihood. This opens the door to the use of first-order heuristics like gradient ascent. We connected the SPs to the expected likelihood function and its gradient. As an application, we illustrated performance benefits of our formulations via numerics for the deletion channel. An open question is to understand for what classes of system channels these techniques offer benefits over existing methods, and how large these benefits are. Another open question is to understand how the ML solution relates to error rates, such as Hamming error rate, for the particular system channels considered in this chapter. Finally, it would also be interesting to come up with error-rate guarantees for the heuristics introduced in this chapter.

3.6 Appendix

3.6.1 Proof of Lemma 3.1

Proof. First, note that if $X \sim (\mathbf{P})$, then $\Pr(x) = \prod_{j=1}^{N} \mathbf{P}_{jx_j}$. Consider the right hand side of the equation,

$$\begin{aligned}
\mathbf{P}_{ia}\lambda(\mathbf{P}^{(i\to a)}, Y; \mathcal{C}) &= \mathbf{P}_{ia}\mathbb{E}_{X\sim\mathbf{P}^{(i\to a)}} \Pr(Y|X,\mathcal{C}) \\
&= \mathbf{P}_{ia} \sum_{x} \mathbb{1}_{x_i=a} \prod_{\substack{j=1 \\ j\neq i}}^{N} \mathbf{P}_{jx_j} \Pr(Y|X=x,\mathcal{C})
\end{aligned}$$

$$= \sum_{x:x_i=a} \prod_{j=1}^{N} \mathbf{P}_{jx_j} \Pr(Y|X=x,\mathcal{C})$$
$$= \sum_{x:x_i=a} \Pr(x)\Pr(Y|X=x,\mathcal{C}).$$

□

3.6.2 Complexity of computing the expected likelihood function

Discrete memoryless channel (DMC): When $\mathcal{C}$ is a DMC, $N{=}M$ and

$$\Pr(Y_1...Y_N|X_1...X_N,\mathcal{C}) = \prod_{i=1}^{N} \Pr(Y_i|X_i,\mathcal{C}).$$

This implies that $\lambda(\mathbf{P},Y;\mathcal{C}) = \prod_i \lambda(\mathbf{P}_i,Y_i;\mathcal{C})$. Now $\lambda(\mathbf{P}_i,Y_i;\mathcal{C}) = \sum_{x\in\mathcal{A}} \mathbf{P}_{ix}\Pr(Y_i|X_i{=}x)$ which can be computed in $O(A)$ time complexity. And thus, the complexity of computing $\lambda(\mathbf{P},Y;\mathcal{C})$ is $O(NA)$. Note that the ML formulation in Theorem 3.1 breaks down into N optimization problems each of which can be solved efficiently. It can also be proved easily that this is equivalent to the symbolwise MAP estimate.

Probabilistic finite state machine (FSM): Assume $\mathcal{C}$ is a FSM with states in $\mathcal{S}$. Further, assume that each input symbol x_i to $\mathcal{C}$ results in exactly K output symbols $y_{i1},...,y_{iK}$. Thus $M{=}NK$ in Fig. 3.1. The FSM is defined by the emission probabilities $\Pr(y_{i1},...,y_{iK},s|x_i,s^{\text{current}})$ for outputs $y_{i1},...,y_{iK}$. Note that the start and end states of the encoder s^{init} and s^{end} are included in channel information $\mathcal{C}$. For such a model, we have the following dynamic program to compute $\lambda(\cdot)$:

$$\lambda(\mathbf{P},Y;\mathcal{C},s^{\text{cur}},s^{\text{end}}) = \sum_{s\in\mathcal{S}} \Bigg(\mathop{\mathbb{E}}_{X_1\sim\mathbf{P}_1} \Pr(Y_1...Y_K,s|X_1,s^{\text{cur}})$$
$$\lambda(\mathbf{P}_{[2:N]},Y_{[K+1:NK]};\mathcal{C},s,s^{\text{end}})\Bigg).$$

Computing the expectation term is $O(A)$. The dynamic program involves computation over the indices $[1{:}N]$ as well as the states $\mathcal{S}$ and the computation of each term takes $O(A|\mathcal{S}|)$

operations. Thus the total worst case time complexity is $O(A|\mathcal{S}|^2N)$. We note that this complexity is same as the complexity of Viterbi and Forward-Backward algorithms for HMMs (which can be thought of as a FSM). The class of probabilistic FSM channels encompasses a variety of situations. For example, a convolutional encoder followed by a DMC can be represented by such a probabilistic FSM – each independent message symbol x_i is mapped to K coded symbols $z_{i1}, ..., z_{iK}$ by the convolutional encoder and each of these coded symbols results in K output symbols $y_{i1}, ..., y_{iK}$ when passed through a DMC. The states of the FSM are precisely the states of the convolutional encoder.

Deletion channel: $\mathcal{C}$ is a deletion channel which deletes each input symbol with a constant probability of deletion δ. We give a dynamic program for the expected likelihood which directly arises from the channel definition. Consider $\Pr(Y_1Y_2...Y_M|X_1...X_N)$. Clearly $M \leq N$ for a deletion channel. We have two possibilities for the input symbol X_1 – it is either deleted from Y or it is reflected as Y_1. Therefore,

$$\begin{aligned}\Pr(Y_{[1:M]}|X_{[1:N]}) &= \Pr(X_1 \text{ deleted}) \Pr(Y_{[1:M]}|X_{[2:N]}) + \Pr(X_1 \text{ gives } Y_1) \Pr(Y_{[2:M]}|X_{[2:N]}) \\ &= \delta \Pr(Y_{[1:M]}|X_{[2:N]}) + \mathbb{1}_{X_1=Y_1}(1-\delta) \Pr(Y_{[2:M]}|X_{[2:N]}).\end{aligned}$$

Taking expectation on both sides w.r.t to $X \sim (\mathbf{P})$ gives the dynamic program:

$$\lambda(\mathbf{P}, Y; \delta) = \delta\lambda(\mathbf{P}_{[2:N]}, Y; \delta) + \mathbf{P}_{1Y_1}(1-\delta)\lambda(\mathbf{P}_{[2:N]}, Y_{[2:M]}; \delta). \tag{3.13}$$

Note that the above dynamic program provides a way to compute $\lambda(\mathbf{P}, Y; \delta)$ in $O(NM)$ time complexity and is equivalent to the one derived in chapter 2.

Sticky channel: A sticky channel with parameter p repeats each input symbol $K \in \{1, 2, ...\}$ times with a probability $(1-p)^{K-1}p$. Recent works have looked at the capacity of sticky channels [CR19]. Clearly $M \geq N$ for a sticky channel. We now have multiple possibilities for the leading input symbol X_1 – it is repeated K times, where $K \in \{1, 2, ...M\}$. This leads to a similar dynamic program for the sticky channel (we omit the steps here for brevity):

$$\lambda(\mathbf{P}, Y; p) = \sum_{i=1}^{M} \left(p(1-p)^{i-1} \mathbb{1}_{\{Y_1 = Y_2 = \ldots = Y_i\}} \mathbf{P}_{1Y_1} \lambda(\mathbf{P}_{[2:N]}; Y_{[i+1:M]}; p) \right).$$

$\lambda(\mathbf{P}, Y; p)$ via the above dynamic program can be computed in $O(NM^2)$ time complexity.

3.6.3 Proof of Theorem 3.3

Proof. Let $\mathcal{A} = \{1, 2, ..., A\}$. To prove:

$$\mathbf{P} = \arg\min_{\mathbf{Q} \in \mathcal{D}} \left|\left| \mathbf{Q} - (\mathbf{P} + \epsilon . \nabla_{\mathbf{P}} \lambda(\mathbf{P}, Y; \mathcal{C})) \right|\right|^2 \forall\ \epsilon > 0, \tag{3.14}$$

given that

$$\begin{aligned} \mathbf{cat}^{-1}(\mathbf{P})_i &= \arg\max_{a \in \mathcal{A}}\ \lambda(\mathbf{P}^{(i \to a)}, Y; \mathcal{C}) \quad \forall\ i \\ &= \arg\max_{a \in \mathcal{A}}\ \frac{\partial \lambda(\mathbf{P}, Y; \mathcal{C})}{\partial \mathbf{P}_{ia}} \quad \forall\ i, \end{aligned} \tag{3.15}$$

where $\mathcal{D} = \{\mathbf{Q} \geq \mathbf{0}, \mathbf{Q} \cdot \mathbf{1} = \mathbf{1}\}$.

Notice that the constraints in $\mathcal{D}$ are independent constraints for each row of $\mathbf{Q}$ and $\left|\left| \mathbf{Q} - (\mathbf{P} + \epsilon . \nabla_{\mathbf{P}} \lambda(\mathbf{P}, Y; \mathcal{C})) \right|\right|^2 = \sum_i \left|\left| \mathbf{Q}_i - (\mathbf{P}_i + \epsilon . \nabla_{\mathbf{P}_i} \lambda(\mathbf{P}, Y; \mathcal{C})) \right|\right|^2$. Hence it is sufficient to prove (3.14) for each row, i.e, to prove that

$$\mathbf{P}_i = \arg\min_{\mathbf{Q}_i \in \mathcal{D}_i} \left|\left| \mathbf{Q}_i - (\mathbf{P}_i + \epsilon . \nabla_{\mathbf{P}_i} \lambda(\mathbf{P}, Y; \mathcal{C})) \right|\right|^2 \forall\ \epsilon > 0, \tag{3.16}$$

given (3.15) and where $\mathcal{D}_i = \{\mathbf{Q}_i \geq \mathbf{0}, \mathbf{Q} \cdot \mathbf{1} = 1\}$.

Since $\mathbf{P}$ is a lattice point (fixed points of coordinate refinement can only be lattice points) we will assume for simplicity that $\mathbf{P}_i = [1, 0, ...0]$; our arguments are easy to generalize. Also let $\epsilon . \nabla_{\mathbf{P}_i} \lambda(\mathbf{P}, Y; \mathcal{C})) = [\epsilon_1, \epsilon_2, ..., \epsilon_A]$. From (3.15), clearly $\epsilon_1 \geq \epsilon_a \geq 0\ \forall a$. Thus, (3.16) reduces to proving the following:

$$\begin{aligned}\mathbf{P}_i = \arg\min_{\mathbf{Q}_i} \quad & (\mathbf{Q}_{i1} - (1+\epsilon_1))^2 + \sum_{a=2}^{A} (\mathbf{Q}_{ia} - \epsilon_a)^2 \\ \text{s.t.} \quad & \mathbf{Q}_{ia} \geq 0 \quad \forall a \\ & \sum_a \mathbf{Q}_{ia} = 1.\end{aligned} \tag{3.17}$$

In order to prove the above, we do the following: consider the case when $\mathbf{Q}_{i1} < 1$. Therefore, $\exists\ b : \mathbf{Q}_{ib} > 0$ since $\sum_a \mathbf{Q}_{ia} = 1$. We argue that moving the mass of $\mathbf{Q}_{ib}$ into $\mathbf{Q}_{i1}$ will only decrease the objective value of (3.17). More precisely, we prove that

$$\begin{aligned}(\mathbf{Q}_{i1} - (1+\epsilon_1))^2 + & (\mathbf{Q}_{ib} - \epsilon_b)^2 \\ & > (\mathbf{Q}_{i1} + \mathbf{Q}_{ib} - (1+\epsilon_1))^2 + (\epsilon_b)^2.\end{aligned}$$

The above inequality is, in fact, simple to see as it reduces to verifying if $\mathbf{Q}_{ib}(1+\epsilon_1) > \mathbf{Q}_{ib}\epsilon_b$. This is true since by our hypotheses $\mathbf{Q}_{ib} > 0$ and $\epsilon 1 \geq \epsilon_b \geq 0$.

This argument could be applied sequentially for every $b \neq 1 : \mathbf{Q}_{ib} > 0$ thereby moving all the mass into $\mathbf{Q}_{i1}$. Thus the distribution defined by $\mathbf{Q}_{i1} = 1$ and $\mathbf{Q}_{ib} = 0\ \forall\ b \neq 1$ is optimal and this is equal to $\mathbf{P}_i$, proving (3.17) and consequently the theorem. $\square$

Part II

Group Testing and Epidemiology

CHAPTER 4

Static and independent infection model

Summary: *In this chapter, we study group testing for the static infection model with non-uniform priors. We take a practical approach to this problem: we fix the decoder and the number of tests, and we ask what is the best test design one could use? We examine this problem for the definite non-defectives (DND) decoder and formulate it as a (non-convex) optimization problem, where the objective function is the expected number of errors for a particular design. We propose an approximate solution via gradient descent, which we further optimize with informed initialization. We show numerical results that demonstrate that our approach achieves much lower (upto 50% lower in our simulations) error rates in comparison with the existing state-of-the-art test designs.*

Group testing is a strategy that can reduce the number of tests needed to identify infected people in a population, by pooling together diagnostic samples from multiple individuals. The idea has recently attracted significant attention in the context of the pandemic ([GG20, Bro20, Ell20, Ver20, Gho20, KLL20]), and several countries (including India, Germany, US, and China) have already deployed preliminary group-testing strategies ([Mal20, FDA20]).

Group testing has a rich history in academia, dating back to R. Dorfman in 1943, and a number of variations and setups have been examined so far ([Dor43, AJS19, DH93, MA16]). Simply stated, group testing assumes a population of N individuals out of which some are infected, and the goal is to design testing strategies and corresponding decoding algorithms to identify the infections from the test results. Most works revolve around proposing a particular hand-crafted test design (e.g. random Bernoulli design) coupled with a decoding strategy (e.g. Definite Defectives, Definite Non-Defectives), and guarantees are provided on the number of tests required to achieve vanishing probability of error. Additionally, order-optimality results have been proved for the asymptotic regime, where the population size tends to infinity. For example, in a population of $N \to \infty$ members, if very few people (say $k < N^{1-\Omega(1)}$) are infected, one can identify them with $\mathcal{O}(k \log N)$ group tests, using random Bernoulli test designs ([BPS20]).

To the best of our knowledge, the following complementary question remains unexplored: Given a fixed decoding strategy and a number of tests T (perhaps smaller than what is needed to achieve zero error), what is the *best* test design one may use? In this chapter, we examine this in the context of nonadaptive group testing, where all pooled tests are administered in a single stage – hence the delay of the test outcomes is the least possible – and under the assumption of a Definite Non-Defectives (DND) decoder, which eliminates false negatives by construction. Interestingly, a discussion with the General Secretary of Public Health in an EU state has revealed that this question is perhaps the most relevant in practice, as both private and public lab facilities have limited testing capacity per day, and what actually matters is how to use the available tests most efficiently.

In this chapter, we show that the above problem admits a non-convex optimization formulation that can be approximately solved via gradient descent (GD). More specifically, the problem requires finding a test-design matrix G (determining how the diagnostic samples are pooled together into tests) that minimizes the expected number of the erroneous identifications (i.e. false positives). This, however, presents two challenges: (a) the analytical computation of the expected number of false positives turns out to be computationally difficult; (b) because $G \in \{0,1\}^{T \times N}$, this yields a challenging combinatorial optimization problem. To tackle these challenges, we act as follows: First, we provide a lower bound that we use as a proxy in the optimization problem; our lower bound can be computed in $O(N^2)$ runtime. We then reformulate the resulting combinatorial optimization problem based on an equivalence result; we show that the objective function in this continuous formulation as well as its gradient can be computed in $O(N^2)$, using dynamic programming techniques, enabling the use of GD. To further improve the performance of our method, we propose two approaches to GD: (i) an informed initialization with information from classic test designs, such as the Constant Column Weight (CCW) design and the Coupon Collector Algorithm (CCA); (ii) a stochastic re-initialization of the state of the solution every few gradient iterations (e.g. 100 iterations), in a way that allows GD to move across various neighborhoods, while also ensuring that the objective value does not increase by much with re-initialization.

Numerical evaluations show that the GD based approaches significantly outperform classical test designs, achieving upto 58% (in the best case) fewer errors with the DND decoder on simulated infection models. Rather surprisingly, GD based designs also significantly outperform classical test designs when the decoder is switched to definite defectives (DD), indicating transferability to other decoders.

4.1 Related work

We here give a brief overview of group testing; the exact problem we consider in this chapter will be detailed in Section 4.2.1. Three infection models are usually studied in the group testing literature: (i) in the *combinatorial priors model*, a fixed number of individuals k (selected uniformly at random), are infected; (ii) in *i.i.d probabilistic priors model*, each individual is i.i.d infected with some probability p; (iii) in the *non-identical probabilistic priors model*, each item i is infected independently of all others with prior probability p_i, so that the expected number of infected members is $\bar{k} = \sum_{i=1}^{N} p_i$. Infection models (i) and (ii) have received attention from researchers for the most part (see for example, [AS12, ABJ14, CJS14, SC16, JAS19, Ald19, BPS20, CGH20b, AFF20, PS20, CGH20a]). Infection model (iii) is the most general, yet also the least studied one ([LCH14]); we refer the reader to [AJS19] for an excellent summary of existing work on the above infection models. This chapter assumes infection model (iii) with non-identical probabilistic priors and accepts (ii) as a special case.

4.2 Preliminaries

In this section, we first precisely formulate the problem of interest, and then state a simple lemma on combinatorial optimization that is used in our main result.

4.2.1 Problem formulation

We consider the noiseless nonadaptive group testing problem with non-identical priors. There are N individuals in the population and individual i is infected independently with probability p_i. We assume that the value of p_i is known apriori[1]. We assign a binary random variable U_i to be the infection status of individual i: $U_i = \mathbb{1}\{\text{Individual } i \text{ is infected}\}$. As a result, $U_i \sim \text{Ber}(p_i)$. We will denote by $\mathbf{U} = (U_1, U_2, ..., U_N)$ the vector of infection statuses. The aim of group testing is to devise tests (measurements) and decoding algorithms to estimate $\mathbf{U}$.

Testing matrix: A testing matrix $G \in \{0,1\}^{T \times N}$ is a $T \times N$ binary matrix. Row t in the testing matrix represents the individuals participating in test t, i.e., if $G_{ti} = 1$, the individual i participates in test t. The test results corresponding to a particular realization of $\mathbf{U} = (U_1, U_2, ..., U_N)$ and G is defined as the vector $\mathbf{Y} = (Y_1, Y_2, ..., Y_T)$ where

$$Y_t = 1 - \prod_{i=1}^{N}(1 - G_{ti}U_i). \tag{4.1}$$

In words, the test t gives a positive result if any of the individuals participating in the test are infected, otherwise it gives a negative result[2]. In (4.1) it can be verified that $Y_t = 1$ if and only if there exists i such that both $G_{ti} = 1$ (individual i participates in the test) and $U_i = 1$ (individual i is infected). In order to infer $\mathbf{U}$ from Y, a *decoding algorithm*

[1]This is a standard assumption in group testing. Otherwise, epidemiological models for disease spread can be used to estimate these probabilities ([KMS17, SNF21a, SNF21b]).

[2]Most works in group testing express the right-hand side of (4.1) as a Boolean expression. However, we use this particular form (similar expression was given in [AFF20]) as it easily admits continuous-valued extension of the composing variables.

$r : \{0,1\}^T \rightarrow \{0,1\}^N$ constructs an estimate $\widehat{\mathbf{U}}$ of the infection statuses from the test results. In this work, we fix the decoding algorithm, which we describe next.

DND decoder: The definite non-defective (DND) decoder is a well-known decoding algorithm that forms an estimate of $\mathbf{U}$ by identifying those individuals who have participated in at least one negative test as healthy and labeling every other individual as infected – i.e., it operates under the principle "every item is defective unless proved otherwise". More precisely, it outputs an estimate $\widehat{\mathbf{U}}$ where

$$\widehat{U}_i = \prod_{t=1}^{T} Y_{ti}^{G_{ti}}. \tag{4.2}$$

Such an estimate has zero false negatives by construction. To see this, let $U_i = 1$. From (4.1), $Y_t = 1$ for all t where $G_{ti} = 1$. Using this in (4.2), we see that whenever $U_i = 1$, then $\widehat{U}_i = 1$. The number of errors (false positives) that the DND decoder makes for a particular realization $\mathbf{U}$ is given by

$$\sum_{i=1}^{N} \mathbb{1}\{\widehat{U}_i \neq U_i\} = \sum_{i=1}^{N} \mathbb{1}\{U_i = 0\}\mathbb{1}\{\widehat{U}_i = 1 | U_i = 0\},$$

and as a result the expected number of errors $\mathcal{E}(G)$ under the DND decoder for a given G is

$$\begin{aligned}\mathcal{E}(G) &\triangleq \mathbb{E}\left[\sum_{i=1}^{N} \mathbb{1}\{\widehat{U}_i \neq U_i\}\right] \\ &= \sum_{i=1}^{N} \Pr(U_i = 0)\Pr(\widehat{U}_i = 1 | U_i = 0) \\ &= \sum_{i=1}^{N} (1 - p_i)\mathbb{E}\left[\widehat{U}_i | U_i = 0\right], \end{aligned} \tag{4.3}$$

where the expectation is taken over the randomness in each $\widehat{U}_i$. Further, when U_i is fixed to be 0, $\widehat{U}_i$ is a function of G and $\mathbf{U} \setminus \{i\}$, where $\mathbf{U} \setminus \{i\} \triangleq (U_1, ..., U_{i-1}, U_{i+1}, ..., U_N)$ denotes the vector $\mathbf{U}$ without its i^{th} entry. Thus, fixing $U_i = 0$, and using (4.1) and (4.2) we have,

$$\widehat{U}_i = \prod_{t=1}^{T} \left(1 - \prod_{\substack{j=1: \\ j \neq i}}^{N} (1 - G_{tj} U_j) \right)^{G_{ti}} \overset{(a)}{=} \prod_{t=1}^{T} \left(1 - G_{ti} \prod_{\substack{j=1: \\ j \neq i}}^{N} (1 - G_{tj} U_j) \right),$$

where (a) follows because of the following fact: $(1-x)^y = 1-xy$ if $y \in \{0,1\}$. Now, denoting $\gamma_{t,i} \triangleq \left(1 - G_{ti} \prod_{\substack{j=1: \\ j \neq i}}^{N} (1 - G_{tj} U_j)\right)$ in the above expression, we rewrite (4.3) as:

$$\mathcal{E}(G) = \sum_{i=1}^{N} (1 - p_i) \mathbb{E}_{\mathbf{U} \setminus \{i\}} \prod_{t=1}^{T} \gamma_{t,i}. \tag{4.4}$$

Goal: In this chapter, we aim to minimize $\mathcal{E}(G)$ across G matrices of size $T \times N$, i.e., solve

$$G_{opt} = \underset{G \in \{0,1\}^{T \times N}}{\arg\min} \; \mathcal{E}(G). \tag{4.5}$$

Discussion: We first observe that $\gamma_{t,i}$ is not independent of $\gamma_{t',i}$ for $t \neq t'$ as both of them could share common U_j terms. As a result, the expectation of the product term in (4.4) is not trivially the product of expectations, which makes the computation of $\mathcal{E}(G)$ intractable, in general (indeed one could estimate $\mathcal{E}(G)$ using Monte-Carlo methods, belief propagation etc.). In Section 4.3 we provide a lower bound for $\mathcal{E}(G)$ which can be computed efficiently, and which we use as a proxy for $\mathcal{E}(G)$.

Second, in principle, (4.5) could be formulated for any decoder, not just the DND decoder. However, the particular nature of $\mathcal{E}(G)$ for the DND decoder takes a nice form, for which we can propose an approximate solution using the lower bounding technique (Section 4.3). With other decoders, such as the definite defective decoder or belief propagation based ones, we do not have an approach to calculate a non-trivial lower bound; this remains a challenging open problem.

4.2.2 A combinatorial optimization result

We now take a detour to prove a simple result that allows one to formulate a combinatorial optimization problems that aim to optimize over the vertices of an n-dimensional hypercube as an equivalent continuous optimization problem. Note that this extends the idea introduced in Chapter 3 (see Theorem 3.1) to admit general combinatorial optimization problems.

Lemma 4.1. *In order to solve*

$$\arg\min_{\mathbf{x}\in\{0,1\}^n} \; g(\mathbf{x}), \tag{4.6}$$

it is sufficient to solve

$$\arg\min_{\mathbf{q}\in[0,1]^n} \; f(\mathbf{q}), \tag{4.7}$$

$$\textit{where } f(\mathbf{q}) \triangleq \mathbb{E}_{\mathbf{X}\sim\mathbf{q}} g(\mathbf{X})$$

is a continuous extension of $g(\mathbf{x})$. The expectation in the above expression is taken w.r.t the distribution where each $X_i \sim$ Ber(q_i), and the X_is are independent of each other.

Proof. We first note that for any $\mathbf{q} \in [0,1]^n$ we have,

$$\mathbb{E}_{\mathbf{X}\sim\mathbf{q}} g(\mathbf{X}) \geq \min_{\mathbf{x}\in\{0,1\}^n} \; g(\mathbf{x}),$$

since the expectation of a random variable is at least as large as its minimum value over its support. Since the above holds for any $\mathbf{q}$, as a result we have

$$\min_{\mathbf{q}\in[0,1]^n} \; \mathbb{E}_{\mathbf{X}\sim\mathbf{q}} g(\mathbf{X}) \geq \min_{\mathbf{x}\in\{0,1\}^n} \; g(\mathbf{x}). \tag{4.8}$$

Let $\mathbf{x}^*$ be a minimizer of $g(\mathbf{x})$ in (4.6). The choice of $\mathbf{q}^* = \mathbf{x}^*$ (i.e. $\mathbf{X} = \mathbf{x}^*$ with probability 1) gives

$$f(\mathbf{q}^*) = \mathbb{E}_{\mathbf{X}\sim\mathbf{q}^*} g(\mathbf{X}) = g(\mathbf{x}^*) = \min_{\mathbf{x}\in\{0,1\}^n} \; g(\mathbf{x}). \tag{4.9}$$

From (4.8) and (4.9) we conclude that

$$\min_{\mathbf{q}\in[0,1]^n} \; \mathbb{E}_{\mathbf{X}\sim\mathbf{q}} g(\mathbf{X}) = \min_{\mathbf{x}\in\{0,1\}^n} \; g(\mathbf{x}).$$

In order to obtain a solution to (4.6), we obtain a solution $\mathbf{q}^*$ of (4.7) and any $\mathbf{X} \sim \mathbf{q}^*$ (sample $X_i \sim \text{Ber}(q_i)$) is a solution to (4.6). $\square$

Remark: There is a long history of using relaxation techniques to approximate solutions of combinatorial optimization problems (see [Tre11] for an overview). Most of these focus on linear programming relaxation techniques. In Lemma 4.1, there is no assumption on $g(\cdot)$ whatsoever and the resulting formulation may not be a linear program. Moreover, it may not be easy to compute $f(\cdot)$ in all cases and it may also not be easy to compute the gradient $\nabla f(\cdot)$ as well. In cases where exactly computing or approximating the gradient is easy (as is indeed the case in this work), one can use first-order optimization techniques such as GD.

4.3 Main results

In this section, we delineate our approach to find an approximate solution to (4.5). Following the discussion at the end of Section 4.2.1, our approach is three-fold: First, we lower bound $\mathcal{E}(G)$ by another function $\mathcal{E}_{LB}(G)$, whose computation turns out to be tractable; we then use $\mathcal{E}_{LB}(G)$ as a proxy for $\mathcal{E}(G)$. Next, we use Lemma 4.1 to show that it is sufficient to consider a continuous formulation of the resulting combinatorial optimization problem. Finally, we show that the objective function in the continuous formulation and its gradient can also be computed efficiently, thus enabling gradient descent.

4.3.1 A lower bound for $\mathcal{E}(G)$

As a first step, the following theorem states and proves a lower bound for $\mathcal{E}(G)$.

Theorem 4.1. *Consider a random vector $\mathbf{U} = (U_1, U_2, ..., U_N)$ where $U_i \sim$ Ber(p_i). For a given testing matrix G, and under the DND decoder, the expected number of errors (see* (4.4)*) satisfies*

$$\mathcal{E}(G) \geq \mathcal{E}_{LB}(G),$$

where

$$\mathcal{E}_{LB}(G) \triangleq \sum_{i=1}^{N}(1-p_i)\prod_{t=1}^{T}\left(1-G_{ti}\prod_{\substack{j=1:\\ j\neq i}}^{N}(1-G_{tj}p_j)\right).$$

Proof. First we recall the expression for $\mathcal{E}(G)$ in (4.4):

$$\mathcal{E}(G) = \sum_{i=1}^{N}(1-p_i)\mathbb{E}_{\mathbf{U}\setminus\{i\}}\prod_{t=1}^{T}\gamma_{t,i}.$$

Using the FKG inequality (see [FKG71, Kem77, PC] or proof of Lemma 4 in [Ald19]) one could show that

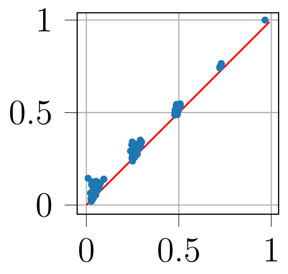
$$\mathbb{E}_{\mathbf{U}\setminus\{i\}}\prod_{t=1}^{T}\gamma_{t,i} \geq \prod_{t=1}^{T}\mathbb{E}_{\mathbf{U}\setminus\{i\}}\gamma_{t,i}.$$

A rigorous proof of the above statement can be found in Appendix 4.7.1. The idea is to show that $\gamma_{t,i}$ is an increasing function on $\mathbf{U}$ (assuming a partial ordering); using this observation, the result follows as an application of the FKG inequality. Thus, we have

$$\begin{aligned}
\mathcal{E}(G) &= \sum_{i=1}^{N}(1-p_i)\mathbb{E}_{\mathbf{U}\setminus\{i\}}\prod_{t=1}^{T}\gamma_{t,i} \\
&\geq \sum_{i=1}^{N}(1-p_i)\prod_{t=1}^{T}\mathbb{E}_{\mathbf{U}\setminus\{i\}}\gamma_{t,i} \\
&= \sum_{i=1}^{N}(1-p_i)\prod_{t=1}^{T}\left(1-G_{ti}\prod_{\substack{j=1:\\ j\neq i}}^{N}(1-G_{tj}p_j)\right) \\
&= \mathcal{E}_{LB}(G)
\end{aligned}$$

□

As it turns out, $\mathcal{E}(G)$ and the lower bound $\mathcal{E}_{LB}(G)$ are highly correlated, as is seen from the example scatter plots in Figure 4.1. In our simulations, we observed that $\mathcal{E}_{LB}(G)$ offers a very good approximation to $\mathcal{E}(G)$ – this property indicates that minimizing $\mathcal{E}_{LB}(G)$ is a good alternative to minimize $\mathcal{E}(G)$.



