Algorithms for Reconstruction Over Single and Multiple Deletion Channels

Sundara Rajan Srinivasavaradhan, *Member, IEEE*, Michelle Du, Suhas N. Diggavi, *Fellow, IEEE*, and Christina Fragouli, *Fellow, IEEE*

***Abstract*— Recent advances in DNA sequencing technology and DNA storage systems have rekindled the interest in deletion channels. Multiple recent works have looked at variants of sequence reconstruction over a single and over multiple deletion channels, a notoriously difficult problem due to its highly combinatorial nature. Although works in theoretical computer science have provided algorithms which guarantee *perfect reconstruction* with multiple independent observations from the deletion channel, they are only applicable in the large blocklength regime and more restrictively, when the number of observations is also large. Indeed, with only a few observations, perfect reconstruction of the input sequence may not even be possible in most cases. In such situations, maximum likelihood (ML) and maximum aposteriori (MAP) estimates for the deletion channels are natural questions that arise and these have remained open to the best of our knowledge. In this work, we take steps to answer the two aforementioned questions. Specifically: 1. We show that solving for the ML estimate over the single deletion channel (which can be cast as a discrete optimization problem) is equivalent to solving its relaxation, a continuous optimization problem; 2. We exactly compute the symbolwise posterior distributions (under some assumptions on the priors) for both the single as well as multiple deletion channels. As part of our contributions, we also introduce tools to visualize and analyze error events, which we believe could be useful in other related problems concerning deletion channels.**

***Index Terms*— Deletion channels, trace reconstruction, symbolwise MAP, edit graph, dynamic programming.**

I. Introduction

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 [4]). To quote [4] – "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 work, we take steps in this direction.

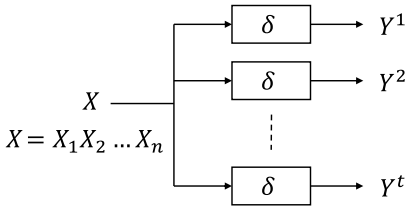

Fig. 1. The t-trace deletion channel model: the sequence X is passed through t independent deletion channels to yield t *traces*. We aim to estimate X from the Y^is.

In this space, the problem of *trace reconstruction*, as introduced in [5], has also received renewed interest in the past few years (see [6]–[12]). 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. 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.

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. 1. We call this the t-trace deletion channel model. We ask four main questions:

1) **Sequencewise maximum-likelihood with one trace:** For $t = 1$ (also called *single-trace deletion channel*, see Fig. 2), what is the maximum-likelihood estimate of X having

Manuscript received November 15, 2019; revised May 26, 2020; accepted September 17, 2020. Date of publication October 26, 2020; date of current version May 20, 2021. This work was supported in part by NSF under Grant 1705077 and Grant 1740047 and in part by the University of California-National Labs (UC-NL) under Grant LFR-18- 548554. This article was presented at the 2018 IEEE International Symposium on Information Theory, the 2019 IEEE International Symposium on Information Theory, and the 2020 IEEE International Symposium on Information Theory. *(Corresponding author: Sundara Rajan Srinivasavaradhan.)*

Sundara Rajan Srinivasavaradhan, Suhas N. Diggavi, and Christina Fragouli are with the Department of Electrical and Computer Engineering, University of California, Los Angeles, Los Angeles, CA 90095 USA (e-mail: sundar@ucla.edu; suhas@ee.ucla.edu; christina.fragouli@ucla.edu).

Michelle Du was with the University of California, Los Angeles (UCLA), Los Angeles, CA 90095 USA. She is now with Google, Seattle, WA 98103 USA (e-mail: michelleruodu@gmail.com).

Communicated by R. Gabrys, Guest Editor for the Special Issue: "From Deletion-Correction to Graph Reconstruction: In Memory of Vladimir I. Levenshtein."