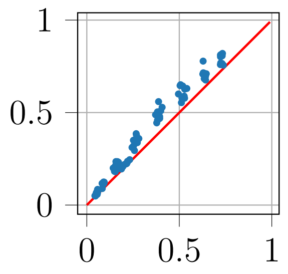


Figure 4.1: Scatter plot of $\mathcal{E}(G)$ (on y-axis) vs. $\mathcal{E}_{LB}(G)$ (on x-axis) normalized by the blocklength N. $\mathcal{E}(G)$ is estimated via Monte-Carlo simulations while $\mathcal{E}_{LB}(G)$ is computed exactly. For a fixed prior distribution, we pick a variety of G matrices and plot the two metrics – the left figure plots for every $G \in \{0,1\}^{2\times 4}$ while the right figure plots for 1000 choices of G sampled from $\{0,1\}^{300\times 500}$.

4.3.2 A continuous optimization formulation

Given the above discussion, we now propose using $\mathcal{E}_{LB}(G)$ as a proxy for $\mathcal{E}(G)$ – more precisely we propose to solve the following optimization problem:

$$\operatorname*{arg\,min}_{G\in\{0,1\}^{T\times N}} \mathcal{E}_{LB}(G). \tag{4.10}$$

We next use Lemma 4.1 to argue that a continuous formulation of (4.10) is equivalent to (4.10). Before stating the main result, we give a definition: we say that the matrix $G \sim Q$ (read as "G is distributed according to the distribution matrix Q") if each $G_{ti} \sim \text{Ber}(Q_{ti})\ \forall\ t, i$ and the G_{ti} variables are independent of each other.

Corollary 4.1. *Suppose $U_i \sim Ber(p_i)\ \forall\ i$. In order to solve the optimization problem*

$$\operatorname*{arg\,min}_{G\in\{0,1\}^{T\times N}} \mathcal{E}_{LB}(G), \tag{4.11}$$

it is sufficient to solve

$$\operatorname*{arg\,min}_{Q\in[0,1]^{T\times N}} \mathbb{E}_{G\sim Q}\mathcal{E}_{LB}(G). \tag{4.12}$$

This is a direct corollary of Lemma 4.1, where the objective function is $\mathcal{E}_{LB}(G)$ and we associate a parameter Q_{ti} corresponding to each G_{ti}.

Thus, we now have the following approximate formulation for which the objective function (and its gradient) can be computed in $O(N^2)$ time complexity (see Section 4.3.3). The hope is that solving (4.13) gives sufficiently good choices of $G \sim Q^*$; our experimental results in Section 4.5 indicate that this is indeed the case.

Approximate formulation: Solve for

$$Q^* = \operatorname*{arg\,min}_{Q \in [0,1]^{T \times N}} f(Q), \tag{4.13}$$

where $f(Q) \triangleq \mathbb{E}_{G \sim Q} \mathcal{E}_{LB}(G)$.

Given the above result, we propose a gradient descent (GD) approach to pick a testing matrix G. In essence, we are searching over the continuous space of distribution matrices Q. If the gradient of $f(Q)$ can be efficiently computed, one could use GD to converge to a local minima Q^* and pick a $G \sim Q^*$.

4.3.3 Expression for $f(Q)$

We now give a closed-form expression for $f(Q)$ and briefly discuss the computational complexity of computing $f(Q)$ and its gradient; the details are deferred to Section 4.7.2, Section 4.7.3 and Section 4.7.4. First, we have

$$\begin{aligned} f(Q) &\triangleq \mathbb{E}_{G \sim Q} \mathcal{E}_{LB}(G) \\ &= \mathbb{E}_{G \sim Q} \sum_{i=1}^{N} (1 - p_i) \prod_{t=1}^{T} \left(1 - G_{ti} \prod_{\substack{j=1: \\ j \neq i}}^{N} (1 - G_{tj} p_j) \right) \\ &\overset{(a)}{=} \sum_{i=1}^{N} (1 - p_i) \prod_{t=1}^{T} \mathbb{E}_{G \sim Q} \left(1 - G_{ti} \prod_{\substack{j=1: \\ j \neq i}}^{N} (1 - G_{tj} p_j) \right) \end{aligned}$$

$$= \sum_{i=1}^{N} (1 - p_i) \prod_{t=1}^{T} \left(1 - Q_{ti} \prod_{\substack{j=1: \\ j \neq i}}^{N} (1 - Q_{tj} p_j) \right), \tag{4.14}$$

where in (a) the expectation is pushed inside the product terms as $\mathcal{E}_{LB}(G)$ is linear when viewed as a function of a single G_{ti}. In Appendix 4.7.2 we discuss an $O(N^2)$ algorithm that simplifies the computation of $f(Q)$ above. Given (4.14), one could derive an expression for the gradient $\nabla f(Q)$ by calculating each partial derivative $\frac{\partial f(Q)}{\partial Q_{lm}}$. The details of the derivation can be found in Appendix 4.7.3. Moreover, in Appendix 4.7.4, we discuss the computation of $\nabla f(Q)$ in $O(N^2)$ runtime. The idea is to observe that many of the terms have similar forms and can be easily obtained from an intermediate common set of terms.

Notably, as it was perhaps already evident from Lemma 4.1 and , the assumption that the G_{ti}s are independent in Corollary 4.1 played a role one could have optimized over any random variable G with full support on $\{0, 1\}^n$, not necessarily the product distribution implied in Q. However, it is because of the product distribution that $f(Q)$ admits a polynomial-time computation for our purpose.

4.4 Algorithms

Given the approximate formulation in (4.13), we now propose a GD approach to find good choices of G. We further improve our method by using informed initialization with information provided by traditional group test designs. To this end, the GD approach can be viewed as a way to refine and improve existing designs via local search. Moreover, we also propose a variation of GD that numerically seems to converge to good choices of G in many situations even without informed initialization.

4.4.1 Baseline test designs

We first briefly describe two existing algorithms from the literature of group testing that we use as baselines for comparison.

• **Constant column weight (CCW) design** (see [AJS16, JAS19]). This design was introduced in the context of group testing for identical priors[3], but we adapt it to be applicable for non-identical priors as well, in addition to identical priors. Here we construct a randomized G assuming that all individuals have the same prior probability of infection p_{mean} (this assumption is trivially true if the priors are identical), where p_{mean} is defined as the mean prior probability of infection $\frac{1}{N}\sum_{i=1}^{N} p_i$. The testing matrix G is constructed column-by-column by placing each individual in a fixed number ($\frac{0.69T}{Np_{mean}}$) of tests, uniformly at random. For the case of identical priors p, CCW attains a vanishing probability of error with a DND decoder, given that the number of tests satisfies $T = \Omega(Np\log N)$; no such guarantees exist for non-identical priors. Nonetheless, we use it in this case as well as a baseline.

• **Coupon Collector Algorithm (CCA)** from [LCH14]. The CCA algorithm was introduced in [LCH14] for the case of non-identical, independent priors. In short, the CCA algorithm constructs a random non-adaptive test design G by sampling each row independently from a distribution (we refer the reader to [LCH14] for the exact description of this distribution). The idea is to place objects which are less likely to be infected in more number of tests and vice-versa. The CCA design achieves a vanishing probability of error with a DND decoder given that the number of tests used satisfies $T = \omega\left(4e\log N\sum_{i=1} p_i\right)$.

[3]Most of these were proposed in the context of combinatorial priors. However, Theorem 1.7 and Theorem 1.8 from [AJS19] imply that any algorithm that attains a vanishing probability of error on the combinatorial priors, also attains a vanishing probability of error on the corresponding i.i.d probabilistic priors.

Algorithm 4.1 Projected gradient descent(GD)

```
Require: Priors $(p_1, p_2, ..., p_N)$, initial point $Q^{(0)} \in [0,1]^{T\times N}$, step-size $\epsilon$, max. gradient
    steps $l_{max}$.
Ensure: Output a good choice of $G$
  $l = 1$
  while $l \leq l_{max}$ do
      Compute $f(Q^{l-1})$                                  ▷ See Appendix 4.7.2
      Compute $\nabla f(Q^{l-1})$                               ▷ See Appendix 4.7.4
      if Stopping criteria satisfied then
          break
      $Q^{(l)} \leftarrow Q^{(l-1)} - \epsilon \nabla f(Q^{l-1})$
      Set negative entries of $Q^{(l)}$ to 0
      Set entries of $Q^{(l)}$ greater than 1 to 1
  Sample $G \sim Q^{(l-1)}$
  return $G$
```

4.4.2 Test designs based on gradient descent

We are now ready to describe the gradient descent (GD) approaches to search for G. The high-level idea for our algorithms is as follows:

- We consider the approximate formulation in (4.13). Pick an initial point $Q^{(0)}$.

- At each gradient iteration l, update $Q^{(l)} \leftarrow Q^{(l-1)} - \epsilon\nabla_Q f(Q)$, where ϵ is the step size. Project $Q^{(l)}$ onto $[0,1]^{T\times N}$ by resetting negative entries to 0 and entries greater than 1 to 1.

- Stop based on some stopping criteria (e.g. limit number of gradient steps or check for convergence to local minima).

- Let Q^* be the resulting output. Sample a matrix G^* where $G^* \sim Q^*$ and return it.

The above algorithm is outlined as Algorithm 4.1. As it turns out, the choice of initializa-

Algorithm 4.2 (GD + Sampling) Sampled projected gradient descent to

```
Require: Priors $(p_1, p_2, ..., p_N)$, number of tests $T$, step-size $\epsilon$, max. gradient steps $l_{max}$,
    frequency of sampling $\nu \in$ N.
Ensure: Output a good choice of $G$
  $l = 1$
  Initialize $Q^{(0)} \leftarrow \mathbf{0}_{T\times N}$
  while $l \leq l_{max}$ do
      Compute $f(Q^{l-1})$                                   ▷ See Appendix 4.7.2
      Compute $\nabla f(Q^{l-1})$                            ▷ See Appendix 4.7.4
      if Stopping criteria satisfied then
          break
      $Q^{(l)} \leftarrow Q^{(l-1)} - \epsilon \nabla f(Q^{l-1})$
      Set negative entries of $Q^{(l)}$ to 0
      Set entries of $Q^{(l)}$ greater than 1 to 1
      if  mod $(l, \nu) == 0$ then
          Sample $G \sim Q^{(l)}$                            ▷ Sampling step
          Reassign $Q^{(l)} \leftarrow G$
  Sample $G \sim Q^{(l-1)}$
  return $G$
```

tion plays a significant role in finding good choices of G. For instance, the performance with all the entries of $Q^{(0)}$ set to $1/2$ was significantly worse as compared to setting all the entries to 0. This is not surprising as GD is a local search algorithm and for a function whose behavior varies substantially in different neighborhoods, initializing at a "good" neighborhood is extremely beneficial for GD. With this intuition, we propose the following initializations.

• **GD + CCW init.** We first sample a testing matrix according to the CCW testing matrix and set $Q^{(0)}$ as this matrix. The GD proceeds with this initialization.

• **GD + CCA init.** We first sample a testing matrix according to the CCA testing matrix and set $Q^{(0)}$ as this matrix. The GD proceeds with this initialization.

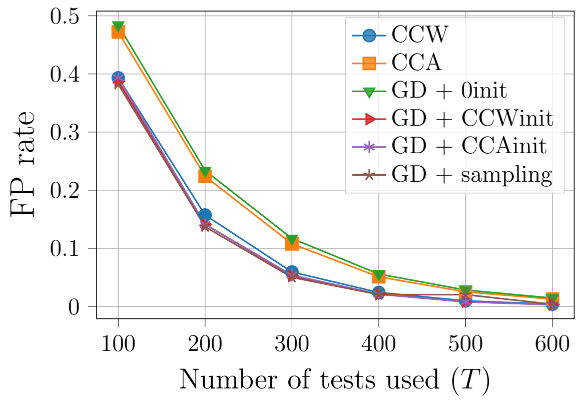
Notably, any other state-of-the-art test design could have been used as initialization. In principle, the above approach can be perceived as a way to refine existing test designs via local search. Alternatively, we also propose a modification to the GD approach called **GD + sampling** that helps avoid getting stuck in a local minima by encouraging GD to explore various neighborhoods. The idea is use stochastic re-initialization of the solution state every few gradient iterations, while ensuring that the value of objective function is approximately preserved. First note that the objective value $f(Q)$ is the mean of $f(G)$ with $G \sim Q$. Therefore, it is reasonable to expect that typical realizations of G will be such that $f(G)$ is close to $f(Q)$. Given this idea, we propose the following: start from the all 0 initialization. However, every few gradient iterations, we replace the current solution state $Q^{(l)}$ by G^s where G^s is sampled from the distributed matrix $Q^{(l)}$, i.e., $G^s \sim Q^{(l)}$. This encourages GD to move to a potentially different neighborhoods while (approximately) preserving the monotonocity property of GD.

4.5 Numerical results

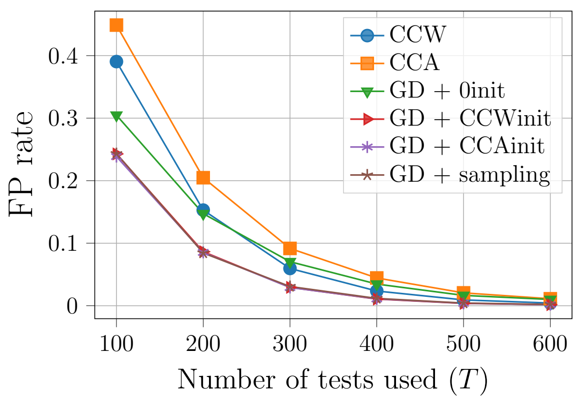
In this section, we show simulation results that demonstrate the improvement our GD based approaches provide in the design of test matrices.

Test designs compared: We compare the testing matrices G obtained via each of the following methods: CCW, CCA, GD + CCW init., GD + CCA init., GD + sampling. For completeness, we consider also the trivial all 0-initialization for GD (which we call **GD + 0 init**), where the initial point $Q^{(0)}$ is set to all zeros.

Set-up: (a) First fix the prior probabilities of infection $(p_1, p_2, ..., p_N)$. We consider two cases: (i) the identical prior model with $p_i = 0.05 \; \forall \; i$ and (ii) non-identical but independent priors where each p_i is sampled from an exponential distribution with mean 0.05; if $p_i > 1$,



(a) Identical priors $p = 0.05$, $N = 1000$.



(b) Priors sampled from an exponential distribution with mean 0.05, $N = 1000$. We average over 10 instances of such priors.

Figure 4.2: Mean FP rates with the DND decoder for each algorithm, as a function of T.

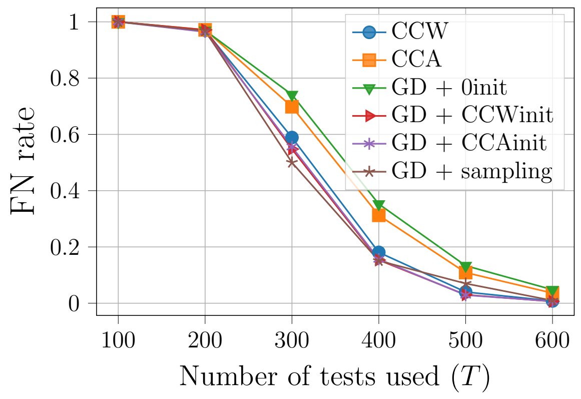
we set it to 1.

(b) For the given set of priors $(p_1, p_2, ..., p_N)$, and for each test design stated above, we estimate $\mathcal{E}(G)$ via Monte-Carlo simulations. Note that there exists randomness in choosing G in all the above test designs. Therefore, when computing the error rates, the average is also taken over all such G matrices.

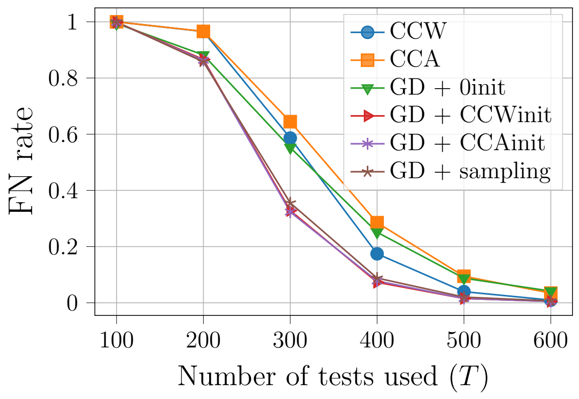
(c) For the case where each p_i is sampled from an exponential distribution, we repeat steps (a) and (b) 10 times and perform one more layer of averaging.

Metrics: We use the false positive (FP) rate (defined as the fraction of uninfected individuals incorrectly determined to be infected) as the metric to measure the performance of each test design w.r.t the DND decoder. Recall that the DND decoder results in 0 false negatives (FN) by construction.

Transferability to other decoders: As the GD based designs aim to find an optimal test matrix w.r.t the DND decoder, a natural follow-up question is how such a design performs



(a) Identical priors $p = 0.05$, $N = 1000$.



(b) Priors sampled from an exponential distribution with mean 0.05, $N = 1000$. We average over 10 instances of such priors.

Figure 4.3: Mean FN rates with the DD decoder for each algorithm, as a function of T.

when the decoder alone is switched. We aim to answer this question by measuring the performance of each of the test designs w.r.t a decoder called the Definite Defective (DD) decoder. One could also consider other decoders, such as ones based on belief propagation. However these decoders result in both FP and FN, and consequently the comparison between different methods is not trivial; it requires weighing FP against FN, which can be application specific. We refer the reader to Section 2.4 in [AJS19] for a precise description of DD decoder. It is worth mentioning that the DD decoder operates under the principle of "every item is not defective until proved otherwise". Consequently, DD has 0 FP by construction. In this case, we use as performance measure the false negative (FN) rate.

Observations: In Figure 4.2, we plot the FP rate for each test design w.r.t DND decoder, as a function of T. When the priors are identical, Figure 4.2a shows that the GD based methods with informed initialization (GD + CCW/CCA init.) and GD + sampling perform marginally better than CCW. Note that CCW is a state-of-the-art algorithm for group testing with identical priors. With non-identical priors, the GD based methods significantly outperform CCW and CCA, as can be seen from Figure 4.2b. Notably, the improvement of our enhanced GD with informed initialization or sampling seems inversely proportional to T, which is of practical importance.

Next, we plot the FN rate of each test design w.r.t the DD decoder, as a function of T in Figure 4.3. The performance trend here is similar to what was observed with the DND decoder, which further cements the usefulness of our GD based approach and its transferability to other decoders. Additional details on the error rates are provided in Appendix 4.7.5.

4.6 Conclusions and Open Questions

In this chapter, we formulated the search for good group-test designs, under the assumption of a DND decoder, as a non-convex optimization problem, and we proposed a solution via enhanced gradient descent. Our solution is approximate in the sense that it minimizes a lower bound on the expected number of identification errors (as opposed to the exact expectation). But, our numerical evaluation, over various infection scenaria, demonstrated that our approach can significantly outperform state-of-the-art designs (upto 58% in the best case). Moreover, our designs performed well with the DD decoder, which allows us to claim that test designs are transferable to other decoders. An open question is to extend the technique introduced in this chapter to other decoders, such as DD and belief propagation. It would also be interesting to extend these techniques to encompass more general infection models, in particular, to include correlated infection models. Performance guarantees for the heuristics introduced in this chapter is another open question.

4.7 Appendix and proofs

4.7.1 Theorem 4.1 proof: filling in the gaps

In the proof of Theorem 4.1 we claimed the following:

$$\mathbb{E}_{\mathbf{U}\setminus\{i\}} \prod_{t=1}^{T} \gamma_{t,i} \geq \prod_{t=1}^{T} \mathbb{E}_{\mathbf{U}\setminus\{i\}} \gamma_{t,i}.$$

where $\gamma_{t,i} \triangleq \left(1 - G_{ti} \prod_{\substack{j=1: \\ j\neq i}}^{N} (1 - G_{tj} U_j)\right)$. We prove this using the Fortuin–Kasteleyn–Ginibre (FKG) inequality (see [FKG71, Kem77, PC] or proof of Lemma 4 in [Ald19]), restated here for convenience.

Lemma 4.2 (FKG inequality)**.** *Consider a finite distributive lattice Γ with partial ordering $\prec$ and meet ($\wedge$) and join operators ($\vee$). Consider a probability measure μ on Γ that is log-supermodular, i.e.,*

$$\mu(a)\mu(b) \leq \mu(a \wedge b)\mu(a \vee b) \ \forall\ a, b \in \Gamma.$$

Then, any two functions f and g which are non-decreasing on Γ are positively correlated, i.e.,

$$\mathbb{E}_\mu(fg) \geq \mathbb{E}_\mu(f)\mathbb{E}_\mu(g).$$

Remark: Consider $\Gamma = \{0,1\}^N$ with partial ordering $\prec$, where $a \prec b$ if every coordinate of b is at least as large as a. When the meet and join operators coincide with logical AND and logical OR respectively, this is a distributive lattice. It can be verified that any product measure μ on Γ is log-supermodular. As a result, any two functions f and g which are non-decreasing on Γ are positively correlated, i.e., $\mathbb{E}_\mu(fg) \geq \mathbb{E}_\mu(f)\mathbb{E}_\mu(g)$. Consequently, given any M non-negative, non-decreasing functions $f_1, f_2, ..., f_M$ one could inductively apply FKG inequality to obtain

$$\mathbb{E}_\mu(\prod_{i=1}^{M} f_i) \geq \prod_{i=1}^{M} \mathbb{E}_\mu f_i. \tag{4.15}$$

Given (4.15) what remains to be shown is that each $\gamma_{t,i}(\mathbf{U})$ is non-negative and non-decreasing as a function of $\mathbf{U} \in \{0,1\}^N$. To see that it is non-negative is straight-forward – we have $G_{tj}U_j \geq 0$ and hence $(1 - G_{tj}U_j) \leq 1$. Therefore, the product $\prod_{\substack{j=1:\\ j\neq i}}^{N} (1 - G_{tj}U_j) \leq 1$ and the non-negativity follows. To see that $\gamma_{t,i}(\mathbf{U})$ is non-decreasing, we first consider $\mathbf{U} \prec \mathbf{U}'$, i.e., $U_j \leq U'_j \ \forall\ j$. Then we have $(1 - G_{tj}U_j) \geq (1 - G_{tj}U'_j)\ \forall\ t, j$ and $\prod_{\substack{j=1:\\ j\neq i}}^{N} (1 - G_{tj}U_j) \geq \prod_{\substack{j=1:\\ j\neq i}}^{N} (1 - G_{tj}U'_j)\ \forall\ t$. Thus, $\gamma_{t,i}(\mathbf{U}) \leq \gamma_{t,i}(\mathbf{U}')$ and $\gamma_{t,i}$ is non-decreasing. Applying (4.15), we have

$$\mathbb{E}_{\mathbf{U}\setminus\{i\}} \prod_{t=1}^{T} \gamma_{t,i} \geq \prod_{t=1}^{T} \mathbb{E}_{\mathbf{U}\setminus\{i\}} \gamma_{t,i}.$$

4.7.2 Computing the objective function $f(Q)$

Here we give a $O(N^2)$ algorithm to compute the objective function $f(Q)$ in (4.13). We assume $T \leq N$ so $T = O(N)$ throughout. We first restate the expression for $f(Q)$ in (4.14):

$$f(Q) = \sum_{i=1}^{N} (1 - p_i) \prod_{t=1}^{T} (1 - Q_{ti} \prod_{j=1, j\neq i}^{N} (1 - Q_{tj}p_j)).$$

Note that this can be rewritten as:

$$f(Q) = \sum_{i=1}^{N} (1 - p_i) F[i],$$

where the intermediate terms are defined as

$$F[i] \triangleq \prod_{t=1}^{T} (1 - Q_{ti} G[t,i])$$

and

$$G[t,i] \triangleq \prod_{j=1, j\neq i}^{N} (1 - Q_{tj}p_j)).$$

Thus, we first compute and store $G[t,i]\ \forall\ t, i$, which is then used to compute $F[i]\ \forall\ i$ in $O(N^2)$ time (assuming $T = O(N)$). Subsequently, $f(Q)$ can be computed from $F[i]$ in $O(N)$. Computing $G[t,i]$ takes $O(N^2)$ as one can first compute $H[t] \triangleq \prod_{j=1}^{N} (1 - Q_{tj}p_j))\ \forall\ t$ in $O(N^2)$ time and obtain $G[t,i] = H[t]/(1 - Q_{ti}p_i)$ in $O(N^2)$. The overall time complexity of computing $f(Q)$ is $O(N^2)$.

4.7.3 Expression for each partial derviative in $\nabla_Q f(Q)$

Here, we give an expression for the gradient $\nabla f(Q)$ by calculating each partial derivative $\frac{\partial f(Q)}{\partial Q_{lm}}$.

$$\begin{aligned}
\frac{\partial f(Q)}{\partial Q_{lm}} &= \frac{\partial}{\partial Q_{lm}} \sum_{i=1}^{N} (1-p_i) \prod_{t=1}^{T} \left(1 - Q_{ti} \prod_{\substack{j=1:\\ j\neq i}}^{N} (1 - Q_{tj} p_j) \right) \\
&= \frac{\partial}{\partial Q_{lm}} \sum_{i=1:i\neq m}^{N} (1-p_i) \prod_{t=1}^{T} \left(1 - Q_{ti} \prod_{\substack{j=1:\\ j\neq i}}^{N} (1 - Q_{tj} p_j) \right) \\
&\quad + (1-p_m) \frac{\partial}{\partial Q_{lm}} \prod_{t=1}^{T} \left(1 - Q_{tm} \prod_{\substack{j=1:\\ j\neq m}}^{N} (1 - Q_{tj} p_j) \right) \\
&\overset{(a)}{=} \sum_{i=1:i\neq m}^{N} (1-p_i) \frac{\partial}{\partial Q_{lm}} \left(1 - Q_{li} \prod_{\substack{j=1:\\ j\neq i}}^{N} (1 - Q_{lj} p_j) \right) \prod_{t=1:t\neq l}^{T} \left(1 - Q_{ti} \prod_{\substack{j=1:\\ j\neq i}}^{N} (1 - Q_{tj} p_j) \right) \\
&\quad + (1-p_m) \frac{\partial}{\partial Q_{lm}} \left(1 - Q_{lm} \prod_{\substack{j=1:\\ j\neq m}}^{N} (1 - Q_{lj} p_j) \right) \prod_{t=1:t\neq l}^{T} \left(1 - Q_{tm} \prod_{\substack{j=1:\\ j\neq m}}^{N} (1 - Q_{tj} p_j) \right) \\
&\overset{(b)}{=} \sum_{i=1:i\neq m}^{N} (1-p_i) \left(Q_{li} p_m \prod_{\substack{j=1:\\ j\neq i, j\neq m}}^{N} (1 - Q_{lj} p_j) \right) \prod_{t=1:t\neq l}^{T} \left(1 - Q_{ti} \prod_{\substack{j=1:\\ j\neq i}}^{N} (1 - Q_{tj} p_j) \right) \\
&\quad + (1-p_m) \left(- \prod_{\substack{j=1:\\ j\neq m}}^{N} (1 - Q_{lj} p_j) \right) \prod_{t=1:t\neq l}^{T} \left(1 - Q_{tm} \prod_{\substack{j=1:\\ j\neq m}}^{N} (1 - Q_{tj} p_j) \right), \qquad (4.16)
\end{aligned}$$

where in (a) we separate out the term corresponding to $t = l$ from the product term $\prod_{t=1}^{T}$ and apply the derivative in (b).

4.7.4 Computing $\nabla_Q f(Q)$

The computation of gradient follows a similar approach as the computation of the objective function $f(Q)$. We assume $T \leq N$ so $T = O(N)$ throughout. We first restate the expression for the gradient in (4.16):

$$\nabla_{Q_{lm}} f(Q) = \sum_{i=1:i\neq m}^{N} (1-p_i) \left(Q_{li} p_m \prod_{\substack{j=1:\\ j\neq i, j\neq m}}^{N} (1 - Q_{lj} p_j) \right) \prod_{t=1:t\neq l}^{T} \left(1 - Q_{ti} \prod_{\substack{j=1:\\ j\neq i}}^{N} (1 - Q_{tj} p_j) \right)$$
$$+ (1 - p_m) \left(- \prod_{\substack{j=1:\\ j\neq m}}^{N} (1 - Q_{lj} p_j) \right) \prod_{t=1:t\neq l}^{T} \left(1 - Q_{tm} \prod_{\substack{j=1:\\ j\neq m}}^{N} (1 - Q_{tj} p_j) \right)$$

As we did in the case of objective function computation, we first simplify and rewrite this in terms of intermediate terms:

$$\begin{aligned}
\nabla_{Q_{lm}} f(Q) &= \sum_{i=1:i\neq m}^{N} (1-p_i) \left(Q_{li} \frac{p_m}{1 - Q_{lm} p_m} G[l,i] \right) \prod_{t=1:t\neq l}^{T} (1 - Q_{ti} G[t,i]) \\
&\quad + (1 - p_m) \left(-G[l,m] \right) \prod_{t=1:t\neq l}^{T} (1 - Q_{tm} G[t,m]) \\
&= \frac{p_m}{1 - Q_{lm} p_m} \sum_{i=1:i\neq m}^{N} (1-p_i) \left(Q_{li} G[l,i] \right) F[l,i] \\
&\quad + (1 - p_m) \left(-G[l,m] \right) F[l,m] \\
&= \frac{p_m}{1 - Q_{lm} p_m} \left(\sum_{i=1}^{N} (1-p_i) Q_{li} G[l,i] F[l,i] - (1-p_m) Q_{lm} G[l,m] F[l,m] \right) \\
&\quad + (1 - p_m) \left(-G[l,m] \right) F[l,m] \\
&= \frac{p_m}{1 - Q_{lm} p_m} \sum_{i=1}^{N} (1-p_i) Q_{li} G[l,i] F[l,i] \\
&\quad - (1 - p_m) G[l,m] F[l,m] \left(\frac{1}{1 - Q_{lm} p_m} \right),
\end{aligned}$$

where the intermediate terms are

$$F[l,i] \triangleq \prod_{t=1:t\neq l}^{T} (1 - Q_{ti}G[t,i])$$

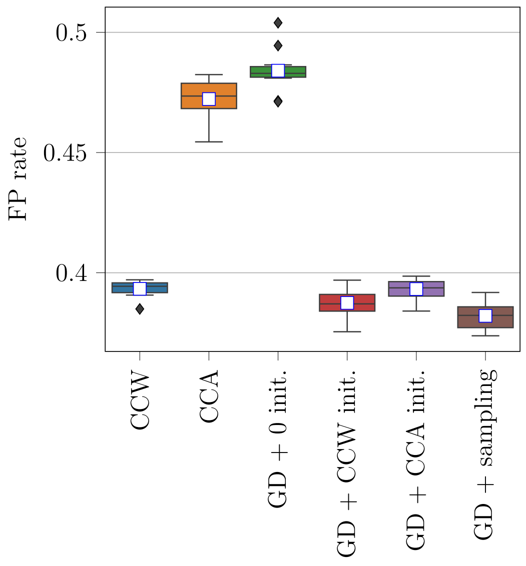
and

$$G[t,i] = \prod_{j=1,j\neq i}^{N} (1 - Q_{tj}p_j).$$

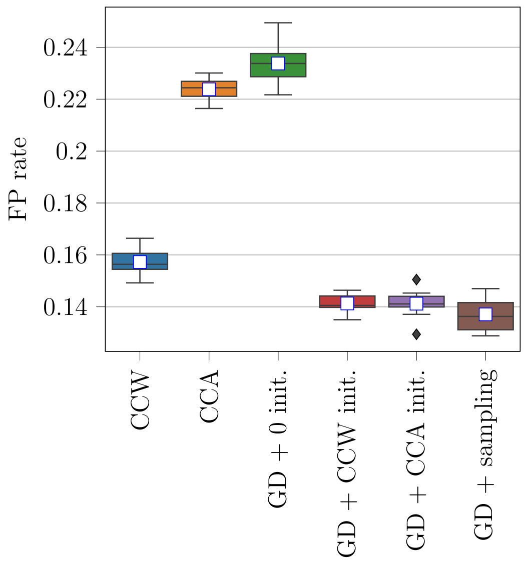
As we showed earlier, computing $G[t,i]\ \forall\ t,i$ can be done in $O(N^2)$ runtime complexity, and $F[l,i]$ can be obtained as $\frac{H[i]}{1-Q_{li}G[l,i]}$ where $H[i] \triangleq \prod_{t=1}^{T}(1 - Q_{ti}G[t,i])$. Clearly, $H[i]\ \forall\ i$ can be obtained once in $O(N^2)$ and reused to compute $F[l,i]\ \forall\ l,i$ in $O(N^2)$. Having computed F and G terms, one could again use a similar trick to precompute $J[l] \triangleq \sum_{i=1}(1-p_i)Q_{li}G[l,i]F[l,i]\ \forall\ l$ in $O(N^2)$. With this, one could now compute each gradient term $\nabla_{Q_{lm}}$ in $O(1)$ thus giving an overall time complexity $O(N^2)$.

4.7.5 Additional numerical results

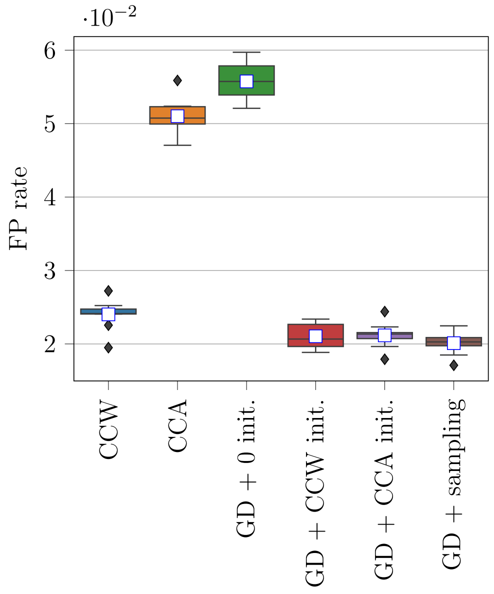
In Figure 4.4-Figure 4.7, we show box plots that provides additional details on the distribution of the error rates illustrated in Section 4.5. The box edges indicate the upper and lower quartiles, the small white box plots the mean, the diamonds are deemed as outlier points and the whiskers indicate the min and the max excluding the outlier points.



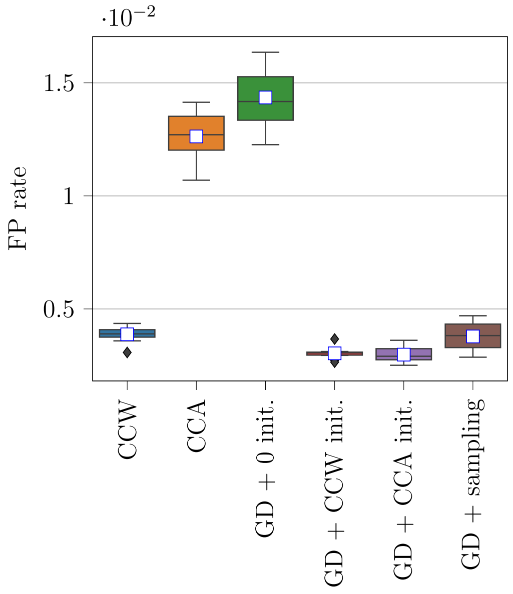
(a) $T = 100$.



(b) $T = 200$.

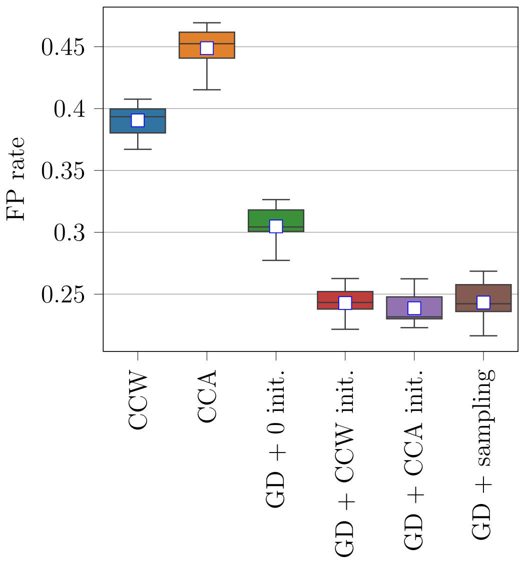


(c) $T = 400$.

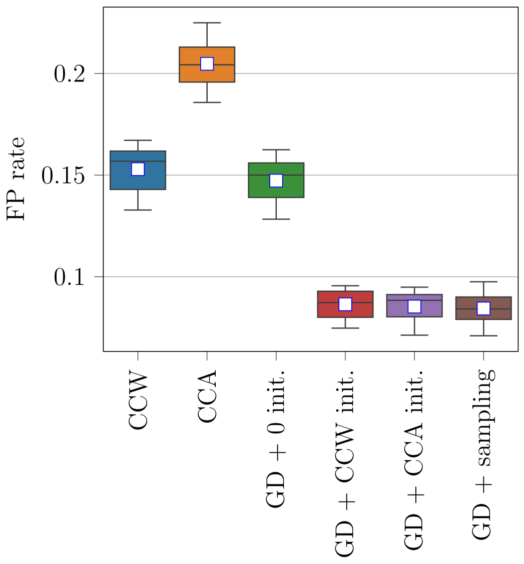


(d) $T = 600$.

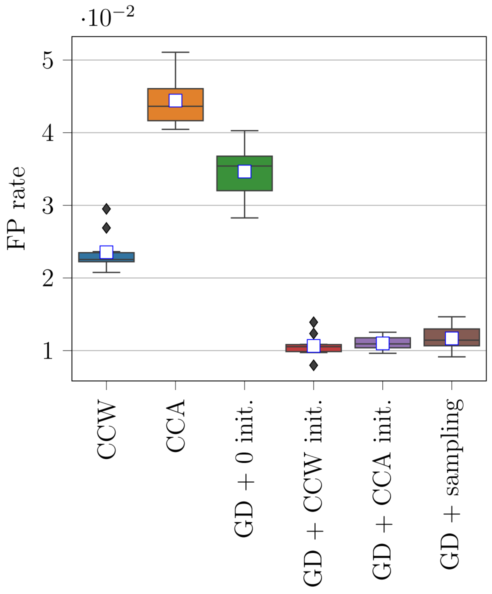
Figure 4.4: Summary of the distribution of error rates plotted in Figure 4.2a



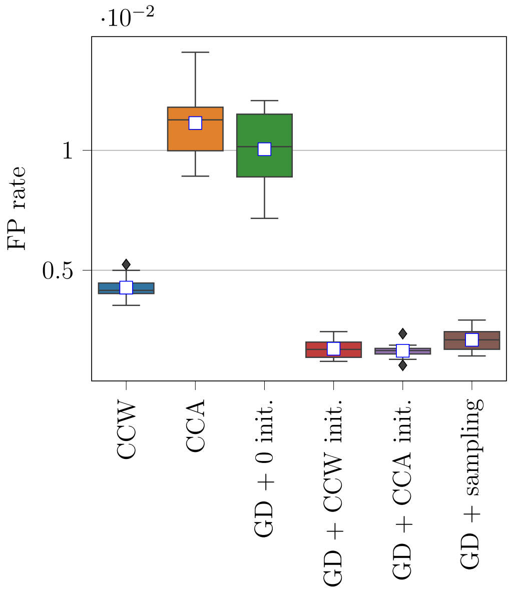
(a) $T = 100$.



(b) $T = 200$.

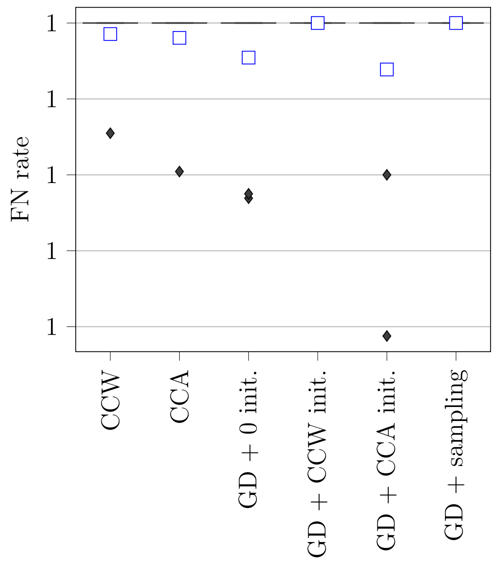


(c) $T = 400$.

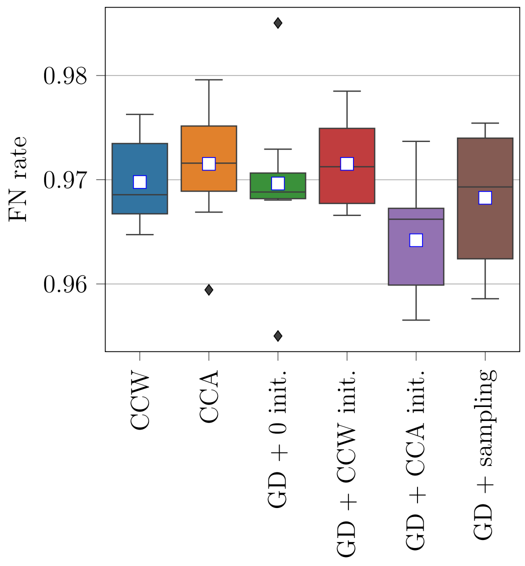


(d) $T = 600$.

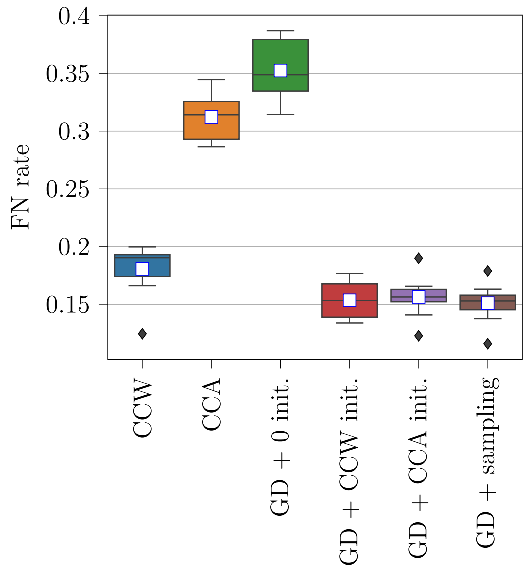
Figure 4.5: Summary of the distribution of error rates plotted in Figure 4.2b



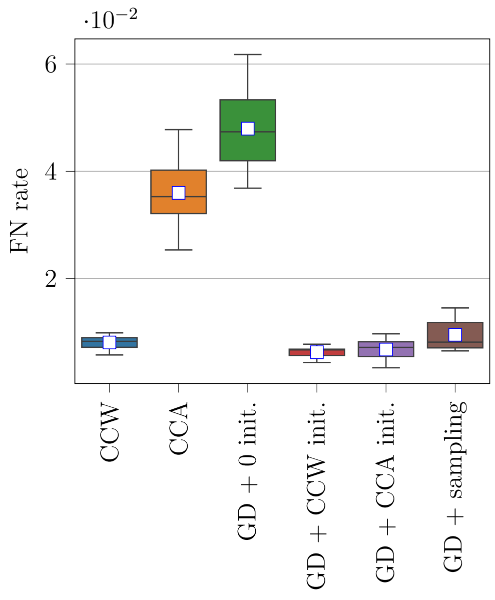
(a) $T = 100$.



(b) $T = 200$.

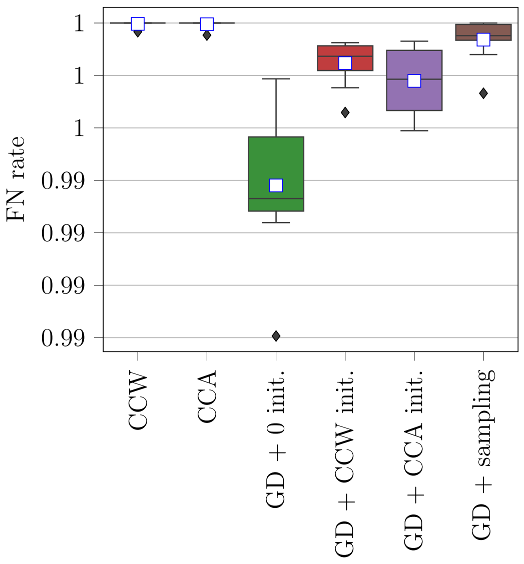


(c) $T = 400$.

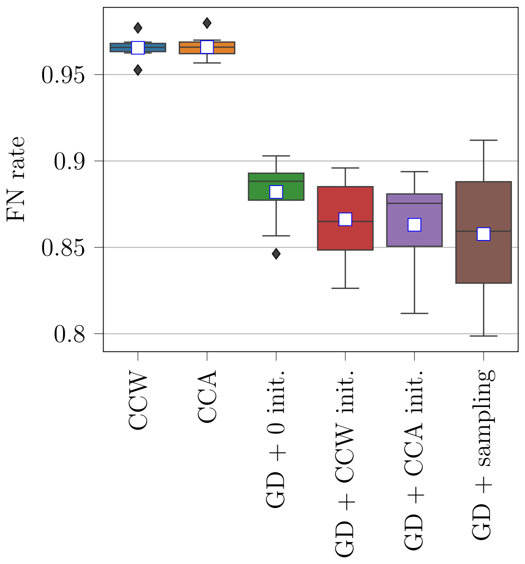


(d) $T = 600$.

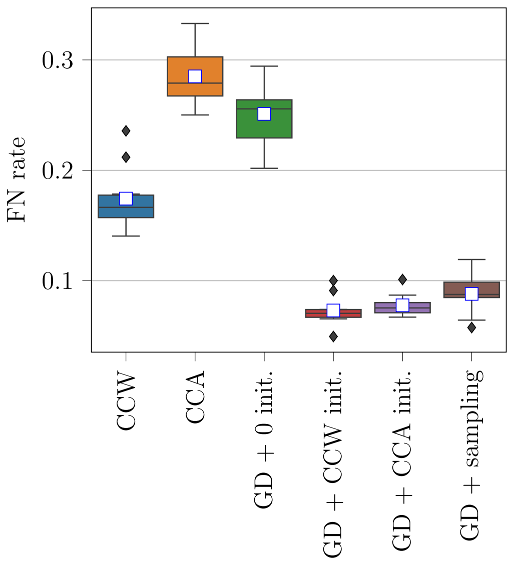
Figure 4.6: Summary of the distribution of error rates plotted in Figure 4.3a



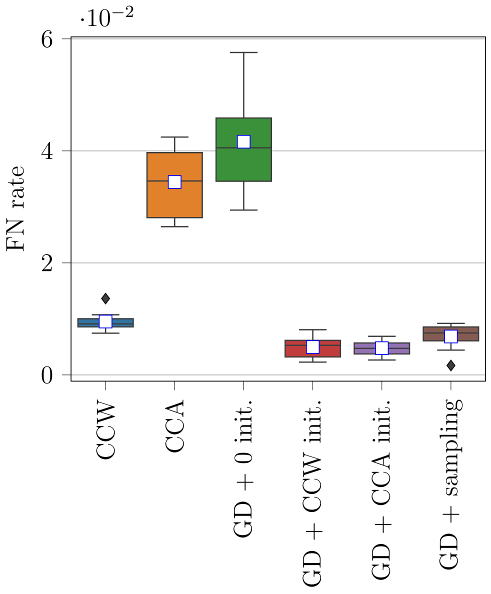
(a) $T = 100$.



(b) $T = 200$.



(c) $T = 400$.



(d) $T = 600$.

Figure 4.7: Summary of the distribution of error rates plotted in Figure 4.3b

CHAPTER 5

Static community infection model

Summary: *In this chapter, we study group testing in the presence of correlations, where the correlation is induced by a community structure. We propose algorithms, both in terms of designing tests as well as designing decoders, that leverage a known community structure to make group testing more efficient.*

In the previous chapter, we improved static group testing algorithms under an independent model of infection. In this chapter, we build group testing algorithms around an idea "whose time has come": we propose to leverage a known community structure to make group testing more efficient. Although traditionally the work in group testing assumes "independent" infections, we note that today it is totally feasible to keep track of community structure - several apps are already doing so [Tra20a, Tra20b]. Moreover, our approach is well aligned with the need for independent grassroots testing (schools testing their students, companies their workers) where the community structure is explicit (shared classrooms, shared common spaces).

We find that taking into account the community structure can reduce the number of tests we need significantly below the well known combinatorial bound [AJS19], the best we can hope for when not taking this structure into account. Moreover, it enlarges the regime where group testing can offer benefits over individual testing. Indeed, a limitation of group testing is that it does not offer benefits when k grows linearly with n [HHW81, Ung60]. Taking into account the community structure allows to identify and remove from the population large groups of infected members, thus reducing their proportion and converting a linear to a sparse regime identification. However, we also find that, although our algorithms do not need assume knowledge of the infection probabilities, they do need to correctly know the community structure, and in particular, the community overlaps: not taking into account the overlaps (assuming communities are disconnected) can deteriorate the performance. Note that the results in this work admit non-overlapping communities as a special case. Our main contributions in this chapter include:

• We derive a lower bound on the number of tests, that generalizes the counting bound [AJS19] to overlapping communities.

• We propose a new adaptive algorithm that requires fewer tests than traditional adaptive algorithms to recover the infection status of all individuals without error.

• We propose two nonadaptive algorithms that leverage the community structure to improve reliability over traditional nonadaptive testing. One leverages the structure at the encoder side with a novel test design, while the other one accounts for it at the decoder side with a new decoding algorithm that is based on loopy belief propagation and is generic enough to work on any structure.

The chapter is organized as follows: we give known results in Section 5.1, our model in Section 5.2, the lower bound in Section 5.3, and non-adaptive algorithms in Sections 5.4 and 5.5. Our numerical evaluation is in Section 5.6.

5.1 Background

As detailed in the previous chapter, traditional group testing assumes a population of n members out of which some are infected. Two infection models are considered: (i) in the *combinatorial model*, there is a fixed number of infected members k, selected uniformly at random among all sets of size k; (ii) in the *probabilistic model*, each item is infected independently of all others with probability p, so that the expected number of infected members is $\bar{k} = np$.

A group test τ takes as input samples from n_τ individuals, pools them together and outputs a single value: positive if any one of the samples is infected, and negative if none is infected. More precisely, let $U_i = 1$ when individual i is infected and 0 otherwise. y_τ takes a binary value calculated as $y_\tau = \bigvee_{i \in \delta_\tau} U_i$,

where $\bigvee$ stands for the `OR` operator (disjunction) and δ_τ is the group of people participating in the test.

The performance of a group testing algorithm is measured by the number of group tests $T = T(n)$ needed to identify the infected members (for the probabilistic model, the expected number of tests needed). Several setups have been explored in the literature, that include:

• *Adaptive vs. non-adaptive testing*: In adaptive testing, we use the outcome of previous tests to decide what tests to perform next. An example of adaptive testing is *binary splitting*, which implements a form of binary search and is known to be optimal when the number of infected members is unknown. Non-adaptive testing constructs, in advance, a *test matrix* $G \in \{0, 1\}^{T \times n}$ where each row corresponds to one test, each column to one member, and the non-zero elements in each row determine the set δ_τ. Although adaptive testing uses less tests than non-adaptive, non-adaptive testing is more practical as all tests can be executed in parallel.

• *Scaling regimes of operation*: assume $k = \Theta(n^\alpha)$, we say we operate in the linear regime if $\alpha = 1$; in the sparse regime if $0 \leq \alpha < 1$; in the very sparse regime if k is constant.

The following are well established performance results (see [AJS19, Joh17, DH93] and references therein):

• Since T tests allow to distinguish among 2^T combinations of test outputs, we need $T \geq \log_2 \binom{n}{k}$ to identify k randomly infected people out of n. This is known as the counting bound [AJS19, Joh17, DH93] and implies that we cannot use less than $T = O(k \log \frac{n}{k})$ tests.

• Noiseless adaptive testing allows to achieve the counting bound for $k = \Theta(n^\alpha)$ and $0 \leq \alpha < 1$; for non-adaptive testing, this is also true of $0 \leq \alpha < 0.409$, if we allow a vanishing (with n) error [AJS19, CGH20a, CGH20b].

• In the linear regime ($\alpha = 1$), group testing offers little benefits over individual testing. In particular, if the infection rate k/n is more than 0.38, group testing does not use fewer tests than one-to-one (individual) testing unless high identification-error rates are acceptable [HHW81, Ung60, RC00].

5.2 Model and Notation

5.2.1 Community model

This chapter extends the above results by assuming an overlapping community structure: members may belong to one or more communities—hence they are infected according to new combinatorial and probabilistic models that are slightly different from the traditional ones and depend on how the communities overlap (Section 5.2.2). Given these new infection models, new lower bounds can be derived (Section 5.3) for both the combinatorial and probabilistic case. Our analysis shows that these bounds can be significantly lower than the above-mentioned counting bounds.

More formally, we assume that all members of the entire population $V = \{1, 2, \cdots, n\}$ are organized in a known structure, which can be perceived in the form of a hypergraph $\mathcal{G}(V, E)$: each vertex $v \in V$ corresponds to an individual, that we simply call a *member*, and each edge $e \in E$ indicates which members belong to the same *community*. Since $\mathcal{G}$ is a hypergraph, an edge may connect any number of vertices; hence, a member may belong to one or more communities. The number of communities that a member belongs to is called the *degree* of the member. Sometimes, we need to reason about the number of communities in the structure and the community size (i.e. the number of members in a community). Instead of the classic cardinality notation, we use the following for brevity: $|E| = F$, $|V_e| = M_j$, and $M_j = M$ in the symmetric case where all communities have the same size.

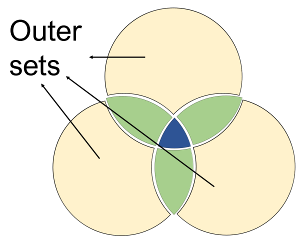


Figure 5.1: Example illustrating outer sets.

The hypergraph $\mathcal{G}$ may be decomposed into connected components, where each component $C(V_C, E_C)$ is a sub-hypergraph. For component C let D_C contain the nonempty subsets of the standard partition of the hyperedges in E_C;

in the example of Fig. 5.1, E_C consists of 3 hyperedges, and D_C contains 7 disjoint sets.

For each set $d \in D_C$, let V_d denote the set of members it contains. Because of the partition, all members of V_d belong to the same community (or set of communities), which we denote with $E(V_d)$—hence, they all have the same degree. As described in the next section, these members get infected according to some common infection principle. We distinguish 2 kinds of sets in D_C: (a) the "outer" sets: $D_{C,out} \triangleq \{d \in D_C : \nexists b \in D_C \text{ s.t. } E(V_b) \subset E(V_d)\}$, and (b) the "inner" sets: $D_{C,in} \triangleq D_C \setminus D_{C,out}$. Fig. 5.1 illustrates the 3 outer (yellow) and 4 inner (green and blue) sets. Note that the members of inner sets have always a higher degree than those of the outer sets.

5.2.2 Infection models

Let $\mathcal{K}(set)$ return the subset of infected elements of any *set* of members, communities or components. We consider the following infection models, that parallel the ones in the traditional setup of Section 5.1.

• **Combinatorial Model (I).** k_f of the communities have at least one infected member (we will call these *infected communities*). The rest of the communities have no infected members. Any combination of infected communities has the same chance of occurring. In each infected community, there are $k_m^j = |\mathcal{K}(V_e)|$ infected members, out of which $k_m^{\mathcal{I}} = |\mathcal{K}\left(\cap_{e\in\mathcal{I}} V_e\right)|$ (with $\cap_{e\in\mathcal{I}} V_e \neq \varnothing$) are shared with a subset of communities $\mathcal{I}$. The infected communities (resp. infected community members) are chosen uniformly at random out of all communities (resp. members of the same community).

- **Probabilistic Model (II).** A community e becomes infected i.i.d. with probability q. If a member v of an infected community e belongs *only* to that community (i.e. has degree 1), then it becomes infected w.p. $p = p_j$, independently from the other members (and other communities).

Also, if v belongs to a subset of infected communities $\mathcal{K}(\mathcal{I}) \subseteq \mathcal{K}(E)$, it is considered to be infected by either of these communities. So, given all the infection rates of these communities $\{p_j : e \in \mathcal{K}(\mathcal{I})\}$, we say that i becomes infected w.p.:

$p = 1 - \prod_{j\in\mathcal{K}(\mathcal{I})}(1 - p_j)$. Note that since each member gets infected by either of the infected communities it belongs to, the product of the RHS term above becomes smaller as $|\mathcal{K}(\mathcal{I})|$ increases. Last, if v does not belong to an infected community, then $p = 0$.

We make two remarks: First, although the communities are infected independently, their structure causes a dependent infection model; in fact, the way communities overlap determines the infection probability of their shared members. Second, our model captures situations where infection is determined by participation in a community rather than the status of community members. Albeit simplistic, we think that this model can be useful in real pandemics. Since the exact community structure of the entire population of a country or a continent can never be known to the test designer, we expect that graph $\mathcal{G}$ only partially describes the reality: there might be members that do not belong to V, yet interact with them in unknown ways, or there might be communities that are simply not captured due to unknown member interactions. In such a case, assuming that communities become infected independently seems a simple yet reasonable model to use. However, once a few communities in $\mathcal{G}$ get infected, we expect that the infection probability of a member will increase with the number of infected communities it belongs to, which is captured by our model in the computation of p.

Non-overlapping communities A special case of our community framework is when the communities have no overlap; this scenario is investigated in in our work [NRG21], where algorithms that take into account the non-overlapping structure are explored. In our experiments, however, these algorithms do not always perform well when communities overlap (e.g. see Fig. 5.3); in fact, there are cases where they perform worse than traditional group testing.

5.3 Lower bound on the number of tests

We compute the minimum number of tests needed to identify all infected members under the zero-error criterion in both community models (I) and (II).

Theorem 5.1 (Community bound for combinatorial model (I)). *Any algorithm that identifies all k infected members without error requires a number of tests T satisfying:*

$$T \geq \log_2 \binom{F}{k_f} + \sum_{C \in \mathcal{G}} \sum_{d \in D_C} \log_2 \binom{|V_d|}{|\mathcal{K}(V_d)|}, \tag{5.1}$$

where $|V_d|$ is the set of members of each disjoint set in D_C and $|\mathcal{K}(V_d)|$ are the infected members of that set.