Color versions of one or more of the figures in this article are available online at https://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIT.2020.3033513

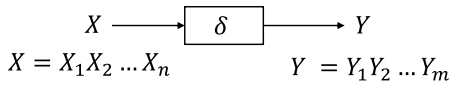

Fig. 2. The single-trace deletion channel model.

observed $Y = y$, i.e., a solution to

$$\arg\max_{x\in\{0,1\}^n} \Pr(Y = y|X = x).$$

2) **Sequencewise maximum-likelihood with multiple traces:** For a fixed t, with $t > 1$, what is the maximum-likelihood estimate of X having observed $Y^1 = y^1, Y^2 = y^2, \ldots, Y^t = y^t$, i.e.,

$$\arg\max_{x\in\{0,1\}^n} \Pr(Y^1 = y^1, Y^2 = y^2, \ldots, Y^t = y^t|X = x).$$

3) **Symbolwise MAP with one trace:** For $t = 1$ and $X_i \sim$ ind. Ber(p_i) in Fig. 2, 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. 1, what are the posterior distributions of X_i given all traces $Y^1 = y^1, Y^2 = y^2, \ldots, Y^t = y^t$, i.e., compute $\Pr(X_i = \alpha|Y^1 = y^1, Y^2 = y^2, \ldots, Y^t = y^t)$.

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 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 work) 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.[1]

- We introduce mathematical tools and constructs to visualize and analyze single-trace and t-trace deletion error events (see Section II).
- 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 III). 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 problem. Furthermore we also provide a polynomial time trace reconstruction heuristic with multiple traces that exploits this formulation.
- In Section IV, we prove the following:
  Theorem 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 V, we prove the following:
  Theorem 2: For the t-trace deletion channel model with priors $X_i \sim$ i.i.d. Ber(0.5) and observed traces $Y^1 = y^1, \ldots, Y^t = y^t$, the symbolwise posterior probabilities $\Pr(X_i = 1|Y^1 = y^1, \ldots, Y^t = y^t)\ \forall\ i$ 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 ([13]–[15]) extensively use algebraic tools for problems in the combinatorics of sequences (or *words*), and our work is inspired by such techniques. We borrow some notation and leverage a few of their results in our work.

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.[2]

[1] Some of the ideas presented in this paper can be found in [1], [2] and [3].
[2] As seen in [16], [17] there are more complicated effects of the nanopore reader not captured in this simple representation.

More related work. Our work falls under the general umbrella of sequence reconstruction over deletion channels (also see Levenshtein's work [18]), 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 ([19]–[21]); as are optimal coding schemes. Prior works have looked at the design of codes for deletion channels ([22]–[24]); 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 [25]–[27]) and we believe that the ideas in this work may prove useful in such a setting.

There are other works on sequence assembly (see for example, [28], [29]), where multiple short reads (from different segments of a sequence) are used to reconstruct the bigger sequence. This work 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).

Paper Organization. Section II introduces our notation and visualization tools for the single and t-trace channel error events; Section III 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 IV answers question 3) for the single-trace channel; Section V) answers question 4) for the t-deletion channel; Section VI gives numerical evaluations; and Section VII concludes the paper.

II. Notation and Tools

Basic notation: We borrow some notation from [13] 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 work, 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 $[i : j] \triangleq \{i, i+1, \ldots, j\}$ if $j \geq i$ and $[i : j] \triangleq \emptyset$ otherwise. We also define $[i] \triangleq [1 : i]$.

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

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

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

Binomial coefficient (section 6.3 in [13]): 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 [30]. For two sequences of lengths n and m, the binomial coefficient can be computed using a dynamic programming approach in $O(nm)$ (see [30] or Proposition 6.3.2 in [13]). 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

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

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:

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

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

$$\arg\max_{x \in \{0,1\}^n} \prod_{j=1}^{t} \binom{x}{y^j}. \tag{3}$$

As yet, there is no known efficient way to come up with a solution for either of the above two formulations (see [4]).

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

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:

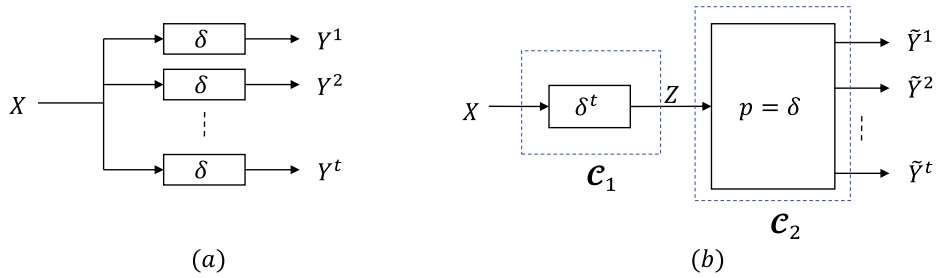

Fig. 3. A channel equivalence result: the t-trace deletion channel model in (a) is probabilistically equivalent to the the cascade of a deletion channel with the *remnant channel* ($\mathcal{C}_2$) in (b).

Definition 1:

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

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

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 B1). 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. 3). More precisely, we define the remnant channel as follows:

Definition 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.

Theorem 3: The t-deletion channel model and the cascade of the deletion channel with remnant channel shown in Fig. 3 are probabilistically equivalent, i.e.,

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

A rigorous proof of this theorem for arbitrary length sequences can be found in Appendix A1. A similar, though not equivalent, decomposition has been exploited in [31] 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 [31].

Edit graph ([32]): Similar graph constructs have been defined in related problems on common supersequences and subsequences (see [33] 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 $(i,j), 0 \le i \le |f|,\ 0 \le j \le |g|$. A directed edge $(i_1,j_1) \to (i_2,j_2)$ 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
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 $p = ((i_1,j_1),(i_2,j_2),\ldots,(i_m,j_m))$ 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. 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

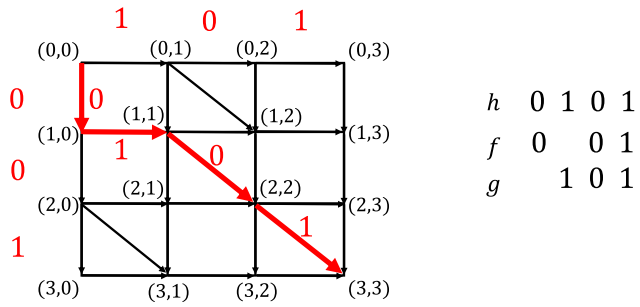

Fig. 4. Edit graph for sequences $f =$ '001' and $g =$ '101'. Make a grid so the vertical edges are aligned with a symbol in f and horizontal edges with g as shown. A diagonal edge $(i-1, j-1) \to (i, j)$ exists if $f_i = g_j$. The thick red edges form a path from the origin to the destination; this 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.

$s(p) \triangleq x_1 x_2 \dots x_{m-1}$ 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 $f_1, f_2, \dots, f_t$, construct a t-dimensional grid with a number of vertices $(|f_1|+1)(|f_2|+1)\dots(|f_t|+1)$ labeled from $(0, 0, \dots, 0)$ to $(|f_1|, |f_2|, \dots, |f_t|)$. A vertex $u = (i_1, i_2, \dots, i_t)$ is connected to $v = (j_1, j_2, \dots, j_t)$ (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., $(i_1, \dots, i_t)$ and $(j_1, \dots, j_t)$ 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 $f_{25} = f_{39}$. 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 $(0, \dots, 0)$ to be the origin of this graph and the vertex $(|f_1|, \dots, |f_t|)$ 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 [13]): The infiltration product has been extensively used in [13], 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 C. 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. 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 [13]). 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 [13]. The definition of infiltration extends to two polynomials via distributivity (precisely defined in Appendix C), and consequently to multiple sequences as well. For multiple sequences, infiltration has the same edit graph interpretation: $\langle f_1 \uparrow f_2 \uparrow \dots \uparrow f_t, w \rangle$ is the number of distinct ways of constructing w as a supersequence of $f_1, f_2, \dots, f_t$ so that the construction covers w, i.e., construct the t-dimensional edit graph of $f_1, f_2, \dots, f_t$ and count the number of paths corresponding to w.

III. Sequencewise ML for the Deletion Channel

A. A Continuous Optimization Formulation for the Single Trace ML

We here consider the single-trace ML decoding in (2), assuming that the output sequence $Y = y$ is non-empty. To the best of our knowledge, the only known method to solve (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 relaxation of (2) to obtain a solution for (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 1.

Lemma 1: For $p = (p_1, p_2, .., p_i, \dots, p_n)$ and $Y = y = y_1 \dots y_m$ with $n \geq m > 0$, we have

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

TABLE I
QUICK REFERENCE FOR NOTATION AND DEFINITIONS

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
$[i:j]$	$\{i, i+1, ..., j\}$
$x^{(i\to s)}$	$(x_1, x_2, ..., x_{i-1}, s, x_{i+1}, ..., x_n)$
$\binom{f}{g}$	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

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 A2. It is clear from Lemma 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) is equivalent to its real-valued relaxation. The following result makes this precise.