Observation: Consider a usual epidemic scenario, where the population is composed of a large number of communities with members that have close contacts (e.g. relatives, work colleagues, students who attend the same classes, etc.). In such a case, one should expect that most members of infected communities are infected (i.e. $k_m^j \approx M_j$), even though the number of infected communities k_f and the overall number of infected members k may still follow a sparse regime (i.e., $k_f = \Theta(F^{\alpha_f})$ and $k = \Theta(n^{\alpha})$ for $\alpha_f, \alpha \in [0, 1)$).

Theorem 5.1 shows the significant benefit of taking the community structure into account in the test design: the community bound increases almost *linearly* with k_f, as opposed to k, which is what happens with the traditional counting bound (that does not account for any community structure). This is due to the second term of Equation (5.1) tending to 0 and $\log_2 \binom{F}{k_f} \sim k_f \log_2 \frac{F}{k_f} \sim (1-\alpha_f) k_f \log_2 F$.

A similar bound exists for the probabilistic model (II). Consider a component $C \in \mathcal{G}$. Let $\mathbf{V}$ be the indicator random vector for the infection stat of all communities in C, and $\mathcal{X}_C = \{0,1\}^{|C|}$ be the set of all possible such indicator vectors. Note that since each community becomes infected w.p. q, $\Pr(\mathbf{V} = \mathbf{x}_C) = q^{\lambda}(1-q)^{|C|-\lambda}$, where λ is the weight of $\mathbf{x}_C$. Also, the value of $\mathbf{x}_C$ determines the infection probabilities p of each member $i \in V_C$.

5.4 Algorithms

In this section, we provide group-testing algorithms for the noiseless case that leverage the community structure. We start from adaptive algorithms and then proceed to non-adaptive.

5.4.1 Adaptive algorithm

Algorithm 5.1 describes our adaptive algorithm. It is built on top of traditional adaptive testing,which we will generally denote as *AdaptiveTest*(). *AdaptiveTest*() is an abstraction of any existing (or future) adaptive algorithm that assumes independent infections. We distinguish 2 different inputs for *AdaptiveTest*(): (a) a set of members; or (b) a set of *mixed samples*. A mixed sample is created by pooling together samples from multiple members. For example, mixed sample $z(r_d)$ is an pooled sample of some representative members r_d from disjoint set d. Because we only care whether a mixed sample is positive or not, we can treat it in the same way as an individual sample—hence use group testing to identify the state of mixed samples as we do for individuals.

Part 1: For each component of the graph $\mathcal{G}$, we first identify the outer sets $D_{C,out}$. Then, from each outer set d, we select a representative subset of members r_d, whose samples are pooled together into a mixed sample $z(r_d)$. There can be many selection methods for *SelectRepresentatives*(); however, we typically use uniform (random) sampling without replacement. Finally, we determine the state of all mixed samples (line 4).

Part 2: We treat $\hat{U}_{z(r_d)}$ as a rough estimate of the infection regime inside each set d: if $\hat{U}_{z(r_d)}$ is positive, we consider d to be heavily infected and we individually test its members

Algorithm 5.1 Adaptive Community Testing ($\mathcal{G}\left(V, E\right)$)

$\hat{U}_i$ is the estimated infection state of member i ("+" or "-").

$\hat{U}_z$ is the estimated infection state of a mixed sample z.

```
 1: for $d \in D_{C,out}$, $\forall C \in \mathcal{G}$ do
 2:     $r_d \leftarrow \mathit{SelectRepresentatives}\left(V_d\right)$
 3: $\left\{\hat{U}_{z(r_d)}\right\} \leftarrow \mathit{AdaptiveTest}\left(\{z(r_d\}\right)$
 4: Set $A := \varnothing$
 5: for $C \in \mathcal{G}$ do
 6:     for $d \in D_{C,out}$ do
 7:         if $\hat{U}_{z(r_d)} =$ "positive" then
 8:             Individually test $V_d$: $\hat{U}_i \leftarrow U_i, \forall i \in V_d$.
 9:             $\hat{p}_d \leftarrow 1/|V_d| \cdot \sum_{v \in V_d} \mathbb{1}_{\left\{\hat{U}_i = \text{'positive'}\right\}}$
10:         else
11:             $A \leftarrow A \cup \{i : i \in V_d\}$
12:     for $b \in D_{C,in}$ (in increasing order of degree) do
13:         if $\exists d \in D_C$ s.t. $E(V_d) \subset E(V_b)$ & $\hat{p}_d > \theta$ then
14:             Individually test $V_b$: $\hat{U}_i \leftarrow U_i, \forall i \in V_b$.
15:             $\hat{p}_b \leftarrow 1/|V_b| \cdot \sum_{v \in V_b} \mathbb{1}_{\left\{\hat{U}_i = \text{'positive'}\right\}}$
16:         else
17:             $A \leftarrow A \cup \{i : i \in V_b\}$
18: $\left\{\hat{U}_i : i \in A\right\} = \mathit{AdaptiveTest}\left(A\right)$
19: return $\left[\hat{U}_1, \ldots, \hat{U}_n\right]$
```

(line 9); otherwise, we consider it lightly infected and we include its members in set A (line 12). For our rough estimate of the infection regime to be a good one, we choose the number of representatives based on some prior information about infection rate of each outer set; for example if $p_j < 0.38$ then only one representative is enough, otherwise pooling together the entire set is one's best option. Note that the exact knowledge of p_j and a rough prior may be easily acquired. For example, in realistic scenarios, where the infection rates are not expected to be very low inside the communities, pooling together the entire outer set is a good heuristic.

Due to individually testing the heavily-infected outer sets, we obtain more accurate estimates of their infection rates, $\hat{p}_d$, by computing the average proportion of infected members (line 10). We use these estimates to decide how to test the inner sets of the component: if an outer set d exists whose members belong to a subset of communities in $E(V_b)$ and its estimated infection rate $\hat{p}_d$ is above a threshold θ, then members of V_b are tested individually (line 17) and a new estimate $\hat{p}_b$ for the infection rate of that set is computed (line 18). Else, members of V_b are included in set A. Our rationale follows the infection model described in Section 5.2.2, which implies that the infection probability of the members of an inner set b will always be at least equal to the infection probability of the members (of an outer set d) whose community(ies) are a subset of $E(V_d)$. Hence, if an outer set is heavily infected then a corresponding inner set will be heavily infected, too. In our experiments, we numerically examine the impact of θ.

Finally, we test all members of set A that are not tested individually (because infection probability is presumably low) using traditional group testing (line 23).

5.4.2 Non-adaptive algorithms

For simplicity, we describe our non-adaptive algorithm using the symmetric case.

Test Matrix

We divide the $(T_1 + T_2) \times n$ matrix G into two sub-matrices G_1 and G_2 of sizes $T_1 \times n$ and $T_2 \times n$.

$\triangleright$ The sub-matrix G_1 identifies the non-infected outer sets using one mixed sample for each outer set (Section 5.2.1). If the number of tests available is large, we set T_1 to be the number of outer sets, i.e., we use one test for each outer set; otherwise, in sparse k_f regimes, T_1 can be closer to $\mathcal{O}(k_f \log \frac{F}{k_f})$.

$\triangleright$ Assume that $T_2 = \frac{n}{c}$, for some constant c. The sub-matrix G_2 of size $T_2 \times n$ has one "1" in each column (each of the n member participates in one test) and c "1"s in each row (each test pools together c members); equivalently, G_2 is a concatenation of c identity matrices I_{T_2}, i.e., $G_2 = [I_{T_2}\ I_{T_2} \cdots I_{T_2}]$. For $c = 1$, this reduces to individual testing. The design of G_2 amounts to deciding which members are placed in the same test. We propose that no two members from the same outer set are placed in the same test and that members from the same inner set are placed in the same test (c members in each test).

Decoding

We use the test outcomes of G_1 to identify the non-infected outer sets and proceed to remove the corresponding columns from G_2. We next use the remaining columns of G_2 and combinatorial orthogonal matching pursuit (COMP) to identify the infected members, namely: (i) A member is identified as non-infected if it is included in at least one negative test in G_2. (ii) All other members, that are only included in positive tests in G_2, are identified as infected.

Intuition

Suppose infected communities have a large percentage (say $> 40\%$) of infected members. The idea is that pooling together multiple highly correlated items in the same test (such as people in the same outer/inner set) enables COMP to mark all these items as non-defective in case of a negative result.

Example

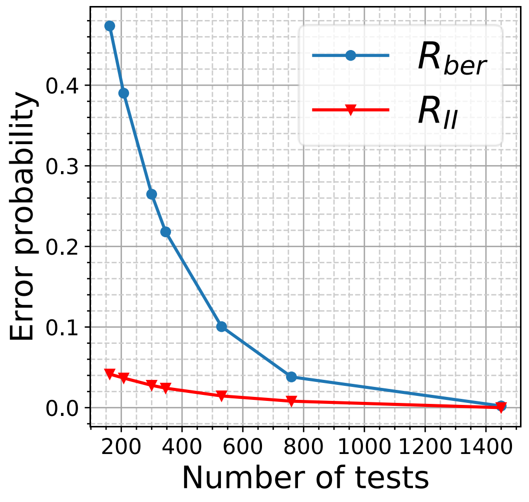
We here illustrate for a special case our proposed design for matrix G_2 and the resulting error rate our algorithm achieves. Assume that we have F communities, where $2F_o$ communities pairwise overlap (each community overlaps with exactly one other community) and the remaining $F - 2F_o$ communities do not overlap with any other community. Assume each community has M members, and overlapping communities share M_o members. We construct the sub-matrix G_2 of size $T_2 \times n$ as in the following example that uses $F = 6$, $F_o = 2$, $M = 3$, $M_o = 1$:

$$\mathbf{G}_2 = \begin{bmatrix} I_3 & & & & I_3 & & & \\ & I_2 & & & & I_2 & & \\ & & I_1 & & & & I_1 & \\ & & & I_2 & & & & I_2 \end{bmatrix}.$$

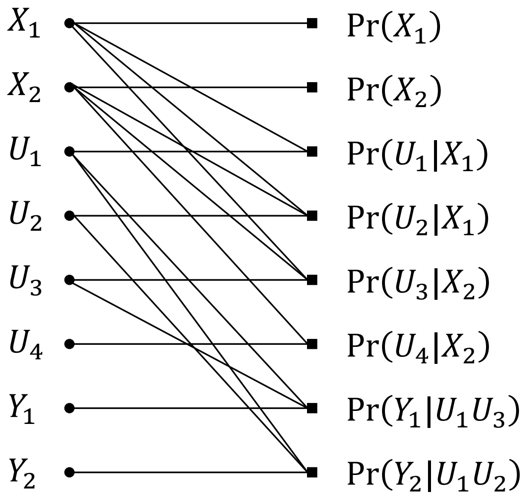
This matrix starts with $b_1 = \frac{F-2F_o}{c}$ block-rows that each contains c identity matrices I_M, one corresponding to each non-overlapping community. We then have $b_2 = \frac{F_o}{c}$ block-rows each containing c identity matrices I_{2M-M_o}, one for each pair of overlapping communities. Each I_{2M-M_o} matrix contains three matrices I_{M-M_o}, I_{M_o}, and I_{M-M_o} corresponding to the members that belong only in one of the communities, or in both. Note that $F = (b_1 + 2b_2)c$ and $T_2 = b_1 M + b_2(2M - M_o)$.

Error Rate

Note that our decoding strategy leads to zero FN errors. The following lemma provides an analysis of the error (FP) rate for the design of G_2 in the example which is defined as: $R(\text{error}) \triangleq {}^1\!/_n \cdot |\{i : \hat{U}_i \neq U_i\}|$. We provide the expected error rate for only the probabilistic model (II) for the purpose of comparison with traditional Bernoulli design in Fig. 5.2(a).



(a)



(b)

Figure 5.2: (a). Error rate for Bernoulli design vs G_1G_2 design for the example. (b). An example of factor graph.

Lemma 5.1. *For G_2 as in the example, the error rate is calculated for the probability model (II) as:*

$$R_{II}(error) = \frac{1}{n}\left[\left(1-(1-pq)^{c-1}\right)\cdot N_1 + \left(1-(1-pq)^{2(c-1)}\right)\cdot N_2\right], \tag{5.2}$$

where N_1 and N_2 are the expected number of non-overlapped and overlapped members in infected communities that are non-infected, respectively, and can be obtained as

$$N_1 = (F-2F_o)q(1-p)M + 2F_oq(1-p)(M-M_o)$$
$$N_2 = F_o\left(1-(1-q)^2\right)(1-p)M_o.$$

The error rate of traditional group testing using Bernoulli design (with parameter $\frac{1}{k}$) and COMP decoding has an error rate of $R_{\text{tradition}}(\text{error}) = {}^1/_n \cdot (n-k)\left(1-1/k(1-1/k)^k\right)^T$. Fig. 5.2(a) depicts $R(\text{error})$ for parameters $F=150$, $F_o=60$, $M=10$, $M_o=2$, $q=0.2$, and $p=0.2$.

5.5 Loopy belief propagation decoder

Apart from COMP, we also use loopy belief propogation (LBP [KFL01]) to infer the infection status of the individual (and communities). LBP forms an estimate of the posterior probability that an individual (or a community) is infected, given the test results. This estimate is exact when the underlying factor graph describing the joint distribution is a tree, however this is rarely the case. Nevertheless, it is an algorithm of practical importance and has achieved success on a variety of applications. Also, LBP offers soft information (posterior distributions), which is more useful than hard decisions in the context of disease-spread management.

We use LBP for our probabilistic model, because it is fast and can be easily configured to take into account the community structure. Many inference algorithms exist that estimate the posterior marginals, some of which have also been employed for group testing. For example, GAMP [ZRB20] and Monte-Carlo sampling [CTV20] yield more accurate decoders. The focus of this work is to examine whether benefits from accounting for the community structure (both at the test design and the decoder) exist; hence we think that considering a simple (possibly sub-optimal) decoder based on LBP is a good first step.

We next describe the factor graph and the belief propagation update rules for our probabilistic model (II). Let the infection status of each community j be $V_j \sim \text{Ber}(q)$.

Moreover, let S_v denote the set of communities that U_i belongs to. Then:

$$\Pr(V_1,..., V_F, U_1, ..., U_n, Y_1, ..., Y_T) = \prod_{j=1}^{F} \Pr(V_j) \prod_{i=1}^{n} \Pr(U_i | V_{S_v}) \prod_{\tau=1}^{T} \Pr(Y_\tau | U_{\delta_\tau}), \tag{5.3}$$

where δ_τ is the group of people included in test τ. Equation (5.3) can be represented by a factor graph, where the variable nodes correspond to the variables V_j, U_i, Y_τ and the factor nodes correspond to $\Pr(V_j), \Pr(U_i | V_{S_v}), \Pr(Y_\tau | U_{\delta_\tau})$;

Given the result of each test is y_τ, i.e., $Y_\tau = y_\tau$, LBP estimates the marginals $\Pr(V_j = v|Y_1 = y_1, ..., Y_T = y_T)$ and $\Pr(U_i = u|Y_1 = y_1, ..., Y_T = y_T)$, by iteratively exchanging

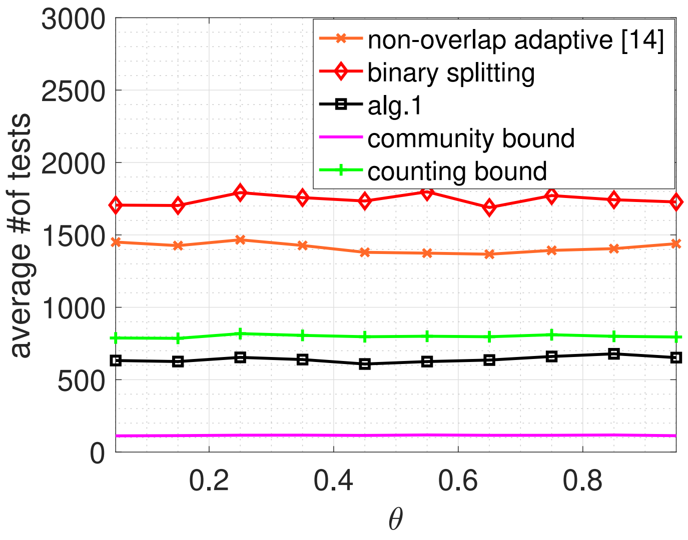


Figure 5.3: Average number of tests comparison of various adaptive algorithms and combinatorial bound.

messages across the variable and factor nodes. The messages are viewed as *beliefs* about that variable or distributions (a local estimate of Pr(variable|observations)). Since all random variables are binary, each message is a 2-dimensional vector.

We use the factor graph framework from [KFL01] to compute the messages: Variable nodes Y_τ continually transmit the message $[0, 1]$ if $y_\tau = 1$ and $[1, 0]$ if $y_\tau = 0$ on its incident edge, at every iteration. Each other variable node (V_j and U_i) uses the following rule: for incident edge ϵ, the node computes the elementwise product of the messages from every other incident edge and transmits this along ϵ. For the factor node messages, we derive closed-form expressions for the sum-product update rules (akin to equation (6) in [KFL01]). The exact messages are described in the Appendix.

5.6 Numerical evaluation

In this section, we evaluate the benefit of accounting for the community structure, in terms of error rate and number of tests required, using 100 random structures, each having $n = 3000$

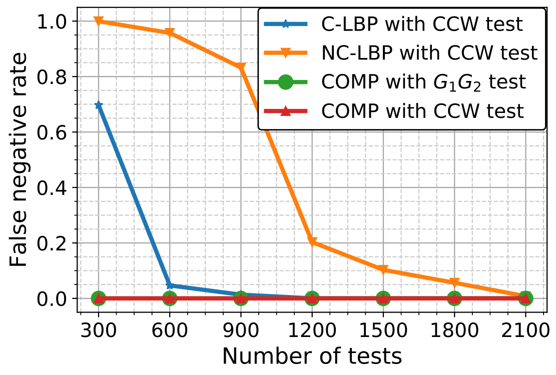


Figure 5.4: FN rate comparison of various non-adaptive test designs with corresponding decoding algorithms.

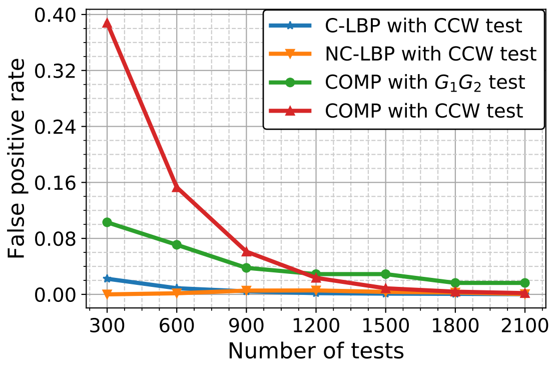


Figure 5.5: FP rate comparison of various non-adaptive test designs with corresponding decoding algorithms.

members participating in about 200 overlapping communities.

Experimental setup. We generate each structure using the following rules: the size of each community is selected uniformly at random from the range $[15, 25]$, and each member is randomly allocated in at most 4 communities (according to a geometric distribution). Then, the members become infected according to the probabilistic model (II): each community e gets infected w.p. $q = 0.05$; and if infected, then its infection rate p_j is randomly chosen from the interval $[0.3, 0.9]$. We remark that our experimental setup yields a linear infection regime; the fraction of infected members about 5% overall. We preferred such a setup in order to stress the performance of our algorithms, as we know that group testing generally shows less benefits in linear regimes.

For the adaptive algorithms, we compare: the binary splitting algorithm (BSA) [AJS19], which is the best traditional alternative when the number of infected members is unknown; the algorithm proposed in [NRG21] that considers communities but no overlap; and Alg. 5.1 with BSA in the place of $AdaptiveTest()$.

For the non-adaptive test matrix designs, we compare: G_1G_2, our proposed test design in Section 5.4.2; and CCW, constant-column-weight algorithms, where each item is included in a fixed number w of tests selected uniformly at random. w is assumed to be of the form $w = \alpha\frac{T}{k}$, where k is an estimate of the number of defectives in the population. We exhaustively search to find the best value of $\alpha \in [0, 1]$.

We also compare LBP and COMP decoding: *C-LBP* is our proposed algorithm in section 5.5, that takes into account the community structure. *NC-LBP*, does not take into account the community structure, i.e., assumes that each individual is i.i.d infected with the same probability p_{iid}. *COMP*, described in [AJS19], has a zero FN probability by design.

Results.

(i) *Adaptive test designs.* For each community structure, we measured the number of tests needed by each adaptive algorithm to achieve zero-error identification. Since Alg. 5.1 depends on θ, the threshold used at line 16, we wanted to evaluate its performance of various values of θ. Figure 5.3 depicts the average performance of our algorithm (for each θ, we average over 100 randomly generated structures). Alg. 5.1 was proved resilient to the choice of θ and needed on average $> 60\%$ fewer tests than the other algorithms. Its performance was also better than the counting bound, which is our best hope with traditional group testing. Our findings were similar in sparse infection regimes as well, and there were cases where our algorithm performed close to the community bound [NGF20].

(ii) *Non adaptive test designs.* In our experiments, we measured the FN/FP rates achieved by the non-adaptive test designs and the corresponding decoders. Fig. 5.4 and Fig. 5.5 depict FN and FP rates as a function of $T \in [300, 2100]$, respectively. The key takeaways are as follows:

• C-LBP with CCW attains zero FP and FN at 1200 tests while COMP and NC-LBP with CCW (which are agnostic to the community structure) attain zero FP and FN only at 1800 and 2100 tests respectively. This illustrates potential benefits of making the decoder aware of the community structure.

• If we desire a zero FN rate (or if we would like to use a simple decoder) and we are constrained to use less than 1000 tests, the G_1G_2 test design with COMP gives the lowest

FP rates. This illustrates the benefit of designing tests matrices that take into account the community structure.

5.7 Conclusions and Open Questions

In this chapter, we studied group testing in the presence of correlations, where the correlation is induced by a community structure. We showed that exploiting the knowledge provided by the community structure allows us to use upto 60% fewer tests compared to traditional test designs. We also showed how to incorporate this information into a loopy belief propagation decoder to see performance benefits. An open question is to understand which community structures, beyond the examples considered in this chapter, offer benefits and how large these benefits can be. Another open question is to design algorithms for more general testing models (see [GPR20], for example) and/or under limited information about the community structure (for example, due to technological limitations, privacy issues and fast-changing structures).

5.8 Appendix

5.8.1 Proof of Theorem 5.1

Proof. Ineq. (5.1) is because of the following counting argument: There are only 2^T combinations of test results. But, because of the community model I, there are $\binom{F}{k_f} \cdot \prod_{C \in \mathcal{G}} \prod_{d \in D_C} \binom{|V_d|}{|\mathcal{K}(V_d)|}$ possible sets of infected members that each must give a different set of results. Thus,

$$2^T \geq \binom{F}{k_f} \cdot \prod_{C \in \mathcal{G}} \prod_{d \in D_C} \binom{|V_d|}{|\mathcal{K}(V_d)|},$$

which reveals the result. The RHS of the latter inequality is because there are $\binom{F}{k_f}$ possible combinations of infected communities, each of which has the same chance of occurring and is associated with a structure of infected components. For each infected component, we consider the standard partition D_C (as in Figure 5.1). In each disjoint set d of the partition, there are $\binom{|V_d|}{|\mathcal{K}(V_d)|}$ possible combinations of infected members, each of which having the same chance of occurring–hence the double product of the RHS. Since the binomial coefficient is equal to 1, if the number of infected members in a disjoint set d is 0, instead of focusing only on the infected components and infected disjoint sets, we can let the products at the RHS be over all disjoint sets of all components in the graph. $\square$

5.8.2 Proof of Lemma 5.1

For a non-overlapped non-infected member i that belongs to only one community, the probability that i is misidentified as infected is $1-(1-pq)^{c-1}$. For an overlapped non-infected member i that belongs to more than one communities, the probability that i is misidentified as infected is $1-(1-pq)^{2(c-1)}$. Note that we assume the decoding of G_1 has no errors, i.e., it identifies all non-infected outer sets correctly. Then for the pairwise overlap structure in the example, the infection status of all non-overlapped communities and non-overlapped parts are identified correctly. The COMP decoding of G_2 has no FNs. The expected total number of FPs N_0 can be obtained as $N_0 \leq (1-(1-pq)^{c-1}) \cdot N_1 + \left(1-(1-pq)^{2(c-1)}\right) \cdot N_2$, where the inequality is because the RHS have not used the testing resluts of G_1, N_1 and N_2 are the expected number of non-overlapped and overlapped members in infected communities, respectively. We can calculate N_1 as follows,

$$N_1 = (F-2F_o)q(1-p)M + 2F_oq(1-p)(M-M_o), \tag{5.4}$$

where $(F-2F_o)q$ is the expected number of infected non-overlapped communities, $(1-p)M$ is the expected number of non-infected members in each infected non-overlapped community, $2F_oq$ is the expected number of infected overlapped communities, and $(1-p)(M-M_o)$ is the expected number of non-infected members in each infected overlapped community. Similarly, N_2 can be calculated as

$$N_2 = F_o\left(1-(1-q)^2\right)(1-p)M_o, \tag{5.5}$$

where $F_o(1-(1-q)^2)$ is the expected number of overlaps, and $(1-p)M_o$ is the expected number of non-infected members in each overlapped part.

5.8.3 LBP: message passing rules

We here describe our loopy belief propagation algorithm (LBP) and update rules for our probabilistic model (II). We use the factor graph framework of [KFL01] and derive closed-form expressions for the sum-product update rules (see equations (5) and (6) in [KFL01]).

The LBP algorithm on a factor graph iteratively exchanges messages across the variable and factor nodes. The messages to and from a variable node V_j or U_i are *beliefs* about the variable or distributions (a local estimate of $\Pr(V_j|\text{observations})$ or $\Pr(U_i|\text{observations})$). Since all the random variables are binary, in our case each message would be a 2-dimensional vector $[a, b]$ where $a, b \geq 0$. Suppose the result of each test is y_t, i.e., $Y_t = y_t$ and we wish to compute the marginals $\Pr(X_e = x|Y_1 = y_1, Y_2 = y_2, ..., Y_T = y_T)$ and $\Pr(U_v = u|Y_1 = y_1, Y_2 = y_2, ..., Y_T = y_T)$ for $x, u \in \{0, 1\}$. The LBP algorithm proceeds as follows:

1. *Initialization:* The variable nodes V_j and U_i transmit the message $[0.5, 0.5]$ on each of their incident edges. Each variable node Y_τ transmits the message $[1 - y_\tau, y_\tau]$, where y_τ is the observed test result, on its incident edge.

2. *Factor node messages:* Each factor node receives the messages from the neighboring variable nodes and computes a new set of messages to send on each incident edge. The rules on how to compute these messages are described next.

3. *Iteration and completion.* The algorithm alternates between steps 2 and 3 above a fixed number of times (in practice 10 or 20 times works well) and computes an estimate of the posterior marginals as follows – for each variable node V_j and U_i, we take the coordinatewise product of the incoming factor messages and normalize to obtain an estimate of $\Pr(V_j = x|y_1...y_T)$ and $\Pr(U_i = u|y_1...y_T)$ for $x, u \in \{0, 1\}$.

Next we describe the simplified variable and factor node message update rules. We use equations (5) and (6) of [KFL01] to compute the messages.

Leaf node messages: At every iteration, the variable node y_τ continually transmits the message $[0,1]$ if $y_\tau = 1$ and $[1,0]$ if $y_\tau = 0$ on its incident edge. The factor node $\Pr(V_j)$ continually transmits $[1-q, q]$ on its incident edge; see Fig. 5.6 (a) and (b).

Variable node messages: The other variable nodes Vj and U_i use the following rule to transmit messages along the incident edges: for incident each edge e, a variable node takes the elementwise product of the messages from every other incident edge e' and transmits this along e; see Fig. 5.6 (c).

Factor node messages: For the factor node messages, we calculate closed form expressions for the sum-product update rule (equation (6) in [KFL01]). The simplified expressions are summarized in Fig. 5.6 (d) and (e). Next we briefly describe these calculations.

Firstly, we note that each message represents a probability distribution. One could, without loss of generality, normalize each message before transmission. Therefore, we assume that each message $\mu = [a,b]$ is such that $a + b = 1$. Now, the the leaf nodes labeled $\Pr(V_j)$ perennially transmit the prior distribution corresponding to V_j.