Theorem 4: The ML decoding problem for the single-trace deletion channel

$$\max_{x\in\{0,1\}^n} \binom{x}{y} \tag{4}$$

is equivalent to the problem

$$\max_{p\in[0,1]^n} \mathbf{F}(p, y). \tag{5}$$

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:

$$\max_{x\in\{0,1\}^n} \mathbf{F}(x, y) \equiv \max_{p\in[0,1]^n} \mathbf{F}(p, y), \tag{6}$$

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 $p = (p_1, p_2, \ldots, p_i, \ldots, p_n)$, at least one of the following holds true:

- $\mathbf{F}(p^{(i\to 0)}, y) \geq \mathbf{F}(p, y)$. Recall from notation that $p^{(i\to 0)} = (p_1, p_2, \ldots, p_{i-1}, 0, p_{i+1} \ldots, p_n)$ 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 (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 (5) and consequently to (4).

It remains to prove our claim. We use Lemma 1 to factor out p_i terms in $\mathbf{F}(p, Y)$:

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

Now we express $\mathbf{F}(p^{(i\to 0)}, y)$ and $\mathbf{F}(p^{(i\to 1)}, y)$ as

$$\begin{aligned}\mathbf{F}(p^{(i\to 0)}, y) &= \mathbf{F}(p_{[n]\setminus\{i\}}, y)\\ &\quad + \sum_{k|\, y_k=0} \mathbf{F}(p_{[1:i-1]}, y_{[1:k-1]})\mathbf{F}(p_{[i+1:n]}, y_{[k+1,m]}),\\ \mathbf{F}(p^{(i\to 1)}, y) &= \mathbf{F}(p_{[n]\setminus\{i\}}, y)\\ &\quad + \sum_{k|\, y_k=1} \mathbf{F}(p_{[1:i-1]}, y_{[1:k-1]})\mathbf{F}(p_{[i+1:n]}, y_{[k+1,m]}).\end{aligned}$$

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

$$\begin{aligned}&\min\left\{\mathbf{F}(p^{(i\to 0)}, y), \mathbf{F}(p^{(i\to 1)}, y)\right\}\\ &\quad\leq \mathbf{F}(p, y)\\ &\quad\leq \max\left\{\mathbf{F}(p^{(i\to 0)}, y), \mathbf{F}(p^{(i\to 1)}, y)\right\},\end{aligned}$$

thus proving our claim. □

The real-valued optimization problem in (5) falls under the umbrella of signomial optimization which is, in general, NP-hard (see for example, [34], [35]). A standard technique for signomial optimization uses convexification strategies to approximate the optimal value. In particular, as stated in [35], 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 1.

Algorithm 1 Single Trace Projected Gradient Ascent for ML

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

As a result, such strategies turn out to not be useful for the ML optimization problem. For instance, the techniques in [35] resulted in the bound $\mathbf{F}(p, Y) \leq \binom{|p|}{|Y|}$ 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, (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 (5) is solvable in polynomial time or not.

B. ML via Gradient Ascent

Given the continuous variable formulation of the ML problem in (5), a natural heuristic to find an estimate of the ML sequence is to employ *projected gradient ascent* to solve (5). Such projected gradient methods are well-known methods for constrained optimization problems (see [36] for example). The algorithm, in short, can be described as follows (the exact algorithm is detailed as Alg. 1):

Step I: Start from a randomly chosen interior point (in our case, we start from $p = (0.5, 0.5, \ldots, 0.5)$, 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 B2, we show using Lemma 1 that $\nabla_p \mathbf{F}(p, y)$ can be computed in $O(n^2)$ as a "by-product" of computing $\mathbf{F}(p, y)$.

C. A Heuristic for Multiple Traces

The continuous variable ML formulation in (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 $Y^1 = y^1, Y^2 = y^2, \ldots, Y^t = y^t$. The following theorem gives such a continuous optimization formulation with multiple traces.

Theorem 5: The ML decoding with multiple traces

$$\max_{x \in \{0,1\}^n} \binom{x}{y^1} \binom{x}{y^2} \cdots \binom{x}{y^t} \tag{7}$$

is equivalent to

$$\max_{p \in [0,1]^n} \mathbb{E}_{Z \sim p} \left[\binom{Z}{y^1} \binom{Z}{y^2} \cdots \binom{Z}{y^t} \right]. \tag{8}$$

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

Proof: This theorem can be proved in the same way as Theorem 4, by showing that $\mathbb{E}_{Z \sim p} \left[\binom{Z}{y^1} \binom{Z}{y^2} \cdots \binom{Z}{y^t} \right]$ 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 4.