Next, consider the factor node $\Pr(U_i|X_{S_i})$ as shown in Fig. 5.6 (e). The message sent to U_i is calculated as

$$\begin{aligned}\nu_0 &= \sum_{\{x_e \in \{0,1\}: e \in S_i\}} \Pr(U_i = 0 | X_{S_i} = x_{S_i}) \prod_{e \in S_i} s^{(e)}_{x_e} \\ &= \sum_{\{x_e \in \{0,1\}: e \in S_i\}} \prod_{e \in S_i} (s^{(e)}_{x_e} (1-p_e)^{x_e}) \\ &= \prod_{e \in S_i} (s^{(e)}_0 + s^{(e)}_1 (1-p_e)).\end{aligned}$$

Similarly, ν_1 can be computed to be $\nu_1 = 1 - \nu_0$. Now, the message sent to each X_e is

$$\mu_{x_e} = \sum_{\substack{u \in 0,1, \\ \{x_{e'} \in \{0,1\}: e' \in S_i \setminus \{e\}\}}} \Pr(U_i = u | X_{S_i} = x_{S_i}) w_u \prod_{e' \in S_i \setminus \{e\}} s^{(e')}_{x_{e'}}$$

$$= \sum_{\{x_{e'} \in \{0,1\}: e' \in S_i \setminus \{e\}\}} \left(\prod_{e' \in S_i \setminus \{e\}} s_{x_{e'}}^{(e')} \right) \left(w_0 \prod_{e' \in S_i} (1-p_{e'})^{x_{e'}} + w_1 (1 - \prod_{e' \in S_i} (1-p_{e'})^{x_{e'}}) \right)$$

$$= w_0 (1-p_e)^{x_e} \prod_{e' \neq e} (s_0^{(e')} + s_1^{(e')}(1-p_{e'}))$$

$$+ w_1 \left[1 - (1-p_e)^{x_e} \prod_{e' \neq e} (s_0^{(e')} + s_1^{(e')}(1-p_e')) \right].$$

Finally for the factor nodes $\Pr(Y_\tau | U_{\delta_\tau})$ as shown in Fig. 5.6 (d), note that the messages to Y_τ play no role since they are never used to recompute the variable messages. The messages to U_i nodes are expressed as

$$\mu_u = \sum_{\substack{y \in \{0,1\}, \\ \{u_{i'} \in \{0,1\}: i' \in \delta_\tau \setminus \{i\}\}}} \left(\Pr(Y_\tau = y | U_{\delta_\tau} = u_{\delta_\tau})(1-y_\tau)^{1-y} y_\tau^y \prod_{i' \in \delta_\tau \setminus \{i\}\}} s_{u_{i'}}^{(i')} \right)$$

$$= (1-y_\tau) \sum_{\substack{\{u_{i'} \in \{0,1\}: \\ i' \in \delta_\tau \setminus \{i\}\}}} \left(\Pr(Y_\tau = 0 | U_{\delta_\tau} = u_{\delta_\tau}) \prod_{i' \in \delta_\tau \setminus \{i\}\}} s_{u_{i'}}^{(i')} \right)$$

$$+ y_\tau \sum_{\substack{\{u_{i'} \in \{0,1\}: \\ i' \in \delta_\tau \setminus \{i\}\}}} \left(\Pr(Y_\tau = 1 | U_{\delta_\tau} = u_{\delta_\tau}) \prod_{i' \in \delta_\tau \setminus \{i\}\}} s_{u_{i'}}^{(i')} \right).$$

From our Z-channel model, recall that $\Pr(Y_\tau = 0 | U_{\delta_\tau} = u_{\delta_\tau}) = 1$ if $u_i = 0 \ \forall \ i \in \delta_\tau$ and $\Pr(Y_\tau = 0 | U_{\delta_\tau} = u_{\delta_\tau}) = z$ otherwise. Thus we split the summation terms into 2 cases – one where $u_{i'} = 0$ for all i' and the other its complement. Also combining this with the assumption that the messages are normalized, i.e., $s_0^{(i)} + s_1^{(i)} = 1$, we get

$$\sum_{\substack{\{u_{i'} \in \{0,1\}: \\ i' \in \delta_\tau \setminus \{i\}\}}} \left(\Pr(Y_\tau = 0 | U_{\delta_\tau} = u_{\delta_\tau}) \prod_{i' \in \delta_\tau \setminus \{i\}\}} s_{u_{i'}}^{(i')} \right)$$

$$= \mathbb{1}_{u=1} z + \mathbb{1}_{u=0} \left\{ 1 - (1-z)(1 - \prod_{\substack{i' \in \delta_\tau \\ i' \neq i}} s_0^{(i')}) \right\},$$

and

$$\sum_{\substack{\{u_{i'} \in \{0,1\}: \\ i' \in \delta_\tau \setminus \{i\}\}}} \left(\Pr(Y_\tau = 1 | U_{\delta_\tau} = u_{\delta_\tau}) \prod_{i' \in \delta_\tau \setminus \{i\}\}} s_{u_{i'}}^{(i')} \right)$$

$$= \mathbb{1}_{u=1}(1-z) + \mathbb{1}_{u=0}\left((1-z)(1-\prod_{\substack{i'\in\delta_\tau\\ i'\neq i}} s_0^{(i')})\right).$$

Substituting $u=0$, and $u=1$ we obtain the messages

$$\mu_0 = (1-y_\tau)\left\{1-(1-z)(1-\prod_{\substack{i'\in\delta_\tau\\ i'\neq i}} s_0^{(i')})\right\} + y_\tau(1-z)(1-\prod_{\substack{i'\in\delta_\tau\\ i'\neq i}} s_0^{(i')}),$$

and

$$\mu_1 = (1-y_\tau)z + y_\tau(1-z).$$

For our probabilistic model, the complexity of computing the factor node messages increases only linearly with the factor node degree.

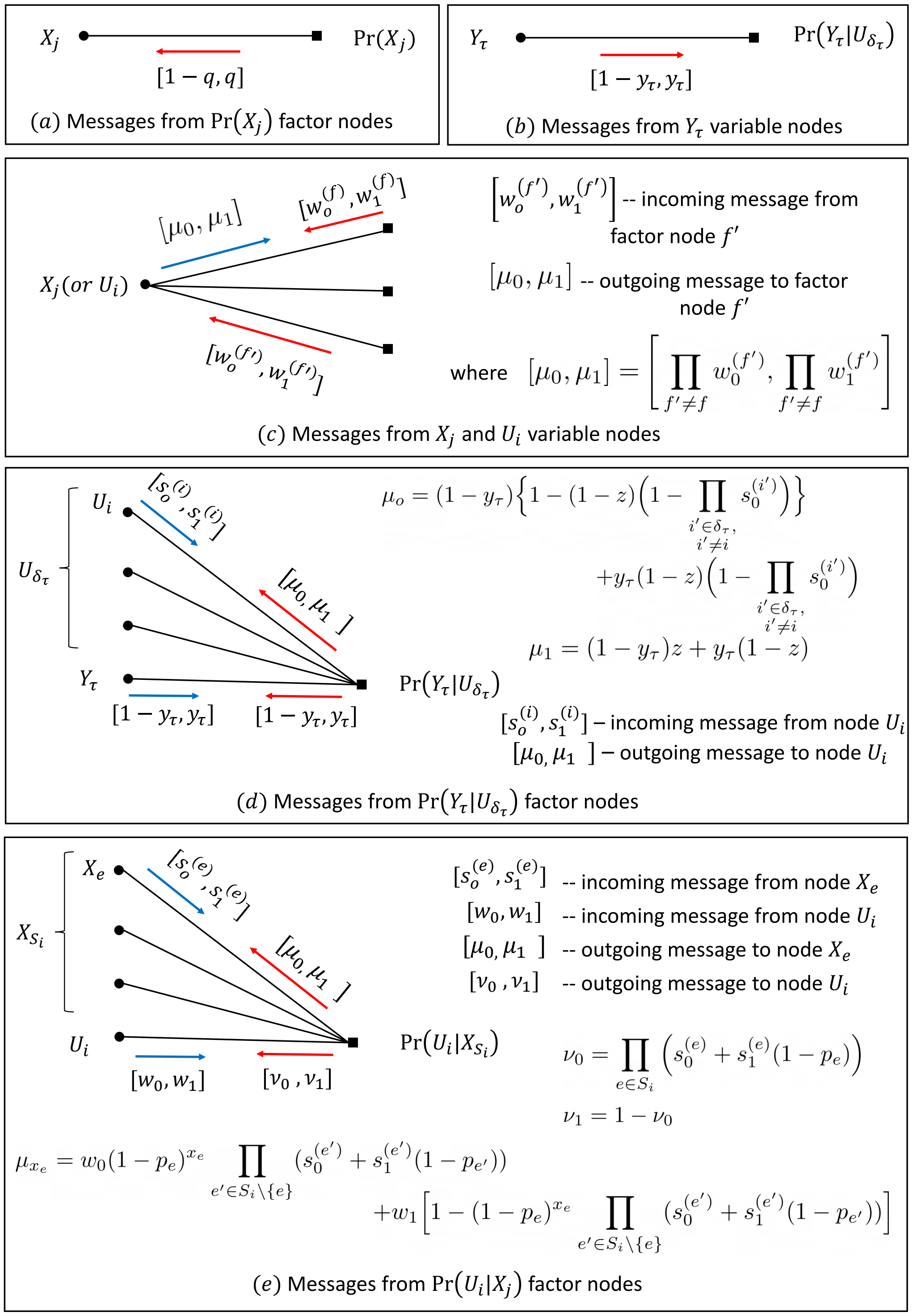

X_j
$\Pr(X_j)$
$[1-q,q]$
(a) Messages from $\Pr(X_j)$ factor nodes
Y_τ
$\Pr(Y_\tau|U_{\delta_\tau})$
$[1-y_\tau, y_\tau]$
(b) Messages from Y_τ variable nodes
$X_j(or\ U_i)$
$[\mu_0,\mu_1]$
$[w_o^{(f)}, w_1^{(f)}]$
$[w_o^{(f')}, w_1^{(f')}]$
$[w_o^{(f')}, w_1^{(f')}]$ -- incoming message from factor node f'
$[\mu_0,\mu_1]$ -- outgoing message to factor node f'
where $[\mu_0,\mu_1] = \left[\prod_{f'\neq f} w_0^{(f')}, \prod_{f'\neq f} w_1^{(f')}\right]$
(c) Messages from X_j and U_i variable nodes
U_{δ_τ}
U_i
$[s_o^{(i)}, s_1^{(i)}]$
$[\mu_0,\mu_1]$
Y_τ
$[1-y_\tau, y_\tau]$
$[1-y_\tau, y_\tau]$
$\Pr(Y_\tau|U_{\delta_\tau})$
$\mu_o = (1-y_\tau)\Big\{1-(1-z)\Big(1-\prod_{\substack{i'\in\delta_\tau,\\ i'\neq i}} s_0^{(i')}\Big)\Big\} + y_\tau(1-z)\Big(1-\prod_{\substack{i'\in\delta_\tau,\\ i'\neq i}} s_0^{(i')}\Big)$
$\mu_1 = (1-y_\tau)z + y_\tau(1-z)$
$[s_o^{(i)}, s_1^{(i)}]$ – incoming message from node U_i
$[\mu_0,\mu_1]$ – outgoing message to node U_i
(d) Messages from $\Pr(Y_\tau|U_{\delta_\tau})$ factor nodes
X_{S_i}
X_e
$[s_o^{(e)}, s_1^{(e)}]$
$[\mu_0,\mu_1]$
U_i
$[w_0, w_1]$
$[\nu_0, \nu_1]$
$\Pr(U_i|X_{S_i})$
$[s_o^{(e)}, s_1^{(e)}]$ -- incoming message from node X_e
$[w_0, w_1]$ -- incoming message from node U_i
$[\mu_0,\mu_1]$ -- outgoing message to node X_e
$[\nu_0, \nu_1]$ -- outgoing message to node U_i
$\nu_0 = \prod_{e\in S_i}\Big(s_0^{(e)} + s_1^{(e)}(1-p_e)\Big)$
$\nu_1 = 1-\nu_0$
$\mu_{x_e} = w_0(1-p_e)^{x_e}\prod_{e'\in S_i\setminus\{e\}}(s_0^{(e')} + s_1^{(e')}(1-p_{e'})) + w_1\Big[1-(1-p_e)^{x_e}\prod_{e'\in S_i\setminus\{e\}}(s_0^{(e')} + s_1^{(e')}(1-p_{e'}))\Big]$
(e) Messages from $\Pr(U_i|X_j)$ factor nodes

Figure 5.6: The update rules for the factor and variable node messages.

CHAPTER 6

Dynamic infection model

Summary: *In this chapter, we consider the dynamics of disease spread and study how static group testing can be used in such a setting. We first prove our new lower bound for the static case and then show the order-optimality of existing group testing algorithms. We then make precise the conditions under which the dynamic testing problem reduces to the static case, enabling the use of static group testing algorithms.*

So far we dealt with static infection models ignoring the fact that infections continually proliferate in a community. For the dynamic case, recent works have identified the significance of proactive testing and individual-level intervention for the control of the disease spread (e.g. [TKL21, TRL20, BBL20]), but to the best of our knowledge none of them addresses the challenges above efficiently. Most solutions rely on the idea of "testing everyone individually", which is inefficient for two reasons: on one hand, using cheap rapid testing usually results in many people (false positives) ending up in isolation without reason and at non-negligible societal cost; on the other hand, using accurate tests like PCR can be forbiddingly expensive. As a result, these works need to either neglect the cost of the former or alleviate the cost of the latter by scheduling tests on a (bi)weekly or monthly basis.

Therefore, a critical question is still open: can we use accurate/expensive tests more efficiently? In other words, can we identify all new infections that happen each day (complete testing performance), using significantly fewer accurate/expensive tests than complete testing? Complete accurate testing (e.g. PCR) on a daily basis and isolation of infected individuals can significantly reduce the number of infected people, as the example in Fig. 6.1

illustrates. Note that even with complete testing new infections still occur due to the delay between testing and receiving the test results (Fig. 6.1 assumes the usual delay of one day). Still, this is the best performance we can hope for, both in terms of containing infection and alleviating the societal impact of "false" quarantines; we thus ask how many tests do we really need to replicate it.

Traditional group testing strategies offer a powerful toolset for minimizing the number of tests, but they do not account for the time dynamics of a disease spread and do not take into account community structure. When the number of available tests is limited, two strategies are usually applied: sample testing, which tests only a sample of selected individuals, and/or group testing, which pools together diagnostic samples to reduce the number of tests needed to identify infected individuals in a population (e.g., see [AJS19] and references therein). Both examine a static scenario: the state of individuals is fixed (infected or not), and the goal is to identify all infected ones.

To the best of our knowledge, our work in [SNF21b] was the first paper that targeted community-aware, group-test design for the dynamic case. In that work we used the well-established continuous-time SIR stochastic network model in [KMS17], where individuals are regarded as the vertices of a graph $\mathcal{G}$ and an edge denotes a contact between neighboring vertices, and explored group testing strategies that were able to track the epidemic state evolution at an individual level, using a small number of tests. However, due to the complexity of the continuous time model, we were not able to provide theoretical guarantees for the minimum number of tests, and although we did consider testing delays.

In this chapter, we allow interventions, we use discrete-time SIR models for disease spread and we derive theoretical guarantees. Discrete time models fit more naturally with testing and intervention (which happen at discrete time-intervals), and are more amenable to analysis enabling methods to derive guarantees on the number of tests needed to achieve close-to-complete-testing accuracy. In this chapter, we use a model called the "discrete-time SIR stochastic block model," which can be considered as a discrete version of the continuous-time SIR stochastic network model over a specific type of weighted graph. The graph used captures knowledge of an underlying community structure, as discussed in Section 6.2. In Appendix 6.6.1, we compare the continuous-time model from [KMS17] with the discrete-time model introduced in our work and justify the use of our discrete-time model. We also note that our results are applicable to a larger set of SIR models, as discussed in Section 6.3.3.

Our main conclusion is that we can leverage the knowledge of the community and the dynamic model to inform group testing algorithms that are order-optimal and use a much smaller number of tests than complete testing to achieve the same performance. We arrive at this conclusion building on the following contributions.

We first argue that for discrete-time SIR models, given test results that identify the infection state the previous day, the problem of identifying the new infections each day reduces to static-case group testing with independent (but not identical) priors. So, existing non-adaptive algorithms such as CCA and/or random testing [LCH14] can be reused. Figure 6.2 illustrates the sequence of events taking place on each day t.

We then derive a new lower bound (Theorem 6.2) for the number of tests needed in the case of independent (but not identical) priors. The main benefit of the new bound is not on "improving" upon the well-known entropy lower bound (stated as Lemma 6.1), but having a form that allows to prove order-optimality of group testing algorithms. In particular, we can prove that under mild assumptions existing nonadaptive algorithms are order-optimal in the static case (Corollary 6.2). This, in our opinion, is an interesting result on its own, since non-identical priors static group testing remains a relatively unexplored field compared to i.i.d. probabilistic group testing.

Finally, we derive conditions on the discrete-time SIR stochastic block model parameters under which order-optimal group test designs for the static case are also optimal for the dynamic case (Theorem 6.3). Simulation results show that indeed under these conditions we can achieve the performance of complete individual testing using a much smaller (close to the entropy lower bound) number of tests; for example, over a period of 50 days, group

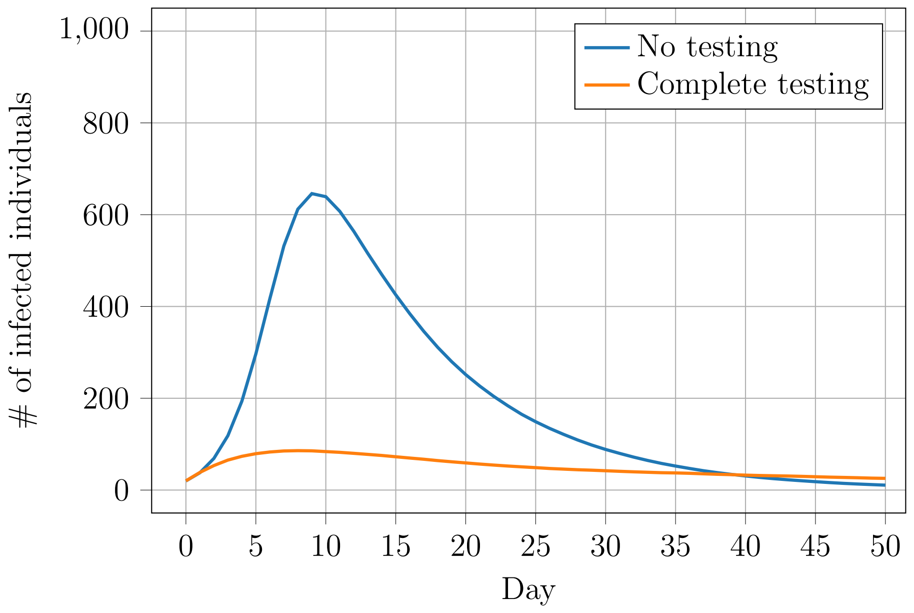


Figure 6.1: Discrete-time SIR stochastic block model simulated on a population of 1000 individuals. Notice that without any testing or intervention a large fraction of the population gets infected. With *complete testing* (individually testing everyone everyday) and intervention (isolating individuals who are identified as infected) we can flatten the curve to a large extent. We assume that test results are only available the next day; if the test results were instantaneous we can identify all infections on the first day and isolate them and there would be no subsequent new infections.

testing needs an average of around 100 tests per day for a population of 1000 individuals. Our simulations use existing non-adaptive test designs - we do not derive new code designs as the existing ones are sufficient. However, we do use marginal probabilities derived daily from the SIR model to inform the group design: that is, the group tests we use vary from day to day, and their design leverages the knowledge of the underlying system dynamics that depend on the community structure, as well as the previous day test results.

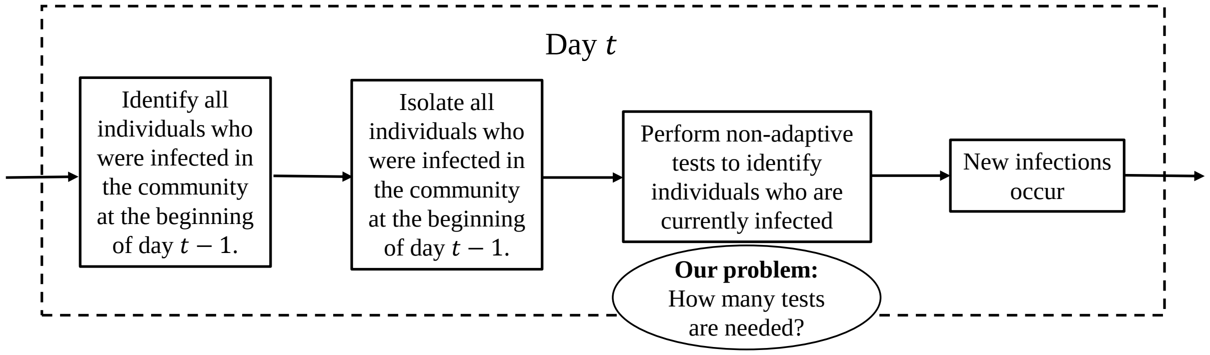

Day t
Identify all individuals who were infected in the community at the beginning of day $t-1$.
Isolate all individuals who were infected in the community at the beginning of day $t-1$.
Perform non-adaptive tests to identify individuals who are currently infected
New infections occur
Our problem: How many tests are needed?

Figure 6.2: The dynamic testing problem with daily interventions. How many tests are needed to achieve complete testing performance everyday, given that test results become available after a day's delay.

6.1 Related Work

This chapter shares similar goals with our prior work in [SNF21b], where we considered the well established continuous-time SIR stochastic network model (see [KMS17]) and focused on how many tests to use and whom to test in order to track the infected individuals in the population. That work also explored how well one can learn the infected individuals given delayed test results, but gave no theoretical guarantees on the methods and did not consider intervention. Our discrete-time model for disease spread in this chapter, however, is more amenable to analysis and illustrates better the usefulness of group testing, being at the same time useful for practical reasons (more about this in Section 6.2). We further note that our results are applicable to a more general set of SIR models as discussed in Section 6.3.3, remark 2.

Our model is closely related to the independent cascade model (see for example [KKT03] and references therein), studied in the context of influence maximization in social networks, where we can interpret influence/rumor propagation as infections in our context. A crucial difference of our model from this is that our model allows multiple opportunities of infections over time whereas the independent cascade model only allows one opportunity to "infect". Therefore, as is noted, in our model the infectious individuals remain infectious until recovered or isolated.

The work in [GCW20] considers a discrete stochastic model for the progression of COVID-19 based on contact networks and leverages the model dynamics to inform a group test decoder; however their scope is different, as they test infrequently and thus infections are highly correlated, do not consider interventions, do not look for optimal group test designs, and do not provide theoretical guarantees on the number of tests needed.

Since we use the main principles of the SIR model our work is closely related to epidemic modeling. Works in epidemiology discuss the implications of testing and intervention for COVID-19 employing stochastic network models (see [BBL20,TRL20] and references therein) but do not consider test designs that exploit the knowledge of the underlying dynamical system. Works in control theory (see [MSO20] and references therein) consider deterministic SIR compartment models (at the population level) and focus on intervention schemes. Here we are interested in both testing and intervention and use an individual-level SIR model.

Further related to static group testing is the work on graph-constrained group testing (see for example [CKM12], [KZ12]), which solves the problem of how to design group tests when there are constraints on which samples can be pooled together, provided in the form of a graph. In our case, no such constraints exist and individuals can be pooled together into tests freely.

6.2 Preliminaries and problem formulation

In this section we formalize our setup. Since our work for the dynamic case builds upon existing ones from static group testing, we first review some major results in that area that we also reuse in this chapter (Section 6.2.1). We then provide our model (Section 6.2.3) and problem formulation (Section 6.2.3).

6.2.1 Preliminary: review of results from static group testing

Traditional group testing typically assumes a population of N individuals out of which some are infected. Three infection models are typically considered: (i) in the *combinatorial priors model*, a fixed number of infected individuals k, are selected uniformly at random among all sets of size k; (ii) in *i.i.d probabilistic priors model*, each individual is i.i.d infected with probability p; (iii) in the *non-identical probabilistic priors model*, each item i is infected independently of all others with prior probability p_i, so that the expected number of infected members is $\bar{k} = \sum_{i=1}^{N} p_i$ [LCH14]. In this paper we mostly use results that apply to the last case.

A group test τ takes as input samples from n_τ individuals, pools them together and outputs a single value: positive if any one of the samples is infected, and negative if none is infected. More precisely, let $U_i = 1$ when individual i is infected and 0 otherwise. Then the group testing output y_τ takes a binary value calculated as $y_\tau = \bigvee_{i \in \mathcal{D}_\tau} U_i$[1], where $\bigvee$ stands for the `OR` operator (disjunction) and $\mathcal{D}_\tau$ is the group of people participating in the test.

• For the probabilistic model (ii), any non-adaptive algorithm with a success probability bounded away from zero as $N \to \infty$ must have $T = \Omega\left(\min\{\bar{k}\log N, N\}\right)$ [BPS20, Theorem 1], [CGH20b]. This means that either any non-adaptive group testing with a number of tests $O(\bar{k}\log N)$ is order optimal, or individual testing is order optimal[2]. In particular, random test designs, such as i.i.d. Bernoulli [ABJ14, AS12, SC16] and near-constant tests-per-item [CGH20a, JAS19] have been proved to be order-optimal in a sparse regime where $\bar{k} = \Theta(N^\alpha)$ and $\alpha \in (0,1)$. In fact, in the same regime, [CGH20b] has provided the precise constants for optimal non-adaptive group testing. Conversely, classic individual testing has been proved to be optimal in the linear ($\bar{k} = \Theta(N)$) [Ald19] and the mildy sublinear regime

[1]We assume that the tests are noiseless here, for simplicity. The group testing literature also extensively studies the case when the testing output is noisy.

[2]The achievability and converse results provided here are usually proved for combinatorial model (i) (a summary can be found in [PS20]), but they are directly applicable to model (ii) by considering $p = k/N$ (see Theorem 1.7 and Theorem 1.8 in [AJS19] or [BPS20]).

($\bar{k} = \omega(\frac{N}{\log N})$) [BPS20].

• For the probabilistic model (iii), a lower bound for the number of tests needed is given by the entropy, stated below:

Lemma 6.1 (Entropy lower bound)**.** *Consider the non-identical probabilistic priors model of static group testing, where each individual $i \in [N]$ is infected independently with probability p_i. The number of tests T needed by a non-adaptive algorithm to identify the infection status of all individuals with a vanishing probability of error satisfies*

$$T \geq \sum_{i=1}^{N} h_2\left(p_i\right),$$

where $h_2(\cdot)$ is the binary entropy function.

See Appendix A in [LCH14] for a proof. On the algorithmic side, two known algorithms are: the adaptive laminar algorithms that need at most $2\sum_{i=1}^{N} h_2\left(p_i\right) + 2\bar{k}$ tests on average, and the "Coupon collector" nonadaptive algorithm (CCA) that needs at most $T \leq 4e(1+\delta)\bar{k}\ln N$ test to achieve an error probability no larger than $2N^{-\delta}$ whenever $p_i \leq 1/2$ [LCH14, CJS14].

6.2.2 Discrete-time SIR stochastic block model

We now describe our infection model via the discrete-time SIR stochastic block model with parameters $(N, C, q_1, q_2, p_{\text{init}})$. Consider a population of size N that is partitioned into multiple communities of size C. For simplicity we assume that N/C is an integer. On any day $t \in \mathbb{N}$, each individual can be in one of three states: Susceptible ($\mathcal{S}$), Infected ($\mathcal{I}$) or Recovered ($\mathcal{R}$). Let $X_i^{(t)} \in \{\mathcal{S}, \mathcal{I}, \mathcal{R}\}$ denote the state of individual i on day t, and define the state of the system as $\mathbf{X}^{(t)} \triangleq (X_1^{(t)}, X_2^{(t)}, ..., X_N^{(t)})$. A small number of individuals are initially infected, and all new infections occur during "transmissible contacts" between infected and susceptible individuals. Recoveries occur independent of infections.

More precisely, on day $t = 0$, every individual is i.i.d infected with probability p_{init}. The following steps repeat everyday starting at $t = 1$:

- An infected individual in some community infects a susceptible one from the same community w.p. q_1, independently of the other infected individuals of the community.
- An infected individual in some community infects a susceptible one from another community w.p. q_2, independently from all other infected individuals.
- An infected individual recovers independently from all other individuals w.p. r.

The discrete-time SIR stochastic block model can be envisioned as a discrete version of the well-established continuous-time SIR stochastic network model [KMS17] on the corresponding weighted graph. It inherits the main properties from the latter; for example, infections are transmitted only from an infected to a susceptible individual and both infections and recoveries are stochastic. The main difference is that in the continuous-time one, the infections and the recoveries happen according to continuous-time Markovian process with transmissibility rate β and recovery rate γ, which means that the time until a new state transition ($\mathcal{S} \to \mathcal{I}$ or $\mathcal{I} \to \mathcal{R}$) is exponentially distributed (with mean β or γ respectively). Indeed this makes the event that an individual got infected and subsequently recovered within a single day possible in the continuous-time model, whereas this is impossible in our discrete-time model.

Learning the intra-community and inter-community structure to model infection transmissions is, we believe, also practically feasible. Close contact "community" data is often readily available; for example students in each classroom in a school could form a community, and so could workers in the same office space. We also note that community-level network models alleviate some of the privacy concerns associated with using contact tracing data which tracks the exact pairs of individuals who come in contact with each other on a daily basis.

A useful remark about our model is that the state of an individual $X_i^{(t)} \in \{\mathcal{S}, \mathcal{I}, \mathcal{R}\}$ is different from the infection state $U_i^{(t)} \in \{0, 1\}$, where 1 (resp. 0) corresponds to the "infected" (resp. "not infected"). Indeed $U_i^{(t)} = 1$ iff $X_i^{(t)} = \mathcal{I}$. This difference is important in our context, because our tests do not distinguish between susceptible and recovered individuals. In the remainder of the paper, $X_i^{(t)}$ will be called the "SIR state" of individual i, while $U_i^{(t)}$ will be i's "infection status." As a result, whether a individual is infected or not changes with the day, and thus we now consider a random variable $U_i^{(t)}$ associated with each individual that describes whether it is infected on day t ($t \in N$).

6.2.3 The dynamic testing problem formulation

As can be seen from Fig. 6.1, testing everyone, everyday, and isolating infected individuals helps drastically reduce the number of infections that happen on a given day. We assume that the results of a test administered on a particular day are available only the next day (as usually is the case with classic PCR testing for SARS-COV-2). We also isolate only the individuals who test positive, and we do so as soon as the test results are available. Moreover, we bring back the isolated individual into the population only when they are completely recovered. Note that in the SIR model, recovered individuals cannot get infected and play no role in transmitting the infection. Therefore, without loss of generality, we could assume that isolated individuals remain isolated for the rest of the testing period.

Given these assumptions, the question we ask is if complete testing is necessary to identify all new infections everyday, or if we can achieve the same performance as complete testing with significantly fewer number of tests. In particular, how many non-adaptive group tests are necessary and sufficient to identify all new infections (with a vanishing error probability) on each day? Our problem formulation is depicted in Fig. 6.2.

To aid a precise mathematical formulation for the problem, we first introduce some notation.

- $I_j^{(t)}$: number of new infections in community j that occurred on day t. Note that in

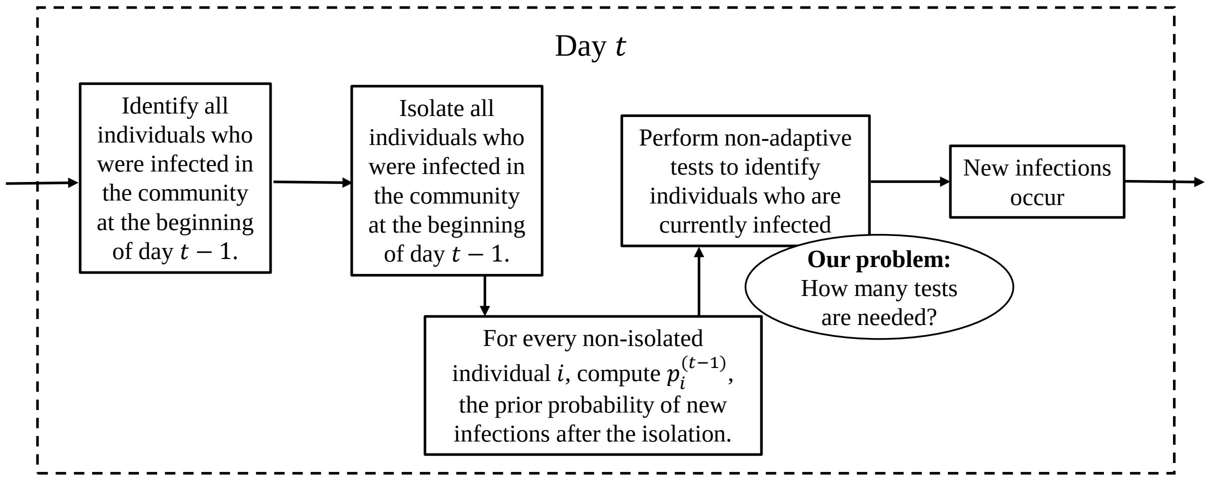

Day t
Identify all individuals who were infected in the community at the beginning of day $t-1$.
Isolate all individuals who were infected in the community at the beginning of day $t-1$.
Perform non-adaptive tests to identify individuals who are currently infected
New infections occur
Our problem: How many tests are needed?
For every non-isolated individual i, compute $p_i^{(t-1)}$, the prior probability of new infections after the isolation.

Figure 6.3: From dynamic to static testing: on day t we perfectly learn the states of all non-isolated individuals at the time of testing on the previous day $t-1$. Given this information, we know that each susceptible individual in community j is later infected with probability $p_j^{(t-1)}$ independent of every other individual. How many tests are needed to attain a vanishing probability of error on this non-identical static group testing problem?

the set-up of Fig. 6.2, this number is also equal to the number of infected individuals remaining in community j after intervention has been decided for day $t+1$. The new infections which happened on day t will only be identified by the tests administered on day $t+1$, whose results are available only on day $t+2$.

- $p_j^{(t)}$: the probability of an individual in community j who was susceptible at the end of day $t-1$ getting infected on day t. Note that this is same for every such individual in community j, by symmetry of the model. Moreover, we can calculate this probability as

$$\begin{aligned} p_j^{(t)} &= 1 - \Pr(\text{individual is not infected on day } t) \\ &= 1 - (1-q_1)^{I_j^{(t-1)}} (1-q_2)^{\sum_{j' \neq j} I_{j'}^{(t-1)}}. \end{aligned}$$

Reduction to static group testing with non-identical probabilistic priors. Note that given $I_j^{(t-1)}$ $\forall j$, an individual belonging to community j is infected *independently* of every other individual with probability $p_j^{(t)}$ on day t. Thus, conditioned on the infection status of all individuals on day $t-1$, the infections which happen on day t are independent (but not identically distributed). Now in our dynamic testing problem set-up, on day t we perfectly learn the infection statuses of all non-isolated individuals at the time of testing on the previous day $t-1$. Given this information, we can exactly calculate the $p_j^{(t-1)}$ $\forall j$ (see Fig. 6.3), i.e. the probability that each susceptible individual in community j was later infected because of the non-isolated infected individuals.

So, given accurate test results, the dynamic testing problem is transformed daily to the problem of static group testing with non identical probabilistic priors (model (iii) in Section 6.2.1). Therefore, the precise question we are after is the following: given that each individual in community j is infected with probability $p_j^{(t)}$ independently of every other individual, how many tests are necessary and sufficient to learn the infection status with a vanishing probability of error? We answer this question in the next section.

6.3 Main results

In this section, we prove our main theoretical results. For brevity, we will use the terms "i.i.d. priors" and "non-identical priors" to refer to i.i.d probabilistic priors model (ii) and non identical probabilistic priors model (iii) from Section 6.2.1, respectively. The contents of this section are ordered as follows:

- First, we provide a new lower bound on the number of tests required for the problem of static group testing with non i.i.d probabilistic priors (Theorem 6.2). To prove Theorem 6.2, we use two intermediate results: (a) we show that any test design that "works" for a given prior probabilities of infection $(p_1, p_2, ..., p_N)$ also works for the *reduced* prior probabilities $(p'_1, p'_2, ..., p'_N)$ where $p'_i \leq p_i \leq 0.5$ $\forall i$. In words, we essentially prove that group testing is easier when the infections are sparser (Theorem 6.1); and (b) we show the following interesting property of the optimal decoder (Lemma 6.3) –

if the optimal decoder correctly infers all the infection statuses when a set $\mathcal{D}$ is the set of infected individuals, then it will also correctly infer all the infection statuses when $\mathcal{D}' \subset \mathcal{D}$ is the set of infected individuals.

- Second, we use simple asymptotic arguments to show that some existing group testing strategies (such as CCA [LCH14] for non-identical priors and random testing for i.i.d priors) are order-optimal for non-identical priors (Corollary 6.2), when $p_{\max} = O(p_{\min})$, where $p_{\max}$ is the maximum entry in $(p_1, p_2, ..., p_N)$ and $p_{\min}$ is the minimum entry. The order $O(\cdot)$ is order with respect to the size of the population N.

- Finally, in Theorem 6.3, we bridge the gap between our dynamic testing problem formulation and the above static testing problem by showing that if $q_1 = O(q_2)$, $p_{\text{init}} \leq 0.5$ and if $q_1 \leq \frac{1-1/\sqrt{2}}{C}$ and $q_2 \leq \frac{1-1/\sqrt{2}}{N}$, then the above two conditions on the prior vector are satisfied everyday in the discrete-time SIR stochastic block model parameterized by $(N, C, p_{\text{init}}, q_1, q_2)$. As a result the existing group testing strategies discussed above are order-optimal even for the dynamic testing problem formulation considered, provided that we use a sufficient number of tests each day to identify all new infections for that day.

6.3.1 Results on static group testing with non i.i.d priors

We first consider the problem of static group testing, in which a person is infected independently with a known prior probability p_i. Denote by $\mathbf{p} = (p_1, p_2, ..., p_N)$ the prior vector which collects the prior probabilities of infection of all individuals. We first define some notation specific to this subsection:

- G: test matrix

- $\mathcal{D}$: set of defectives or infections

- $\mathbf{U} = (U_1, U_2, ..., U_N)$: infection status configuration, i.e., individual i is infected if and only if $U_i = 1$.

- $\mathbf{U}(\mathcal{D}) \triangleq (U_1, ..., U_N)$ where $U_i = 1$ iff $i \in \mathcal{D}$. Basically represents the vector notation for the set of infections given by $\mathcal{D}$. Note that there is a one-one correspondence between $\mathcal{D}$ and $\mathbf{U}(\mathcal{D})$. We will use these two notations interchangeably based on convenience.

- $G(\mathbf{U})$ represents the test results corresponding to the given test design and infection status configuration.

- For a fixed number of tests T, define a decoding function $R : \{0,1\}^{[T]} \to \{0,1\}^{[N]}$ which estimates the infection statuses from the test results.

- A defective set $\mathcal{D}$ "explains" test results $\mathbf{y}$ iff $G(\mathbf{U}(\mathcal{D})) = \mathbf{y}$.

- $\Pr(\mathbf{U}; \mathbf{p})$ denotes the probability of the infection status configuration under priors $\mathbf{p}$, i.e.
$$\Pr(\mathbf{U}; \mathbf{p}) = \prod_{i=1}^{N} p_i^{U_i}(1 - p_i)^{1-U_i}.$$

- Probability of error for a test matrix, decoder pair under given priors
$$\begin{aligned}\mathbb{P}_{err}(G, R; \mathbf{p}) &\triangleq \mathop{\mathbb{E}}_{\mathbf{U} \sim \mathbf{p}} \mathbb{1}\{R(G(\mathbf{U})) \neq \mathbf{U}\} \\ &= \sum_{\mathbf{u} \in \{0,1\}^N} \Pr(\mathbf{U} = \mathbf{u}; \mathbf{p}) \mathbb{1}\{R(G(\mathbf{u})) \neq \mathbf{u}\}.\end{aligned}$$

Definition 6.1 (MAP decoder)**.** For fixed priors $\mathbf{p}$ and testing matrix G with number of tests T, we define the corresponding MAP decoder as $R_{map}(\,\cdot\,; G, \mathbf{p}) : \{0,1\}^T \to \{0,1\}^N$, where
$$R_{map}(\mathbf{y}; G, \mathbf{p}) = \mathop{\arg\max}_{\mathbf{U}: G(\mathbf{U}) = \mathbf{y}} \Pr(\mathbf{U}; \mathbf{p}).$$
In case of ties, the MAP decoder will select the solution which comes the earliest lexicographically.

In words, the MAP decoder chooses the most likely configuration which explains the test results. We next show that the MAP decoder is the optimal decoder for a fixed G and $\mathbf{p}$, i.e., the MAP decoder minimizes the probability of error amongst all decoders for any G, $\mathbf{p}$.

Remark. Though the MAP decoder is optimal, it is unclear if the optimization problem corresponding to the MAP decoder can be solved efficiently. However, many heuristics such as belief propagation (see for example [NRG21]) and random sampling methods exist which approximate well the MAP decoder. That said, in this chapter, we use the MAP decoder only as a tool for theoretical analysis of the error probability.

Lemma 6.2 (Optimality of MAP decoder)**.** *For given test matrix G and priors $\mathbf{p}$, the corresponding MAP decoder minimizes the probability of error for the test matrix under the given priors, i.e.,*

$$\mathbb{P}_{err}(G, R_{map}(\,\cdot\,; G, \mathbf{p}); \mathbf{p}) \leq \mathbb{P}_{err}(G, R; \mathbf{p}) \;\forall R.$$

We give the proof of Lemma 6.2 to Appendix 6.6.2.

Given the optimality of the MAP decoder, we will denote by

$$\mathbb{P}^*_{err}(G, \mathbf{p}) \triangleq \mathbb{P}_{err}(G, R_{map}(\,\cdot\,; G, \mathbf{p}); \mathbf{p}),$$

the optimal probability of error corresponding to a given test design and priors.

We next prove a property of the MAP decoder, and this property will be used in the proof of our main result that follows. The following Lemma says that it is easier for the MAP decoder to identify a sparser defective set.

Lemma 6.3. *Consider a test matrix G and priors $\mathbf{p}$. Suppose the corresponding MAP decoder is erroneous when identifying the defective set $\mathcal{D}$. Then the MAP decoder is also erroneous for the set of defectives is $\mathcal{D} \cup \{j\}$ with $p_j \leq 0.5$, i.e.,*

$$\begin{aligned} &\mathbb{1}\left\{R_{map}\left(G(\mathbf{U}(\mathcal{D} \cup \{j\})); G, \mathbf{p}\right) \neq \mathbf{U}(\mathcal{D} \cup \{j\})\right\} \\ &\geq \mathbb{1}\left\{R_{map}(G(\mathbf{U}(\mathcal{D})); G, \mathbf{p}) \neq \mathbf{U}(\mathcal{D})\right\}. \end{aligned}$$

For the proof of Lemma 6.3, we refer the reader to Appendix 6.6.3.

We next prove the main new result for the static case. In words, the following theorem says that the group testing problem is only easier when the infections are sparser. As a result, this allows us to lower/upper bound the group testing problem with non-identical priors by a group testing problem with identical priors.

Theorem 6.1. *Consider a testing matrix G used with two different sets of priors $\mathbf{p}$ and $\mathbf{p}'$. Further let $p_i' = p_i$ for every $i \in [N], i \neq j$ and $p_j' \leq p_j \leq 0.5$. The two prior vectors are same everywhere except at index j where $\mathbf{p}'$ is smaller. Then*

$$\mathbb{P}^*_{err}(G, \mathbf{p}') \leq \mathbb{P}^*_{err}(G, \mathbf{p}).$$

Proof. We prove this by showing that when the MAP decoder corresponding to $(G, \mathbf{p})$ pair is used as a decoder with $(G, \mathbf{p}')$, the probability of error is always lower, i.e.,

$$\begin{aligned}\mathbb{P}_{err}(G, &R_{map}(\ \cdot\ ; G, \mathbf{p}); \mathbf{p}') \\ &\leq \mathbb{P}_{err}(G, R_{map}(\ \cdot\ ; G, \mathbf{p}); \mathbf{p}) = \mathbb{P}^*_{err}(G, \mathbf{p}).\end{aligned}$$

As a result the optimal decoder for $(G, \mathbf{p}')$ pair has a probability of error not exceeding this quantity.

Now, we can express the probability of error for the MAP decoder of $(G, \mathbf{p})$ pair. For simplicity of notation in the following derivations, $\mathcal{E}(\mathcal{D}) \triangleq \mathbb{1}\left\{R_{map}(G(\mathbf{U}(\mathcal{D})); G, \mathbf{p}) \neq \mathbf{U}(\mathcal{D})\right\}$ denotes the indicator of the event that the MAP decoder is erroneous when the defective set is $\mathcal{D}$ (and under further assumptions that the priors are $\mathbf{p}$ and test matrix is G).

$$\begin{aligned}&\mathbb{P}_{err}(G, R_{map}(\ \cdot\ ; G, \mathbf{p}); \mathbf{p}) \\ &= \sum_{\mathcal{D} \in [N]} \Pr(\mathbf{U}(\mathcal{D}))\mathcal{E}(\mathcal{D}) \\ &= \sum_{\mathcal{D} \in [N]} \prod_{i \in \mathcal{D}} p_i \prod_{l \in [N] \setminus \mathcal{D}} (1 - p_l)\mathcal{E}(\mathcal{D})\end{aligned}$$

$$
\begin{aligned}
&\overset{(a)}{=} \sum_{\substack{\mathcal{D}\in[N]\\ |j\in\mathcal{D}}} \prod_{i\in\mathcal{D}} p_i \prod_{l\in[N]\backslash\mathcal{D}} (1-p_l)\mathcal{E}(\mathcal{D}) \\
&\quad + \sum_{\substack{\mathcal{D}\in[N]\\ |j\notin\mathcal{D}}} \prod_{i\in\mathcal{D}} p_i \prod_{l\in[N]\backslash\mathcal{D}} (1-p_l)\mathcal{E}(\mathcal{D}) \\
&\overset{(b)}{=} p_j \sum_{\mathcal{D}\in[N]\backslash\{j\}} \prod_{i\in\mathcal{D}} p_i \prod_{l\in[N]\backslash\mathcal{D}\cup\{j\}} (1-p_l)\mathcal{E}(\mathcal{D}\cup\{j\}) \\
&\quad + (1-p_j) \sum_{\mathcal{D}\in[N]\backslash\{j\}} \prod_{i\in\mathcal{D}} p_i \prod_{l\in[N]\backslash\mathcal{D}\cup\{j\}} (1-p_l)\mathcal{E}(\mathcal{D}), \qquad (6.1)
\end{aligned}
$$

where in (a) we split the summation into two cases – one where $j \in \mathcal{D}$ and the other where $j \notin \mathcal{D}$; in (b) we take j out of the summation.

Similarly, one could express the probability of error for the same decoder with the pair $(G, \mathbf{p}')$ as

$$
\begin{aligned}
&\mathbb{P}_{err}(G, R_{map}(\,\cdot\,; G, \mathbf{p}); \mathbf{p}') \\
&= p'_j \sum_{\mathcal{D}\in[N]\backslash\{j\}} \prod_{i\in\mathcal{D}} p_i \prod_{l\in[N]\backslash\mathcal{D}\cup\{j\}} (1-p_l)\mathcal{E}(\mathcal{D}\cup\{j\}) \\
&\quad + (1-p'_j) \sum_{\mathcal{D}\in[N]\backslash\{j\}} \prod_{i\in\mathcal{D}} p_i \prod_{l\in[N]\backslash\mathcal{D}\cup\{j\}} (1-p_l)\mathcal{E}(\mathcal{D}). \qquad (6.2)
\end{aligned}
$$

The first error term $\mathbb{P}_{err}(G, R_{map}(\,\cdot\,; G, \mathbf{p}); \mathbf{p})$ is of the form $p_j a + (1-p_j)b$, and the second error term $\mathbb{P}_{err}(G, R_{map}(\,\cdot\,; G, \mathbf{p}); \mathbf{p}')$ is of the form $p'_j a + (1-p'_j)b$. From Lemma 6.3, we have $\mathcal{E}(\mathcal{D}\cup\{j\}) \geq \mathcal{E}(\mathcal{D})$ and hence $a \geq b$. Since $a \geq b$ and $p'_j \leq p_j$, one can verify that $p_j a + (1-p_j)b \geq p'_j a + (1-p'_j)b$, and thus

$$
\mathbb{P}_{err}(G, R_{map}(\,\cdot\,; G, \mathbf{p}); \mathbf{p}) \geq \mathbb{P}_{err}(G, R_{map}(\,\cdot\,; G, \mathbf{p}); \mathbf{p}'),
$$

concluding the proof. □

Now one could repeatedly apply Theorem 6.1 on the prior vector $\mathbf{p}$ to conclude that any test matrix G should only do better on the reduced uniform prior vector $\mathbf{p}_{\min} = (p_{\min}, p_{\min}, ..., p_{\min})$ where $p_{\min} \triangleq \min_{i\in[N]} p_i$. On the other hand, the test matrix G should only do worse on the prior vector $\mathbf{p}_{\max} = (p_{\max}, p_{\max}, ..., p_{\max})$ where $p_{\max} \triangleq \max_{i\in[N]} p_i$. This is stated below without a formal proof.

Corollary 6.1. *Consider a test matrix G and a prior vector $\mathbf{p}$ such that $p_i \leq 0.5$ for all $i \in [T]$. Let $\mathbf{p}_{\min} = (p_{\min}, p_{\min}, ..., p_{\min})$ where $p_{\min} \triangleq \min_{i\in[N]} p_i$ and let $\mathbf{p}_{\max} = (p_{\max}, p_{\max}, ..., p_{\max})$ where $p_{\max} \triangleq \max_{i\in[N]} p_i$. Then*

$$\mathbb{P}^*_{err}(G, \mathbf{p}_{\max}) \geq \mathbb{P}^*_{err}(G, \mathbf{p}) \geq \mathbb{P}^*_{err}(G, \mathbf{p}_{\min}).$$

As a consequence of the above corollary, the number of tests required to attain a fixed (small) probability of error ϵ with prior vector $\mathbf{p}_{\min}$ is not more than the number of tests required to attain probability of error ϵ with prior vector $\mathbf{p}$. This observation allows us to use the lower bound on the number of tests when the priors are identical. This is made precise in the following theorem.

Theorem 6.2. *Consider the non-adaptive group testing problem with N items where the probability of item i being infected is $p_i \leq 0.5$. Let $p_{\min} \triangleq \min\limits_{i\in[N]} p_i$. In order to achieve a probability of error $\to 0$ as $N \to \infty$, the number of tests must be*

$$T(\mathbf{p}) = \Omega(\min\{N, Np_{\min} \log N\}).$$

Proof. From Corollary 6.1, suppose a test matrix G achieves a probability of error ϵ on prior vector $\mathbf{p}$, the same test matrix achieves a probability of error not more than ϵ on the prior vector $\mathbf{p}_{\min}$, where $\mathbf{p}_{\min} = (p_{\min}, p_{\min}, ..., p_{\min})$ and $p_{\min} \triangleq \min_{i\in[N]} p_i$. Any strategy that achieves a probability of error $\to 0$ as $N \to \infty$ with the prior vector $\mathbf{p}_{\min}$ requires a number of tests equal to $\Omega(\min\{N, Np_{\min} \log N\})$. Thus, we need at least as many tests with the prior vector $\mathbf{p}$. □

As discussed in Section 6.2.1, the entropy bound in Lemma 6.1 is an alternate lower bound on the number of tests needed for this problem. We note that the entropy bound might be greater or smaller than the term $Np_{\min} \log N$ in Theorem 6.2. In particular, if $p_i \leq 1/2\ \forall i$ it is easy to see that $\sum_{i=1}^{N} h_2(p_i) \geq N h_2(p_{\min}) \geq N p_{\min} \log {}^1\!/\!_{p_{\min}}$. However the term ${}^1\!/\!_{p_{\min}}$ may be smaller or larger than N; thus our bound, that applies independently of the value of $p_{\min}$ (as long as $p_i \leq 0.5$) cannot be directly derived from the entropy bound, and could be either greater or lesser than the entropy bound. Having said that, the main advantage of the lower bound in Theorem 6.2 is its particular form, which allows the proof of order-optimality of several static group testing algorithms, as we will see in the next subsection.

Now, if the prior vector $\mathbf{p}$ is "bounded", in the sense that the maximum entry and minimum entry in $\mathbf{p}$ differ by a constant factor (constant with respect to N), then the lower bound can be re-written in terms of the maximum entry in $\mathbf{p}$ or the mean of $\mathbf{p}$. Basically we here just use the fact that constant factors do not affect the order. We next make this corollary precise.

Definition 6.2 (Bounded priors)**.** Let $\eta \in [1, \infty)$ be a fixed constant (constant with respect to N). A prior vector $\mathbf{p}$ of length N is called $\eta-$bounded if

$$\frac{\max_i p_i}{\min_i p_i} \leq \eta.$$

Corollary 6.2 (Lower bound for bounded priors)**.** *Consider the non-adaptive group testing problem with N items where the probability of item i being infected is $p_i \leq 0.5$. Let $p_{\max} \triangleq \max_{i \in [N]} p_i$ and $p_{\text{mean}} \triangleq \frac{1}{N} \sum_{i=1}^{N} p_i$. Suppose $\mathbf{p} = (p_1, ..., p_N)$ is η-bounded for some constant η. Any strategy that achieves a probability of error $\to 0$ as $N \to \infty$ requires*

$$\begin{aligned} T(\mathbf{p}) &= \Omega(\min\{N, N p_{\text{mean}} \log N\}) \\ &= \Omega(\min\{N, N p_{\max} \log N\}). \end{aligned}$$

6.3.2 Performance of existing non-adaptive algorithms in the static non-identical priors

Suppose $\mathbf{p}$ is η-bounded and each $p_i \leq 0.5$. The following non-adaptive algorithms can be proved to be order-optimal with respect to the lower bound in Corollary 6.2:

- The Coupon Collector Algorithm (CCA) from [LCH14] for prior vector $\mathbf{p}$, as discussed in Section 6.2.1, achieves a probability of error less than $2N^{-\delta}$ with a number of tests less than $4e(1+\delta)Np_{\text{mean}}\log N$ (see Theorem 3 in [LCH14]). As a result, w.r.t to the lower bound in Corollary 6.2, either CCA is order-optimal (if $N \geq Np_{\text{mean}}\log N$) or individual testing is order optimal (if $N \leq Np_{\text{mean}}\log N$).

- As discussed in Section 6.2.1 for the group testing problem with identical priors (say every item is infected with the same probability p'), a variety of randomized and explicit algorithms have been proposed[3] which achieve a vanishing probability of error with a number of tests $O(Np'\log N)$. From Corollary 6.1, any test matrix that achieves a vanishing probability of error with $\mathbf{p}_{\text{max}}$ should also attain a vanishing probability of error with $\mathbf{p}$, and as a result $O(Np_{\text{max}}\log N)$ tests are sufficient for the prior vector $\mathbf{p}$. Consequently w.r.t our lower bound in Corollary 6.2, any of these designs is order optimal (if $N \geq Np_{\text{max}}\log N$) or individual testing is order optimal (if $N \leq Np_{\text{max}}\log N$).

6.3.3 Dynamic testing - bridging the gap

Given the discussion above, we next show conditions under which the prior probabilities of infections each day (these change everyday) are η-bounded and are each not more than 0.5. If these two conditions are satisfied everyday for our discrete-time SIR stochastic block model set-up in Fig. 6.3, then CCA and the other algorithms discussed in Section 6.3.2 are order-optimal for our dynamic testing problem formulation. (see 6.3). We first define some notation, building upon the notation in Section 6.2.3.

- $p_{\text{max}}^{(t)} \triangleq \max_j p_j^{(t)}$, the maximum probability of new infection on day t.

- $p_{\text{min}}^{(t)} \triangleq \min_j p_j^{(t)}$, the minimum probability of new infection on day t.

[3]Most of these were considered in the context of combinatorial priors. However, Theorem 1.7 and Theorem 1.8 from [AJS19] imply that any algorithm that attains a vanishing probability of error on the combinatorial priors, also attains a vanishing probability of error on the corresponding i.i.d probabilistic priors.

Theorem 6.3. *Consider the testing-intervention problem in Fig. 6.3 where the infections follow the discrete-time SIR stochastic block model* $(N, C, q_1, q_2, p_{\text{init}})$.

(i) Suppose $p_{\text{init}} \le 0.5$, $q_1 \le \frac{1-1/\sqrt{2}}{C}$ *and* $q_2 \le \frac{1-1/\sqrt{2}}{N}$, *then* $p_j^{(t)} \le 0.5$.

(ii) Suppose $\frac{q_1}{q_2} \le \eta$, *then* $\frac{p_{\max}^{(t)}}{p_{\min}^{(t)}} \le \eta$ *and as a result the prior vector for each day is* η*-bounded.*

Proof. We prove (i) first. We first have $p_j^{(0)} = p_{\text{init}} \le 0.5$. For $t \ge 1$, we have

$$\begin{aligned} p_j^{(t)} &= 1 - (1-q_1)^{I_j^{(t-1)}} (1-q_2)^{\sum_{j' \ne j} I_{j'}^{(t-1)}} \\ &\overset{(a)}{\le} 1 - (1-q_1)^C (1-q_2)^N \\ &\overset{(b)}{\le} 1 - (1 - Cq_1)(1 - Nq_2) \overset{(c)}{\le} 0.5, \end{aligned}$$

where in (a) we used the fact that the total number of infections inside a community and overall cannot be greater than C and N, respectively; (b) follows because of the algebraic inequality $(1+x)^y \ge 1 + xy$ if $x \ge -1$ and $y \notin (0,1)$; in (c) we used our assumptions about q_1 and q_2.

We next prove (ii). Since $q_1 \ge q_2$ in our model, we have

$$\begin{aligned} p_j^{(t)} &= 1 - (1-q_1)^{I_j^{(t-1)}} (1-q_2)^{\sum_{j' \ne j} I_{j'}^{(t-1)}} \\ &= 1 - \left(\frac{1-q_1}{1-q_2}\right)^{I_j^{(t-1)}} (1-q_2)^{\sum_{j'} I_{j'}^{(t-1)}} \\ &\le 1 - \left(\frac{1-q_1}{1-q_2}\right)^{\max_j I_j^{(t-1)}} (1-q_2)^{\sum_{j'} I_{j'}^{(t-1)}}, \end{aligned} \tag{6.3}$$

where $\max_j I_j^{(t)}$ is simply the maximum number of infections over all communities. Likewise,

$$\begin{aligned} p_j^{(t)} &= 1 - (1-q_1)^{I_j^{(t-1)}} (1-q_2)^{\sum_{j' \ne j} I_{j'}^{(t-1)}} \\ &\overset{(a)}{\ge} 1 - (1-q_2)^{\sum_{j'} I_{j'}^{(t-1)}}, \end{aligned} \tag{6.4}$$

where in (a) we used $q_2 \le q_1$. Combining (6.3) and (6.4) we have

$$\frac{p_{\max}^{(t)}}{p_{\min}^{(t)}} = \frac{1 - \left(\frac{1-q_1}{1-q_2}\right)^{\max_j I_j^{(t-1)}} (1-q_2)^{\sum_{j'} I_{j'}^{(t-1)}}}{1 - (1-q_2)^{\sum_{j'} I_{j'}^{(t-1)}}}$$

$$\overset{(a)}{\leq} \frac{1-\left(\frac{1-q_1}{1-q_2}\right)^{\max_j I_j^{(t-1)}} (1-q_2)^{\max_j I_j^{(t-1)}}}{1-(1-q_2)^{\max_j I_j^{(t-1)}}}$$

$$= \frac{1-(1-q_1)^{\max_j I_j^{(t-1)}}}{1-(1-q_2)^{\max_j I_j^{(t-1)}}} \overset{(b)}{\leq} \frac{q_1}{q_2} \leq \eta.$$

where (a) follows from the following facts: the function $f_1(x) = \frac{1-\kappa x}{1-x}$ is increasing for $\kappa \in (0,1)$, the function $f_2(x) = (1-q_2)^x$ is decreasing for $q_2 \in (0,1)$, and the sum $\sum_{j'} I_{j'}^{(t-1)}$ is lower bounded by $\max_j I_j^{(t-1)}$; and (b) follows from the fact that the function $f_3(x) = \frac{1-(1-q_1)^x}{1-(1-q_2)^x}$ is decreasing in $x \geq 1$ for $q_1 \geq q_2$, and therefore the maximum of the ratio is obtained for $\max_j I_j^{(t-1)} = 1$. All proofs of the above statements are provided in Appendix 6.6.4. □

Finally, we make three remarks related to the results introduced in this section.