To show this we use Lemma 2 stated below; this Lemma is also closely related to the channel equivalence of Theorem 3 (see Appendix A3).

Lemma 2: For $h, f_1, f_2, \ldots, f_m \in \mathcal{A}^*$,

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

Using Lemma 2, we now have

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

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

The formulation of (8), by itself, is not very useful as it is unclear on how to efficiently compute $\mathbb{E}_{Z \sim p} \left[\binom{Z}{y^1} \binom{Z}{y^2} \cdots \binom{Z}{y^t} \right]$. Indeed, if $\binom{Z}{y^i} \perp\!\!\!\perp \binom{Z}{y^j}$, the expectation of products would decompose into the product $\prod_j \mathbb{E}_{Z \sim p} \binom{Z}{y^j} = \prod_j \mathbf{F}(p, y^j)$, and each of the terms in the product can be computed in $O(n^2)$ as detailed in Appendix B1 – 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 $\arg\max_{p \in [0,1]^n} \mathbb{E}_{Z \sim p} \left[\binom{Z}{y^1} \ldots \binom{Z}{y^t} \right]$; Algorithm 2 makes this

Algorithm 2 Trace Reconstruction Heuristic via Projected Gradient Ascent

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

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 VI. As for the complexity, note that Alg. 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 $O(n^2 t)$.

IV. 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, where $X = X_1 \ldots X_n$, each input symbol is generated $X_i \sim$ ind. Ber (p_i), and we observe the trace $Y = y = y_1 y_2 \ldots y_m$ with $m \leq n$. Define the vector of priors as $p \triangleq (p_1, p_2, \ldots, p_n)$. 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

$$\begin{aligned}\Pr(X_i = 1 | Y = y) &= \frac{\Pr(X_i = 1, Y = y)}{\Pr(Y = y)} \\ &= \frac{\sum_{x | x_i = 1} \Pr(X = x) \Pr(Y = y | X = x)}{\sum_x \Pr(X = x) \Pr(Y = y | X = x)} \\ &\overset{(a)}{=} \frac{\sum_{x | x_i = 1} \Pr(X = x) \binom{x}{y}}{\sum_x \Pr(X = x) \binom{x}{y}}, \end{aligned} \tag{9}$$

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 6 expresses (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 (9).

Theorem 6: Let $X = X_1 \ldots X_n$ 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,

$$\begin{aligned}\Pr(X_i = 1 | Y = y) = \frac{p_i}{\mathbf{F}(p, y)} \Bigg(&\mathbf{F}(p_{[n] \setminus \{i\}}, y) \\ &+ \sum_{k | y_k = 1} \mathbf{F}(p_{[1:i-1]}, y_{[1:k-1]}) \mathbf{F}(p_{[i+1:n]}, y_{[k+1,m]}) \Bigg). \end{aligned} \tag{10}$$

Proof: The proof of this theorem employs the same trick used in the proof of Lemma 1. From (9), we have

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

Now,

$$\begin{aligned}\sum_{x | x_i = 1} \Pr(X = x) \binom{x}{y} &= \sum_{x | x_i = 1} \Pr(X = x) \sum_{\substack{\mathcal{S} \subseteq [n] \\ |\mathcal{S}| = m}} \mathbb{1}\{x_{\mathcal{S}} = y\} \\ &= \sum_{\substack{\mathcal{S} \subseteq [n] \\ |\mathcal{S}| = m}} \sum_{\substack{x | x_i = 1 \\ x_{\mathcal{S}} = y}} \Pr(X = x). \end{aligned} \tag{11}$$