Remark 1. Both assumptions (i) and (ii) on the parameters in Theorem 6.3 will hold true when the number of communities is a constant, i.e., the size of each community is $C = \Theta(N)$ (as is the case when the population is well-mixed, or if we just consider a single community); assumption (i) does not require $C = \Theta(N)$. In our simulations, we observed empirically that assumption (ii) also holds when $C << N$; we do not have a formal proof of Theorem 6.3 for this case however.

Remark 2. Our results hold not just for the specific model introduced in Section 6.2 (where in particular we assume symmetric intra and inter community transmissions) but for any underlying community structure where the two conditions (bounded prior vectors and the value of each prior not exceeding ½) are satisfied. For example, one could have a model where an infected individual can transmit the infection only to a subset of his fellow community members with probability q_1 (he/she cannot transmit to the rest of the individuals in his/her community) and only to a subset of individuals outside his/her community with probability q_2. For this example model, the conditions in Theorem 6.3 are sufficient to prove the two requirements on the prior vector.

Remark 3. Intervention is a crucial aspect for our results to hold true. Without intervention in our dynamic model, many of the prior probabilities would be greater than $1/2$ and our requirements on the prior vector would not be satisfied.

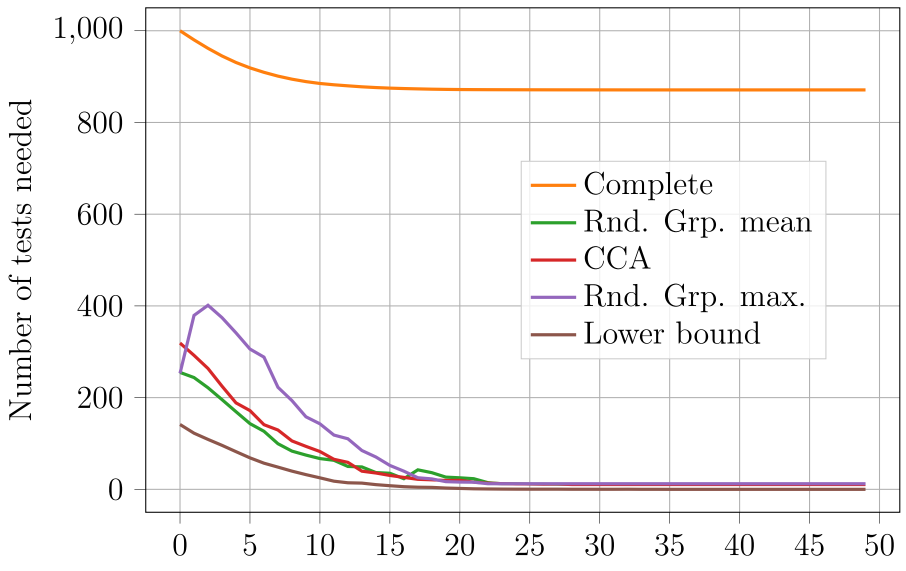
Remark: At this point, we want to acknowledge the fact that $I_{\max}$ cannot be known in advance, as it is itself a function of the model $(N, C, q_1, q_2, p_{\text{init}})$. However, a trivial bound for it is $I_{\max} \leq N$, which in turn returns a rather conservative condition for statement (i) of Theorem 6.3: $q_2 \leq q_1 \leq \frac{1}{2N}$.

6.4 Numerical results

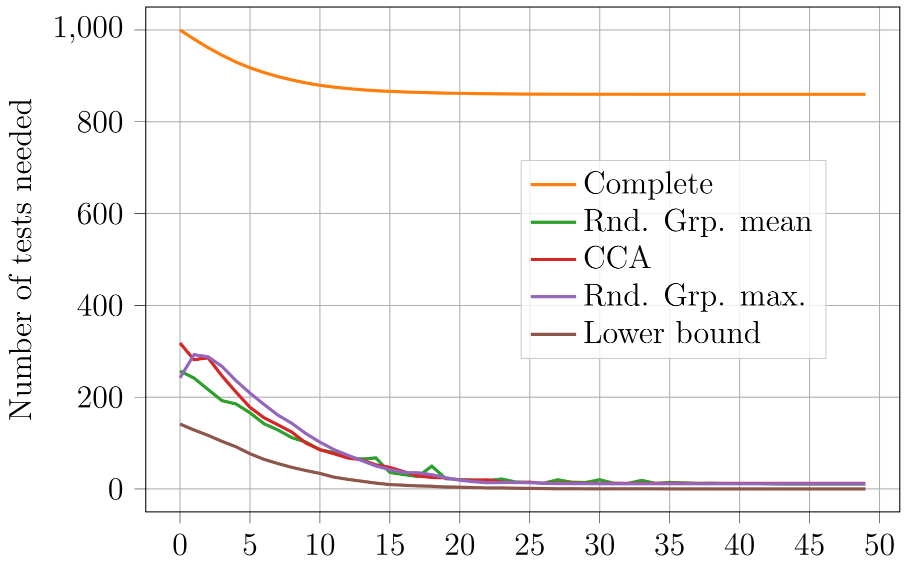
In this section, we show illustrative numerical results on the necessary and sufficient number of tests required for the discrete-time SIR stochastic block model. We next describe the experimental set-up.

- We simulate multiple instances (or *trajectories*) of the pipeline in Fig. 6.2 where the infections follow the discrete-time SIR stochastic block model $(N, C, p_{\text{init}}, q_1, q_2)$, and for different testing strategies. We simulate 200 trajectories and in Fig. 6.4, plot the daily average of the quantities across these trajectories.
- For each of these testing strategies, we empirically find the number of tests required on each day to identify all the infections on the previous day. To do this, on each day for a given trajectory, we start with 1000 tests and decrease this number (at a certain granularity) until the testing strategy makes a mistake. The smallest number of tests for which the strategy worked is plotted.
- On the other hand, we also plot the *entropy lower bound* in Lemma 6.1; it is easy to estimate this for our model via Monte-Carlo approximations. This bound holds for any set of values for $p_i^{(t-1)}$, regardless of whether the conditions required for Theorem 6.3 hold or not. The reason we use the entropy bound instead of our lower bound in Theorem 6.2 is that the entropy bound was numerically observed to be larger. Indeed, the lower bound in Theorem 6.2 contains some accompanying hidden constants which

are small when used for our particular choice of N.



(a) $(N, C, p_{\text{init}}, q_1, q_2) = (1000, 20, 0.02, 0.03, 0.0004)$.



(b) $(N, C, p_{\text{init}}, q_1, q_2 = (1000, 50, 0.02, 0.012, 0.0004)$.

Figure 6.4: Experimental results. We plot the average number of tests required by each strategy to identify the infection statuses of all non-isolated individuals each day for 2 different sets of parameters.

We compare the following testing strategies in our numerical simulations.

- **Complete testing**. We test every non-isolated individual remaining in the population each day.
- **Coupon Collector Algorithm (CCA) from [LCH14].** We showed the order-optimality of this algorithm for the dynamic testing problem at the beginning of Section 6.3.3. In short, on each day, the CCA algorithm constructs a random non-adaptive test design which depends on $p_j^{(t)}$. The idea is to place objects which are less likely to be infected in more number of tests and vice-versa. We refer the reader to [LCH14] for the exact description of the algorithm.
- **Random group testing for max probability (Rnd. Grp. max.)** Here we construct a randomized design assuming that each individual has a prior probability of infection $p_{\max}^{(t)}$. From Corollary 6.1, such a design must also work for the actual priors $p_j^{(t)}$. We construct a constant column-weight design (see e.g. [JAS19]) where each individual is placed in $L = \lfloor T/(Np_{\max}^{(t)} \log 2) \rfloor$ tests. Such a test design achieves a vanishing probability of error with $O(Np_{\max}^{(t)} \log N)$ tests (see for example [JAS19] for a proof), and hence is order-optimal under the conditions in Theorem 6.3.
- **Random group testing for mean probability (Rnd. Grp. mean)** Here we construct a randomized design assuming that each individual has a prior probability of infection $p_{\text{mean}}^{(t)}$, where $p_{\text{mean}}^{(t)}$ is defined as the mean prior probability of infection across all individuals. Unlike Rnd. Grp. max., there is no guarantee on how many tests are needed by such a design to identify the infection statuses of all individuals. However, the numerical results in Fig. 6.4 show that such a design requires fewer tests than CCA or Rnd. Grp. max. designs.

The numerical results in Fig. 6.4 are illustrated for two different parameter values of the discrete-time SIR stochastic block model. In both cases, we see that Rnd. Grp. mean requires the least number of tests to identify the infection statuses of all non-isolated individuals. Moreover, the number of tests required by all three testing strategies considered is much less than the number required by complete testing. In fact the numerics in Fig. 6.4 indicate that if we use a number of tests equal to $1/5$ of number of tests required for complete individual testing, all these algorithms would achieve the same performance as complete individual testing, at least for the particular examples that we considered.

A natural follow-up question to ask is if there is a systematic way to choose the number of tests that need to be administered, given the upper bounds discussed in Section 6.3.2. In Appendix 6.6.5, we discuss one such heuristic and show that it achieves close-to-complete-testing performance.

6.5 Conclusions and open questions

In this chapter, we proposed the problem of dynamic group testing which asks the question of how to continually test given that infections spread during the testing period. Our numerical results answer the question we started with – in the dynamic testing problem formulation, given a day of testing delay, is it possible to achieve close to complete testing performance with significantly fewer number of tests? The answer is yes, and in this chapter we not only showed numerical evidence supporting this fact, but also gave theoretical bounds on the optimal number of tests needed in order to achieve this.

Many open questions remain. In particular, it would be interesting to study the same problem when tests are noisy, or when we can use other models of group tests, or simply when one cannot perfectly learn the states of all individuals on a testing day. In addition, it would also be of interest to study/use other test designs. Finally, it remains open to see how these results translate to the continuous-time SIR stochastic network model.

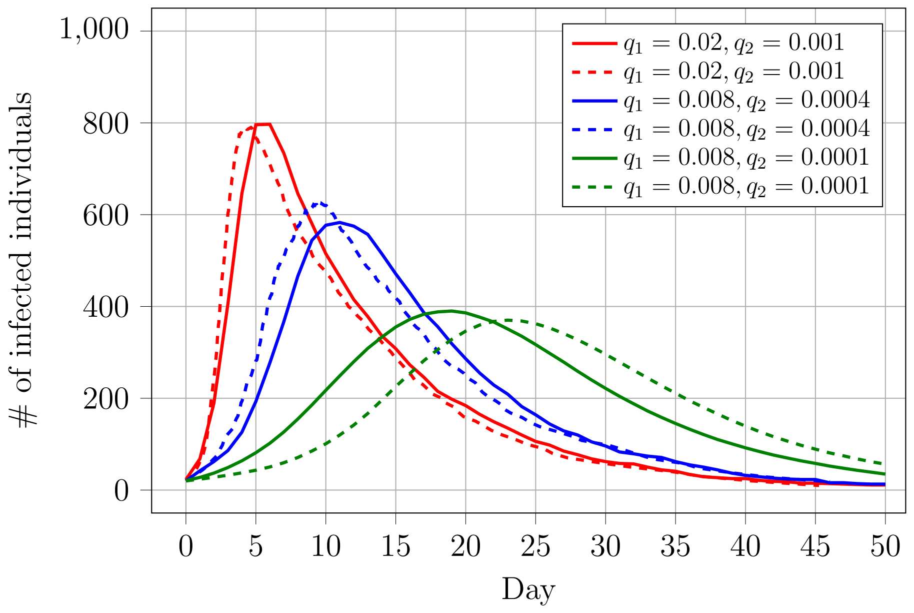


Figure 6.5: Continuous vs discrete-time model. Continuous model in dashed and discrete model in solid curves. Recovery probability $r = 0.1$ in all cases.

6.6 Appendix

6.6.1 Comparison of discrete and continuous-time SIR models

The well-studied continuous-time SIR stochastic network model from [KMS17] has been the main motivation for our discrete-time SIR stochastic block model. In fact, the discrete-time SIR stochastic block model described in Section 6.2.2 can be considered as a discretized version of the continuous-time SIR stochastic network model over the weighted graph, where 2 individuals belonging to the same community are connected by an edge with weight q_1 and 2 individuals belonging to different communities are connected by an edge with weight q_2, and recoveries occur at the rate r/day – i.e., an infected individual transmits the disease to a susceptible individual in the same community at the rate q_1/day and to a susceptible individual in a different community at the rate q_2/day. In Fig. 6.5, we compare the continuous-time model above and the discrete-time model for a few example values of q_1, q_2 and r for illustration.

We make a few observations:

• The progression of the disease in the discrete-time and continuous-time models, though not identical, follow a similar pattern, justifying the use of the discrete-time model.

• In both the models, ${}^1\!/_{q_1}$ is the expected time for an infected individual to transmit the disease to a susceptible individual in the same community, ${}^1\!/_{q_2}$ is the expected time for an infected individual to transmit the disease to a susceptible individual in a different community and ${}^1\!/_{r}$ is the expected time for an infected individual to recover.

• In the continuous-time model, an individual can get infected and recovered in the same day, whereas this is not possible in our discrete-time model (infected individuals can recover starting from the day after they are infected).

6.6.2 Proof of Lemma 6.2

The optimality of the MAP decoder is a standard result in statistics and signal processing. We however give the proof in the context of our problem, for completeness.

Lemma 6.2 (Optimality of MAP decoder)**.** *For given test matrix G and priors $\mathbf{p}$, the corresponding MAP decoder minimizes the probability of error for the test matrix under the given priors, i.e.,*

$$\mathbb{P}_{err}(G, R_{map}(\,\cdot\,; G, \mathbf{p}); \mathbf{p}) \leq \mathbb{P}_{err}(G, R; \mathbf{p}) \; \forall R.$$

Proof. As stated at the beginning of Section 6.3, the Probability of error for a test matrix, decoder pair under given the priors is

$$\begin{aligned}\mathbb{P}_{err}(G, R; \mathbf{p}) &\triangleq \underset{\mathbf{U}\sim\mathbf{p}}{\mathbb{E}} \mathbb{1}\{R(G(\mathbf{U})) \neq \mathbf{U}\} \\ &= \underset{\mathbf{Y}}{\mathbb{E}} \underset{\mathbf{U}|\mathbf{Y}}{\mathbb{E}} \mathbb{1}\{R(G(\mathbf{U})) \neq \mathbf{U}\},\end{aligned}$$

where $\mathbf{Y}$ is the set of test results. For the MAP decoder, the term inside $\mathbb{E}_{\mathbf{Y}}$ is

$$\underset{\mathbf{U}|\mathbf{Y}}{\mathbb{E}} \mathbb{1}\{R_{map}(\mathbf{Y}; G, \mathbf{p})) \neq \mathbf{U}\}$$

$$
\begin{aligned}
&= \mathop{\mathbb{E}}_{\mathbf{U}|\mathbf{Y}} \mathbb{1}\{\mathop{\arg\max}_{\mathbf{U}:G(\mathbf{U})=\mathbf{Y}} \Pr(\mathbf{U};\mathbf{p}) \neq \mathbf{U}\} \\
&= \sum_{\mathcal{D}\in[N]} \Pr\left(\mathbf{U}(\mathcal{D})|\mathbf{Y};\mathbf{p}\right) \cdot \mathbb{1}\{\mathop{\arg\max}_{\mathbf{U}(\mathcal{D}):G(\mathbf{U}(\mathcal{D}))=\mathbf{Y}} \Pr(\mathbf{U}(\mathcal{D});\mathbf{p}) \neq \mathbf{U}(\mathcal{D})\} \\
&= 1 - \Pr\left(\mathop{\arg\max}_{\mathbf{U}:G(\mathbf{U})=\mathbf{Y}} \Pr(\mathbf{U}|\mathbf{Y};\mathbf{p})\right) \\
&= 1 - \max_{\mathbf{U}:G(\mathbf{U})=\mathbf{Y}} \frac{\Pr(\mathbf{U};\mathbf{p})}{\Pr(\mathbf{Y})}. \qquad (6.5)
\end{aligned}
$$

Similarly, for any decoder R, we have

$$
\begin{aligned}
&\mathop{\mathbb{E}}_{\mathbf{U}|\mathbf{Y}} \mathbb{1}\{R(\mathbf{Y}) \neq \mathbf{U}\} \\
&\quad = \sum_{\mathcal{D}\in[N]} \Pr\left(\mathbf{U}(\mathcal{D})|\mathbf{Y};\mathbf{p}\right) \cdot \mathbb{1}\{R(\mathbf{Y}) \neq \mathbf{U}\} \\
&\quad = 1 - \Pr\left(R(\mathbf{Y})|\mathbf{Y};\mathbf{p}\right) \geq 1 - \max_{\mathbf{U}:G(\mathbf{U})=\mathbf{Y}} \frac{\Pr(\mathbf{U};\mathbf{p})}{\Pr(\mathbf{Y})}. \qquad (6.6)
\end{aligned}
$$

Comparing (6.5) and (6.6) concludes the proof. □

6.6.3 Proof of Lemma 6.3

Lemma 6.3. *Consider a test matrix G and priors $\mathbf{p}$. Suppose the corresponding MAP decoder is erroneous when identifying the defective set $\mathcal{D}$. Then the MAP decoder is also erroneous for the set of defectives is $\mathcal{D} \cup \{j\}$ with $p_j \leq 0.5$, i.e.,*

$$
\begin{aligned}
&\mathbb{1}\left\{R_{map}\left(G(\mathbf{U}(\mathcal{D}\cup\{j\}));G,\mathbf{p}\right) \neq \mathbf{U}(\mathcal{D}\cup\{j\})\right\} \\
&\quad \geq \mathbb{1}\left\{R_{map}(G(\mathbf{U}(\mathcal{D}));G,\mathbf{p}) \neq \mathbf{U}(\mathcal{D})\right\}.
\end{aligned}
$$

Proof. We first state the trivial case where $\mathbb{1}\left\{R_{map}(G(\mathbf{U}(\mathcal{D}));G,\mathbf{p}) \neq \mathbf{U}(\mathcal{D})\right\} = 0$. Under that assumption, the inequality of Lemma 6.3 always holds.

We then consider the case where $\mathbb{1}\left\{R_{map}(G(\mathbf{U}(\mathcal{D}));G,\mathbf{p}) \neq \mathbf{U}(\mathcal{D})\right\} = 1$, i.e., the MAP decoder makes an error when the defective set is $\mathcal{D}$. In that case, one of the two situations is possible:

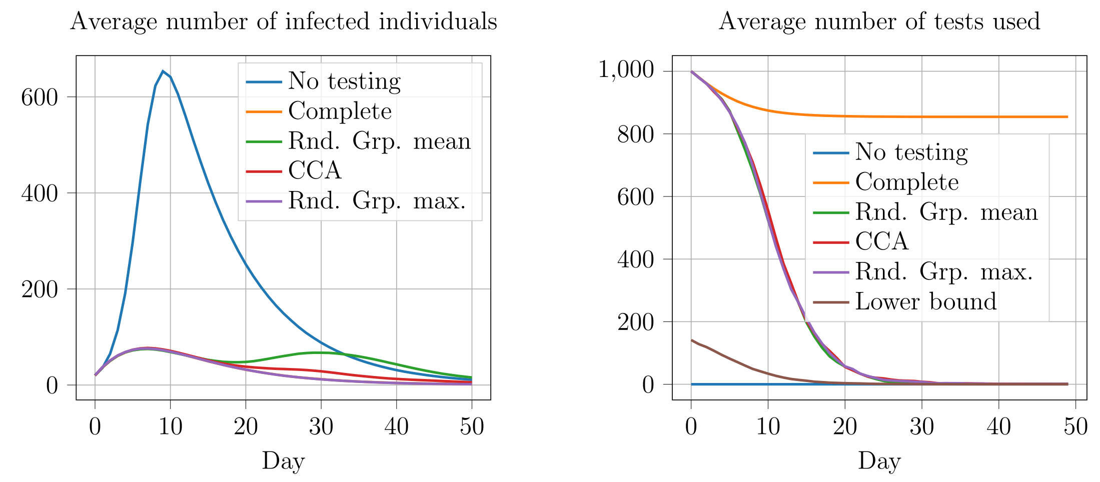
(1) there exists some set $\mathcal{D}' \neq \mathcal{D}$, such that $G(\mathbf{U}(\mathcal{D}')) = G(\mathbf{U}(\mathcal{D}))$ and $\Pr(\mathbf{U}(\mathcal{D}'); \mathbf{p}) > \Pr(\mathbf{U}(\mathcal{D}); \mathbf{p})$ or

(2) there exists some set $\mathcal{D}' \neq \mathcal{D}$, such that $G(\mathbf{U}(\mathcal{D}')) = G(\mathbf{U}(\mathcal{D}))$ and $\Pr(\mathbf{U}(\mathcal{D}'); \mathbf{p}) = \Pr(\mathbf{U}(\mathcal{D}); \mathbf{p})$ and $\mathcal{D}'$ is lexicographically earlier than $\mathcal{D}$.

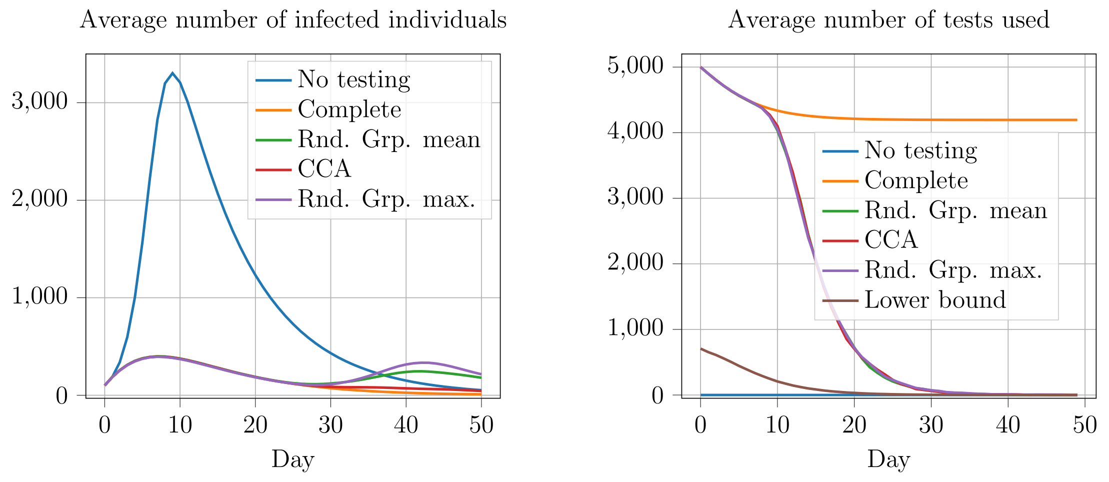
Hence MAP identifies incorrectly $\mathcal{D}'$ as the defective set given that $\mathcal{D}$ was the true defective set. We prove assuming that the first situation occurred; the proof follows identical arguments for the second situation.

Now, we consider two different cases for individual j that is added to $\mathcal{D} \cup \{j\}$:
(i) If $j \notin \mathcal{D}'$, then from our assumption in (1), notice that the defective set $\mathcal{D}' \cup \{j\}$ explains the test results of $\mathcal{D} \cup \{j\} - \mathcal{D}'$ gives the same test results as $\mathcal{D}$ and the extra individual j added to both the sets will still give the same results. We next claim that $\Pr(\mathbf{U}(\mathcal{D}' \cup \{j\}); \mathbf{p}) > \Pr(\mathbf{U}(\mathcal{D} \cup \{j\}); \mathbf{p})$, and consequently the MAP decoder will fail to correctly identify the defective set $\mathcal{D}' \cup \{j\}$. Now to prove our claim, we start with our assumption (1), i.e.,

$$\begin{aligned}
&\Pr(\mathbf{U}(\mathcal{D}'); \mathbf{p}) > \Pr(\mathbf{U}(\mathcal{D}); \mathbf{p}) \\
&\implies \prod_{i \in \mathcal{D}'} p_i \prod_{l \in [N] \setminus \mathcal{D}'} (1 - p_l) > \prod_{i \in \mathcal{D}} p_i \prod_{l \in [N] \setminus \mathcal{D}} (1 - p_l) \\
&\overset{(a)}{\implies} (1 - p_j) \prod_{i \in \mathcal{D}'} p_i \prod_{l \in [N] \setminus \mathcal{D}' \cup \{j\}} (1 - p_l) \\
&\qquad > (1 - p_j) \prod_{i \in \mathcal{D}} p_i \prod_{l \in [N] \setminus \mathcal{D} \cup \{j\}} (1 - p_l) \\
&\overset{(b)}{\implies} p_j \prod_{i \in \mathcal{D}'} p_i \prod_{l \in [N] \setminus \mathcal{D}' \cup \{j\}} (1 - p_l) \\
&\qquad > p_j \prod_{i \in \mathcal{D}} p_i \prod_{l \in [N] \setminus \mathcal{D} \cup \{j\}} (1 - p_l) \\
&\overset{(c)}{\implies} \prod_{i \in \mathcal{D}' \cup \{j\}} p_i \prod_{l \in [N] \setminus \mathcal{D}' \cup \{j\}} (1 - p_l) \\
&\qquad > \prod_{i \in \mathcal{D} \cup \{j\}} p_i \prod_{l \in [N] \setminus \mathcal{D} \cup \{j\}} (1 - p_l)
\end{aligned}$$



(a) $(N, C, p_{\text{init}}, q_1, q_2) = (1000, 50, 0.02, 0.012, 0.0004)$.



(b) $(N, C, p_{\text{init}}, q_1, q_2) = (5000, 50, 0.02, 0.012, 8 \times 10^{-5})$.

Figure 6.6: Experimental results for the heuristic procedure described in Appendix 6.6.5. We plot the average number of infected individuals and the number of tests used as a function of time (in days). For comparison, we also plot the performance when no one is tested (no testing) and when everyone is tested (Complete).

$$\implies \Pr(\mathbf{U}(\mathcal{D}' \cup \{j\}); \mathbf{p}) > \Pr(\mathbf{U}(\mathcal{D} \cup \{j\}); \mathbf{p}),$$

where in (a) we take out the term corresponding to j, also we use the fact that $j \notin \mathcal{D}$ and $j \notin \mathcal{D}'$; (b) follows from multiplying both sides with $p_j/1 - p_j$; in (c) we push the p_j term into the first product term.

(ii) If $j \in \mathcal{D}'$, we again first note that the defective set $\mathcal{D}' \cup \{j\} = \mathcal{D}'$ explains the test results of $\mathcal{D} \cup \{j\}$. We next claim that $\Pr(\mathbf{U}(\mathcal{D}'); \mathbf{p}) > \Pr(\mathbf{U}(\mathcal{D} \cup \{j\}); \mathbf{p})$, and consequently the MAP decoder will fail to correctly identify the defective set $\mathcal{D}' \cup \{j\}$. Now to prove our claim, we start with our assumption (1), i.e.,

$$\begin{aligned}
&\Pr(\mathbf{U}(\mathcal{D}'); \mathbf{p}) > \Pr(\mathbf{U}(\mathcal{D}); \mathbf{p}) \\
&\implies \prod_{i \in \mathcal{D}'} p_i \prod_{l \in [N] \setminus \mathcal{D}'} (1 - p_l) > \prod_{i \in \mathcal{D}} p_i \prod_{l \in [N] \setminus \mathcal{D}} (1 - p_l) \\
&\overset{(a)}{\implies} p_j \prod_{i \in \mathcal{D}' \setminus \{j\}} p_i \prod_{l \in [N] \setminus \mathcal{D}'} (1 - p_l) \\
&\qquad\qquad > (1 - p_j) \prod_{i \in \mathcal{D}} p_i \prod_{l \in [N] \setminus \mathcal{D} \cup \{j\}} (1 - p_l) \\
&\overset{(b)}{\implies} p_j \prod_{i \in \mathcal{D}' \setminus \{j\}} p_i \prod_{l \in [N] \setminus \mathcal{D}'} (1 - p_l) \\
&\qquad\qquad > p_j \prod_{i \in \mathcal{D}} p_i \prod_{l \in [N] \setminus \mathcal{D} \cup \{j\}} (1 - p_l) \\
&\overset{(c)}{\implies} \prod_{i \in \mathcal{D}'} p_i \prod_{l \in [N] \setminus \mathcal{D}'} (1 - p_l) \\
&\qquad\qquad > \prod_{i \in \mathcal{D} \cup \{j\}} p_i \prod_{l \in [N] \setminus \mathcal{D} \cup \{j\}} (1 - p_l) \\
&\implies \Pr(\mathbf{U}(\mathcal{D}'); \mathbf{p}) > \Pr(\mathbf{U}(\mathcal{D} \cup \{j\}); \mathbf{p}),
\end{aligned}$$

where in (a) we take out the term corresponding to j, also note that $j \notin \mathcal{D}$ but $j \in \mathcal{D}'$; (b) follows from the fact that $1 - p_j \geq p_j$ when $p_j \leq 0.5$, so we can replace the $(1 - p_j)$ term on the right-hand side by p_j without affecting the inequality; in (c) we push the p_j term into the first product term. □

6.6.4 Auxiliary results for Theorem 6.3

In this section we prove some auxiliary statements about functions $f_1(x)$, $f_2(x)$ and $f_3(x)$ that are used at the end of the proof of Theorem 6.3:

• $f_1(x) = \frac{1-\kappa x}{1-x}$ is increasing for $\kappa \in (0,1)$, because $f_1'(x) = -\frac{\kappa-1}{(x-1)^2} > 0$.

• $f_2(x) = (1-q_2)^x$ is decreasing for $q_2 \in (0,1)$, because $f_2'(x) = \ln(1-q_2)(1-q_2)^x < 0$.

• $f_3(x) = \frac{1-c_1^x}{1-c_2^x}$ is decreasing for $q_1 \geq q_2$, because of the following: Let $c_1 = 1-q_1$ and $c_2 = 1-q_2$, so that $c_2 \geq c_1$. Then,

$$\begin{aligned} f_3'(x) &= \frac{1}{(1-c_2^x)^2}\left((1-c_1)^x c_2^x \ln c_2 - (1-c_2^x)\, c_1^x \ln c_1\right) \\ &= \frac{1}{x\,(1-c_2^x)^2}\left((1-c_1)^x c_2^x \ln c_2^x - (1-c_2^x)\, c_1^x \ln c_1^x\right) \\ &= \frac{(1-c_1)^x(1-c_2)^x}{x\,(1-c_2^x)^2}\left(\frac{c_2^x \ln c_2^x}{(1-c_2)^x} - \frac{c_1^x \ln c_1^x}{(1-c_1)^x}\right) \overset{(a)}{\leq} 0, \end{aligned}$$

where (a) follows from the fact that $c_2 \leq c_1$ and the function $g(c) = \frac{c\ln c}{1-c}$ is non-increasing for $c \in (0,1)$. The latter can be seen by taking the derivative $g'(c) = \frac{\ln c - c + 1}{(1-c)^2}$, which is always non-positive for $c \in (0,1)$, as $\ln c \leq c-1$.

6.6.5 A heuristics for dynamic group testing

Given the results and discussion in Section 6.4, a natural question to ask is if one could use a number of tests based on the upper bounds discussed in Section 6.3.2. In particular, we focus on the upper bound for CCA which implies that CCA achieves a probability of error less than $2N^{-\delta}$ with a number of tests at most $4e(1+\delta)Np_{\text{mean}}\log N$ (see Theorem 3 in [LCH14]). Note that the probability of error is small, but not zero, for finite values of N. Here, we use a number of tests equal to $12eNp_{\text{mean}}\log N$ each day (corresponding to an error probability less than $2N^{-2}$) and plot the number of errors made by each of the three test designs considered in Section 6.4. The experimental set-up is as follows:

(i) We maintain an estimate of the probability $p_j^{(t)}$ that a susceptible individual belonging to community j becomes infected on day t (see Section 6.2.2 for the precise definition), for each community j, and for each day t.

(ii) At the beginning of day t, we obtain the results of the tests administered on day $t-1$. From these results, we form an estimate $\widehat{U}_i^{(t-1)}$ of the infection statuses $U_i^{(t-1)}$ of individual i at the beginning of day $t-1$, for each i. In order to learn the statuses, we use the *Definite Defective* (DD) decoder (see Section 2.4 in [AJS19]) which is guaranteed to have no false positives. Indeed, one could use more sophisticated decoders, such as ones based on loopy belief propagation. However, these decoders potentially give rise to both false positives and false negatives, resulting in an unfair comparison across different algorithms[4].

(iii) We isolate all individuals i where $\widehat{U}_i^{(t-1)} = 1$.

(iv) We update $p_j^{(t-1)}$ using our estimates $\widehat{U}_i^{(t-1)}$.

(v) Using our estimates of $p_j^{(t-1)}$, we estimate the value of $p_{\text{mean}}^{(t)}$ and choose a number of tests

$$T = \min\{12eN^{(t)}p_{\text{mean}}^{(t)} \log N^{(t)}, N^{(t)}\},$$

where $N^{(t)}$ is the current number of non-isolated individuals in the community. We next construct a testing matrix with T tests and administer these tests. For complete testing, we use $T = N^{(t)}$.

(vi) Steps $(ii) - (v)$ repeat each day.

Given the above set-up, Figure 6.6 compares the performance of the test designs described in Section 6.4. We make a few observations:

• We see that the algorithms do not always attain the performance of complete testing. This is due to the fact that for finite N, the probability of error is non-zero. However, as seen from Figure 6.6, the performance of CCA improves as N increases. On the other hand, Rnd. Grp. max. has the opposite trend; this is not surprising since the number of tests was chosen based on the upper bound for CCA and as a result there is no guarantee that the same number of tests is sufficient for Rnd. Grp. max.

• In comparison to the plots in fig. 6.4, the number of tests used here is much higher during

[4]Indeed, this begs the very complicated comparison between the impact of false positives and false negatives, which we avoid for the sake of simplicity.

the initial few days, which indicates the looseness of the upper bound; it remains open to show tighter upper bounds for these algorithms.

• Suppose we make an error when identifying the infection status on a particular day t, the estimates of $p_j^{(t-1)}$ are not exact, which in turn leads to potentially insufficient choices for the number of tests needed for subsequent days and inaccurate test designs. This drives an error accumulation and as a result the later days are more prone to error, as also seen in Figure 6.6.

CHAPTER 7

Conclusions and Open Questions

In this dissertation, we studied trace reconstruction and group testing through the lens of statistical inference. Moreover, we also showed how to leverage community structure side-information and epidemiological models to make group testing more efficient. As a part of this work, we introduced a number of theoretical tools that may be of general interest beyond the particular problems studied in this dissertation.

Deletion Channels and Trace Reconstruction

In Chapter 2, we provided, to the best of our knowledge, the first results and techniques to compute posterior distributions over multiple deletion channels. We also provided a new perspective on the maximum-likelihood for the deletion channel by showing an equivalence between a discrete optimization problem and a continuous formulation of it. In this process, we introduced a variety of tools (the relaxed binomial coefficient, edit graph and infiltration product) and demonstrated their use for analyzing deletion channels. We also presented numerical evaluations of our algorithms and showed performance improvements over existing trace reconstruction algorithms. One question that remains open is on coming up with error rate guarantees for the trace reconstruction algorithms introduced in this chapter. Extending these methods to insertion-deletion-substitutions channels is another open question.

Building on the continuous optimization equivalence idea from Chapter 2, in Chapter 3, we formulated the ML estimate over *any* system channel (which admits deletion channels as a special case) as a continuous optimization problem; in particular, we optimize the *expected likelihood function* over the space of product distributions for X, instead of optimizing the actual likelihood. This opens the door to the use of first-order heuristics like gradient ascent. We connected the SPs to the expected likelihood function and its gradient. As an application, we illustrated performance benefits of our formulations via numerics for the deletion channel. An open question is to understand for what classes of system channels these techniques offer benefits over existing methods, and how large these benefits are. Another open question is to understand how the ML solution relates to error rates, such as Hamming error rate, for the particular system channels considered in this chapter. Finally, it would also be interesting to come up with error-rate guarantees for the heuristics introduced in this chapter.

Group testing and Epidemiology

In Chapter 4, we used a similar approach as in Chapter 3, and formulated the search for optimal group-test designs, under the assumption of a DND decoder, as a non-convex optimization problem, and proposed a solution via enhanced gradient descent. Our solution is approximate in the sense that it minimizes a lower bound on the expected number of identification errors (as opposed to the exact expectation). But, our numerical evaluation, over various infection scenaria demonstrated that our approach can significantly outperform state-of-the-art designs (upto 58% in the best case). Moreover, our designs performed well with the DD decoder, which allows us to claim that test designs are transferable to other decoders. An open question is to extend the technique introduced in this chapter to other decoders, such as DD and belief propagation. It would also be interesting to extend these techniques to encompass more general infection models, in particular, to include correlated infection models. Performance guarantees for the heuristics introduced in this chapter is another open question.

While Chapter 4 considered independent infections, in Chapter 5, we studied group testing in the presence of correlations, where the correlation is induced by a community structure. We showed that exploiting the knowledge provided by the community structure allows us to use upto 60% fewer tests compared to traditional test designs. We also showed how to incorporate this information into a loopy belief propagation decoder to see performance benefits. An open question is to understand which community structures, beyond the examples considered in this chapter, offer benefits and how large these benefits can be. Another open question is to design algorithms for more general testing models (see [GPR20], for example) and/or under limited information about the community structure (for example, due to technological limitations, privacy issues and fast-changing structures).

Finally, in Chapter 6, we proposed the problem of dynamic group testing which asks the question of how to continually test given that infections spread during the testing period. Our numerical results answer the question we started with – in the dynamic testing problem formulation, given a day of testing delay, is it possible to achieve close to complete testing performance with significantly fewer number of tests? The answer is yes, and in this chapter we not only showed numerical evidence supporting this fact, but also gave theoretical bounds on the optimal number of tests needed in order to achieve this. Many open questions remain in this space. In particular, it would be interesting to study the same problem when tests are noisy, or when we can use other models of group tests, or simply when one cannot perfectly learn the states of all individuals on a testing day. In addition, it would also be of interest to study/use other test designs. It is also an open question to see how these results translate to the continuous-time SIR stochastic network model.

REFERENCES

[ABJ14] Matthew Aldridge, Leonardo Baldassini, and Oliver Johnson. "Group Testing Algorithms: Bounds and Simulations." *IEEE Transactions on Information Theory*, **60**(6):3671–3687, 2014.

[AFF20] Inés Armendáriz, Pablo A Ferrari, Daniel Fraiman, José M Martínez, and Silvina Ponce Dawson. "Group testing with nested pools." *arXiv preprint arXiv:2005.13650*, 2020.

[AJS16] Matthew Aldridge, Oliver Johnson, and Jonathan Scarlett. "Improved group testing rates with constant column weight designs." In *2016 IEEE International Symposium on Information Theory (ISIT)*, pp. 1381–1385. Ieee, 2016.

[AJS19] Matthew Aldridge, Oliver Johnson, and Jonathan Scarlett. "Group testing: an information theory perspective." *CoRR*, **abs/1902.06002**, 2019.

[Ald19] M. Aldridge. "Individual Testing Is Optimal for Nonadaptive Group Testing in the Linear Regime." *IEEE Trans. Inf. Theory*, **65**(4), 2019.

[AS12] George K. Atia and Venkatesh Saligrama. "Boolean Compressed Sensing and Noisy Group Testing." *IEEE Transactions on Information Theory*, **58**(3):1880–1901, 2012.

[AVD19] Mahed Abroshan, Ramji Venkataramanan, Lara Dolecek, and Albert Guillén i Fàbregas. "Coding for Deletion Channels with Multiple Traces.", 2019.

[BBL20] Ted Bergstrom, Carl T Bergstrom, and Haoran Li. "Frequency and accuracy of proactive testing for COVID-19." *medRxiv*, 2020.

[BCJ74] Lalit Bahl, John Cocke, Frederick Jelinek, and Josef Raviv. "Optimal decoding of linear codes for minimizing symbol error rate (corresp.)." *IEEE Transactions on information theory*, **20**(2):284–287, 1974.

[BKK04] Tuğkan Batu, Sampath Kannan, Sanjeev Khanna, and Andrew McGregor. "Reconstructing Strings from Random Traces." In *SODA '04*, pp. 910–918, 2004.

[BLS19] Joshua Brakensiek, Ray Li, and Bruce Spang. "Coded trace reconstruction in a constant number of traces.", 2019.

[BMT06] E. Berlekamp, R. McEliece, and H. van Tilborg. "On the Inherent Intractability of Certain Coding Problems (Corresp.)." *IEEE Trans. Inf. Theor.*, **24**(3):384–386, September 2006.

[BPS20] Wei Heng Bay, Eric Price, and Jonathan Scarlett. "Optimal Non-Adaptive Probabilistic Group Testing Requires $\Theta(\min\{k \log n, n\})$ Tests.", 2020.

[Bro20] Maria Broadfoot. “Coronavirus Test Shortages Trigger a New Strategy: Group Screening.” See `https://www.scientificamerican.com/article/coronavirus-test-shortages-trigger-a-new-strategy-group-screening2/`, May 2020.

[CGH20a] A. Coja-Oghlan, O. Gebhard, M. Hahn-Klimroth, and P. Loick. “Information-theoretic and algorithmic thresholds for group testing.” *IEEE Trans. Inf. Theory*, 2020.

[CGH20b] Amin Coja-Oghlan, Oliver Gebhard, Max Hahn-Klimroth, and Philipp Loick. “Optimal Group Testing.” volume 125 of *Proceedings of Machine Learning Research*, pp. 1374–1388, Jul. 2020.

[CGK12] George M Church, Yuan Gao, and Sriram Kosuri. “Next-generation digital information storage in DNA.” *Science*, **337**(6102):1628–1628, 2012.

[CGM19] Mahdi Cheraghchi, Ryan Gabrys, Olgica Milenkovic, and João Ribeiro. “Coded trace reconstruction.”, 2019.

[Cha19] Zachary Chase. “New Lower Bounds for Trace Reconstruction.”, 2019.

[CJS14] C. L. Chan, S. Jaggi, V. Saligrama, and S. Agnihotri. “Non-adaptive group testing: Explicit bounds and novel algorithms.” *IEEE Trans. Inf. Theory*, **60**(5):3019–3035, 2014.

[CKM12] Mahdi Cheraghchi, Amin Karbasi, Soheil Mohajer, and Venkatesh Saligrama. “Graph-constrained group testing.” *IEEE Transactions on Information Theory*, **58**(1):248–262, 2012.

[CLS19] Michael B Cohen, Yin Tat Lee, and Zhao Song. “Solving linear programs in the current matrix multiplication time.” In *Proceedings of the 51st annual ACM SIGACT symposium on theory of computing*, pp. 938–942, 2019.

[CNS19] Luis Ceze, Jeff Nivala, and Karin Strauss. “Molecular digital data storage using DNA.” *Nature Reviews Genetics*, **20**(8):456–466, 2019.

[CR19] Mahdi Cheraghchi and João Ribeiro. “Sharp analytical capacity upper bounds for sticky and related channels.” *IEEE Transactions on Information Theory*, 2019.

[CS16] Venkat. Chandrasekaran and Parikshit. Shah. “Relative Entropy Relaxations for Signomial Optimization.” *SIAM Journal on Optimization*, **26**(2):1147–1173, 2016.

[CTV20] Marco Cuturi, Olivier Teboul, and Jean-Philippe Vert. “Noisy Adaptive Group Testing using Bayesian Sequential Experimental Design.” *arXiv preprint arXiv:2004.12508*, 2020.

[DG01] Suhas N Diggavi and Matthias Grossglauser. "On transmission over deletion channels." In *Proceedings of the Annual Allerton Conference on Communication Control and Computing*, 2001.

[DG06] Suhas Diggavi and Matthias Grossglauser. "On information transmission over a finite buffer channel." *IEEE Transactions on Information Theory*, 2006.

[DH93] D-Z Du and F.K. Hwang. *Combinatorial Group Testing and Its Applications.* Series on Applied Mathematics, 1993.

[DMP] Suhas Diggavi, Michael Mitzenmacher, and H Pfister. "Capacity upper bounds for deletion channels." In *2007 ISIT*.

[Dor43] Robert Dorfman. "The Detection of Defective Members of Large Population." *The Annals of Mathematical Statistics*, **14**:436–440, 1943.

[DOS] Anindya De, Ryan O'Donnell, and Rocco A. Servedio. "Optimal Mean-based Algorithms for Trace Reconstruction." In *STOC 2017*.

[Ell20] Jordan Ellenberg. "Five People. One Test. This Is How You Get There." *NY-times*, May 2020.

[ERW08] Cees Elzinga, Sven Rahmann, and Hui Wang. "Algorithms for subsequence combinatorics." *Theoretical Computer Science*, **409**(3):394–404, 2008.

[FDA20] FDA. "Pooled Sample Testing and Screening Testing for COVID-19.", 2020.

[FKG71] Cees M Fortuin, Pieter W Kasteleyn, and Jean Ginibre. "Correlation inequalities on some partially ordered sets." *Communications in Mathematical Physics*, **22**(2):89–103, 1971.

[For73] G David Forney. "The viterbi algorithm." *Proceedings of the IEEE*, **61**(3):268–278, 1973.

[FP85] Ulrich Fincke and Michael Pohst. "Improved methods for calculating vectors of short length in a lattice, including a complexity analysis." *Mathematics of computation*, **44**(170):463–471, 1985.

[FRH83] K Farrell, L Rudolph, C Hartmann, and L Nielsen. "Decoding by local optimization (corresp.)." *IEEE transactions on information theory*, **29**(5):740–743, 1983.

[Gal62] Robert Gallager. "Low-density parity-check codes." *IRE Transactions on information theory*, **8**(1):21–28, 1962.

[GCW20] Ritesh Goenka, Shu-Jie Cao, Chau-Wai Wong, Ajit Rajwade, and Dror Baron. "Contact Tracing Enhances the Efficiency of COVID-19 Group Testing." *arXiv preprint arXiv:2011.14186*, 2020.

[GG20] Christian Gollier and Olivier Gossner. "Group testing against Covid-19." See `https://www.tse-fr.eu/publications/group-testing-against-covid-19`, April 2020.

[Gho20] Sabyasachi Ghosh et al. "Tapestry: A Single-Round Smart Pooling Technique for COVID-19 Testing." *medRxiv*, 2020.

[GPR20] Ryan Gabrys, Srilakshmi Pattabiraman, Vishal Rana, João Ribeiro, Mahdi Cheraghchi, Venkatesan Guruswami, and Olgica Milenkovic. "AC-DC: Amplification curve diagnostics for Covid-19 group testing." *arXiv preprint arXiv:2011.05223*, 2020.

[Gus97] Dan Gusfield. *Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology.* Cambridge University Press, New York, NY, USA, 1997.

[GV05] Venkatesan Guruswami and Alexander Vardy. "Maximum-Likelihood Decoding of Reed–Solomon Codes is NP-Hard." *Information Theory, IEEE Transactions on*, **51**:2249– 2256, 08 2005.

[GYA18] Parikshit S Gopalan, Sergey Yekhanin, Siena Dumas Ang, Nebojsa Jojic, Miklos Racz, Karen Strauss, and Luis Ceze. "Trace reconstruction from noisy polynucleotide sequencer reads.", July 26 2018. US Patent App. 15/536,115.

[HB98] Arash Hassibi and Stephen Boyd. "Integer parameter estimation in linear models with applications to GPS." *IEEE Transactions on signal processing*, **46**(11):2938–2952, 1998.

[HHW81] M. C. Hu, F. K. Hwang, and J. K. Wang. "A boundary problem for group testing." *SIAM Jour. on Algebraic Discrete Methods*, 1981.

[HL18] Nina Holden and Russell Lyons. "Lower bounds for trace reconstruction.", 2018.

[HM14] Bernhard Haeupler and Michael Mitzenmacher. "Repeated deletion channels." In *2014 IEEE Information Theory Workshop (ITW 2014)*, pp. 152–156. IEEE, 2014.

[HMP08] Thomas Holenstein, Michael Mitzenmacher, Rina Panigrahy, and Udi Wieder. "Trace Reconstruction with Constant Deletion Probability and Related Results." In *ACM-SIAM SODA '08*, pp. 389–398, 2008.

[HPP18] Nina Holden, Robin Pemantle, and Yuval Peres. "Subpolynomial trace reconstruction for random strings and arbitrary deletion probability." In *Proceedings of the 31st Conference On Learning Theory*, 2018.

[HV02] Babak Hassibi and Haris Vikalo. "On the expected complexity of integer least-squares problems." In *2002 IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 2, pp. II–1497. IEEE, 2002.

[JAS19] O. Johnson, M. Aldridge, and J. Scarlett. "Performance of Group Testing Algorithms With Near-Constant Tests Per Item." *IEEE Trans. Inf. Theory*, **65**(2):707–723, 2019.

[Joh17] O. T. Johnson. "Strong converses for group testing from finite block- length results." *IEEE Trans. Inf. Theory*, **63**(9), 2017.

[Kem77] JHB Kemperman. "On the FKG-inequality for measures on a partially ordered space." In *Indagationes Mathematicae (Proceedings)*, volume 80, pp. 313–331. North-Holland, 1977.

[KFL01] Frank R Kschischang, Brendan J Frey, and H-A Loeliger. "Factor graphs and the sum-product algorithm." *IEEE Transactions on information theory*, **47**(2):498–519, 2001.

[KKT03] David Kempe, Jon Kleinberg, and Éva Tardos. "Maximizing the spread of influence through a social network." In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 137–146, 2003.

[KLL20] Lauren M Kucirka, Stephen A Lauer, Oliver Laeyendecker, Denali Boon, and Justin Lessler. "Variation in false-negative rate of reverse transcriptase polymerase chain reaction–based SARS-CoV-2 tests by time since exposure." *Annals of Internal Medicine*, **173**:262–267, Aug. 2020.

[KMS17] Istvan Kiss, Joel Miller, and Péter Simon. *Mathematics of Epidemics on Networks*, volume 46. 01 2017.

[KZ12] Amin Karbasi and Morteza Zadimoghaddam. "Sequential group testing with graph constraints." In *2012 IEEE information theory workshop*, pp. 292–296. Ieee, 2012.

[LCH14] T. Li, C. L. Chan, W. Huang, T. Kaced, and S. Jaggi. "Group testing with prior statistics." In *2014 IEEE International Symposium on Information Theory*, pp. 2346–2350, 2014.

[LD09] Heng Li and Richard Durbin. "Fast and accurate short read alignment with Burrows–Wheeler transform." *Bioinformatics*, **25**(14):1754–1760, 05 2009.

[Lev01] Vladimir I Levenshtein. "Efficient reconstruction of sequences." *IEEE Transactions on Information Theory*, **47**(1):2–22, 2001.

[Lot97] M. Lothaire. *Combinatorics on Words*. Cambridge Mathematical Library. Cambridge University Press, 1997.

[Lot02] M. Lothaire. *Algebraic combinatorics on words*, volume 90. Cambridge University Press, 2002.

[Lot05] M. Lothaire. *Applied combinatorics on words*, volume 105. Cambridge University Press, 2005.

[MA16] Yaakov Malinovsky and Paul S. Albert. "Revisiting nested group testing procedures: new results, comparisons, and robustness." *American Statistician*, August 2016. See also `https://arxiv.org/abs/1608.06330`.

[Mal20] Smriti Mallapaty. "The mathematical strategy that could transform coronavirus testing.", 2020.

[MDK17] Wei Mao, Suhas N. Diggavi, and Sreeram Kannan. "Models and information-theoretic bounds for nanopore sequencing." *2017 ISIT*, 2017.

[Mit09] Michael Mitzenmacher. "A survey of results for deletion channels and related synchronization channels." *Probability Surveys*, **6**:1–33, 2009.

[MSO20] Tamás G Molnár, Andrew W Singletary, Gábor Orosz, and Aaron D Ames. "Safety-Critical Control of Compartmental Epidemiological Models with Measurement Delays." *IEEE Control Systems Letters*, **5**(5):1537–1542, 2020.

[NGF20] Pavlos Nikolopoulos, Tao Guo, Christina Fragouli, and Suhas Diggavi. "Community aware group testing." 2020. https://arxiv.org/abs/2007.08111.

[NO01] G. Nicosia and G. Oriolo. "Solving the Shortest Common Supersequence Problem." In *Operations Research Proceedings*, pp. 77–83, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg.

[NP17] Fedor Nazarov and Yuval Peres. "Trace Reconstruction with exp(O(N1/3)) Samples." In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2017, pp. 1042–1046, New York, NY, USA, 2017. ACM.

[NRG21] Pavlos Nikolopoulos, Sundara Rajan Srinivasavaradhan, Tao Guo, Christina Fragouli, and Suhas Diggavi. "Group testing for connected communities." In *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*, volume 130, pp. 2341–2349. PMLR, 2021.

[NSF21] Pavlos Nikolopoulos, Sundara Rajan Srinivasavaradhan, Christina Fragouli, and Suhas Diggavi. "Group testing for community-based infections." *IEEE BITS the Information Theory Magazine*, pp. 1–1, 2021.

[NSG21] Pavlos Nikolopoulos, Sundara Rajan Srinivasavaradhan, Tao Guo, Christina Fragouli, and Suhas Diggavi. "Group testing for overlapping communities." In *ICC 2021 - IEEE International Conference on Communications*, pp. 1–7, 2021.

[OAC18] Lee Organick, Siena Dumas Ang, Yuan-Jyue Chen, Randolph Lopez, Sergey Yekhanin, Konstantin Makarychev, Miklos Z Racz, Govinda Kamath, Parikshit Gopalan, Bichlien Nguyen, et al. "Random access in large-scale DNA data storage." *Nature biotechnology*, **36**(3):242–248, 2018.

[PC] P.C. Fishburn. *Encyclopedia of Mathematics*. European Mathematical Society. URL: `http://encyclopediaofmath.org/index.php?title=FKG_inequality&oldid=14368`.

[PS20] Eric Price and Jonathan Scarlett. "A Fast Binary Splitting Approach to Non-Adaptive Group Testing." *arXiv preprint arXiv:2006.10268*, 2020.

[PZ17] Yuval Peres and Alex Zhai. "Average-case reconstruction for the deletion channel: subpolynomially many traces suffice." *CoRR*, **abs/1708.00854**, 2017.

[Rat05] Edward A Ratzer. "Marker codes for channels with insertions and deletions." In *Annales des télécommunications*. Springer, 2005.

[RC00] L. Riccio and C. J. Colbourn. "Sharper bounds in adaptive group testing." *Taiwanese Journal of Mathematics*, p. 669–673, 2000.

[RM00] Edward A Ratzer and David JC MacKay. "Codes for channels with insertions, deletions and substitutions." In *In 2nd International Symposium on Turbo Codes and Related Topics*, 2000.

[RMR17] Cyrus Rashtchian, Konstantin Makarychev, Miklós Z Rácz, Siena Ang, Djordje Jevdjic, Sergey Yekhanin, Luis Ceze, and Karin Strauss. "Clustering Billions of Reads for DNA Data Storage." In *NIPS*, volume 2017, pp. 3360–3371, 2017.

[SC16] Jonathan Scarlett and Volkan Cevher. "Phase Transitions in Group Testing." In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*, pp. 40–53. SIAM, 2016.

[SDDa] S. R. Srinivasavaradhan, M. Du, S. Diggavi, and C. Fragouli. "On Maximum Likelihood Reconstruction over Multiple Deletion Channels." In *2018 IEEE International Symposium on Information Theory (ISIT)*.

[SDDb] S. R. Srinivasavaradhan, M. Du, S. Diggavi, and C. Fragouli. "Symbolwise MAP for Multiple Deletion Channels." In *2019 IEEE International Symposium on Information Theory (ISIT)*.

[SDD20] Sundara Rajan Srinivasavaradhan, Michelle Du, Suhas N Diggavi, and Christina Fragouli. "Algorithms for reconstruction over single and multiple deletion channels." *IEEE Transactions on Information Theory*, **67**(6):3389–3410, 2020.

[SDF20] Sundara Rajan Srinivasavaradhan, Suhas Diggavi, and Christina Fragouli. "Equivalence of ML decoding to a continuous optimization problem." In *2020 IEEE International Symposium on Information Theory (ISIT)*, pp. 343–348. IEEE, 2020.

[SGP21] Sundara Rajan Srinivasavaradhan, Sivakanth Gopi, Henry D Pfister, and Sergey Yekhanin. "Trellis BMA: Coded Trace Reconstruction on IDS Channels for DNA Storage." In *2021 IEEE International Symposium on Information Theory (ISIT)*, pp. 2453–2458. IEEE, 2021.

[SKC16] Ilan Shomorony, Samuel H. Kim, Thomas A. Courtade, and David N. C. Tse. "Information-optimal genome assembly via sparse read-overlap graphs." *Bioinformatics*, **32**(17):i494–i502, 2016.

[SNF21a] Sundara Rajan Srinivasavaradhan, Pavlos Nikolopoulos, Christina Fragouli, and Suhas Diggavi. "Dynamic group testing to control and monitor disease progression in a population." *arXiv preprint arXiv:2106.10765*, 2021.

[SNF21b] Sundara Rajan Srinivasavaradhan, Pavlos Nikolopoulos, Christina Fragouli, and Suhas Diggavi. "An entropy reduction approach to continual testing." In *2021 IEEE International Symposium on Information Theory (ISIT)*, pp. 611–616. IEEE, 2021.

[Sno55] John Snow. *On the mode of communication of cholera.* John Churchill, 1855.

[TKL21] Jussi Taipale, Ioannis Kontoyiannis, and Sten Linnarsson. "Population-scale testing can suppress the spread of infectious disease.", 2021.

[Tra20a] "Decentralized Privacy-Preserving Proximity Tracing." *arXiv preprint arXiv:2005.12273*, 2020.

[Tra20b] "Google COVID-19 Community Mobility Reports: Anonymization Process Description (version 1.0)." *arXiv preprint arXiv:2004.04145v2*, 2020.

[Tra21] Anne Trafton. "Could all your digital photos be stored as DNA?", June 2021.

[Tre11] Luca Trevisan. "Combinatorial optimization: exact and approximate algorithms." *Stanford University*, 2011.

[TRL20] Jussi Taipale, Paul Romer, and Sten Linnarsson. "Population-scale testing can suppress the spread of COVID-19." *MedRxiv*, 2020.

[TTV17] Eldho K Thomas, Vincent YF Tan, Alexander Vardy, and Mehul Motani. "Polar coding for the binary erasure channel with deletions." *IEEE Communications Letters*, **21**(4):710–713, 2017.

[Ung60] P. Ungar. "Cutoff points in group testing." *Comm. Pure Appl. Math*, **13**:49–54, 1960.

[Var97] Alexander Vardy. "Algorithmic Complexity in Coding Theory and the Minimum Distance Problem." In *Proceedings of the Twenty-Ninth Annual ACM Symposium on Theory of Computing*, STOC '97, p. 92–109, New York, NY, USA, 1997. Association for Computing Machinery.

[Ver20] Claudio Verdun et al. "Group testing for SARS-CoV-2 allows up to 10-fold efficiency increase across realistic scenarios and testing strategies." *medRxiv*, 2020.

[Xu14] Gongxian Xu. "Global optimization of signomial geometric programming problems." *European Journal of Operational Research*, **233**(3):500 – 510, 2014.

[YR20] Sergey Mikhailovich Yekhanin and Miklos Zoltan Racz. "Trace reconstruction from reads with indeterminant errors.", February 20 2020. US Patent App. 16/105,349.

[ZRB20] Junan Zhu, Kristina Rivera, and Dror Baron. "Noisy Pooled PCR for Virus Testing." *arXiv preprint arXiv:2004.02689*, 2020.