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

$$\begin{aligned}&\sum_{\substack{\mathcal{S} \subseteq [n] \\ |\mathcal{S}| = m, i \notin \mathcal{S}}} \sum_{\substack{x | x_i = 1 \\ x_{\mathcal{S}} = y}} \Pr(X = x) = \sum_{\substack{\mathcal{S} \subseteq [n] \setminus \{i\} \\ |\mathcal{S}| = m}} \sum_{\substack{x | x_i = 1 \\ x_{\mathcal{S}} = y}} \Pr(X = x) \\ &= \sum_{\substack{S \subseteq [n] \setminus \{i\} \\ |\mathcal{S}| = m}} \sum_{\substack{x | x_i = 1 \\ x_{\mathcal{S}} = y}} \Big(\Pr(X_i = 1) \Pr(X_{\mathcal{S}} = y) \\ &\qquad \Pr(X_{[n] \setminus \mathcal{S} \cup \{i\}} = x_{[n] \setminus \mathcal{S} \cup \{i\}}) \Big) \\ &= \sum_{\substack{\mathcal{S} \subseteq [n] \setminus \{i\} \\ |\mathcal{S}| = m}} \Bigg(p_i \Pr(X_{\mathcal{S}} = y) \\ &\qquad \sum_{\substack{x | x_i = 1 \\ x_{\mathcal{S}} = y}} \Pr(X_{[n] \setminus \mathcal{S} \cup \{i\}} = x_{[n] \setminus \mathcal{S} \cup \{i\}}) \Bigg) \\ &= \sum_{\substack{\mathcal{S} \subseteq [n] \setminus \{i\} \\ |\mathcal{S}| = m}} \Bigg(p_i \Pr(X_{\mathcal{S}} = y) \\ &\qquad \sum_{x_j | j \in [n] \setminus \mathcal{S} \cup \{i\}} \Pr(X_{[n] \setminus \mathcal{S} \cup \{i\}} = x_{[n] \setminus \mathcal{S} \cup \{i\}}) \Bigg) \\ &= p_i \sum_{\substack{\mathcal{S} \subseteq [n] \setminus \{i\} \\ |\mathcal{S}| = m}} \Pr(X_{\mathcal{S}} = y) = p_i \mathbf{F}(p_{[n] \setminus \{i\}}, y). \end{aligned} \tag{12}$$

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:

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

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

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

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

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

$$=p_i\sum_{k|y_k=1}\mathbf{F}(p_{[1:i-1]},y_{[1:k-1]})\mathbf{F}(p_{[i+1:n]},y_{[k+1,m]}). \tag{13}$$

Plugging in (12) and (13) in (9) proves the theorem. □

Algorithm 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. 11
4: **for** $i=1:n$ **do**
5: Use (10) to compute $\Pr(X_i=1|Y=y)$

Alg. 3 summarizes the computation of $\Pr(X_i=1|Y=y)$. 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 B1. Given this, the algorithm iterates over the n indices and computes the posteriors in $O(n)$ for each of the indices. 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 4 Trace Reconstruction via Iterative Single-Trace Posterior Probabilities

1: Input: Traces $Y^1=y^1,\dots,Y^t=y^t$, input length n
2: Outputs: Estimate of the input $\hat{X}$
3: Initialize priors $p^{old}=p^{new}\leftarrow(0.5,0.5,\dots,0.5)$
4: **for** $l=1:t$ **do**
5: Use Alg. 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** $\hat{X}_1\hat{X}_2\dots\hat{X}_n$

A trace reconstruction heuristic with t traces. The posterior probability computation in Alg. 3 naturally gives rise to a trace reconstruction heuristic that updates the symbolwise statistics sequentially on the traces, where we use Alg. 3 with one trace at a time to continually update $\Pr(X_i=1|Y=y)$. The overall heuristic is described in Alg. 4. The complexity of Alg. 4 is $O(tn^2)$ since it runs Alg. 3 t times.

V. Symbolwise MAP for the t-Trace Deletion Channel

In this section, we put to use the ideas and constructs introduced in section II 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 D, 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 3.

Let $\mathcal{A}=\{0,1\}$, and assume that $X\sim$ Uniform $\mathcal{A}^n$. Our goal is to compute the symbolwise posterior probabilities $\Pr(X_i=1|Y^1=y^1,\dots,Y^t=y^t)$, where Y^j is the j^{th} trace. Our proposed algorithm is provided in Alg. 7 and estimates the symbolwise MAP (with uniform priors). We can directly leverage Alg. 7 to reconstruct the input as follows: for each index i, compute $\Pr(X_i=1|Y^1=y^1,\dots,Y^t=y^t)$ and decide

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

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

- We first give an expression for $\Pr(X_i=1|Y^1=y^1,\dots,Y^t=y^t)$ 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 $\Pr(X_i=1|Y^1=y^1,\dots,Y^t=y^t)$.

Theorem 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

$$\Pr(X_i = 1 | Y^1 = y^1, \ldots, Y^t = y^t) = \frac{c_1}{c_2}, \tag{14}$$

where c_1 and c_2 are

$$c_1 = \left[\sum_{k=0}^{n} 2^{n-k-1} \binom{n-1}{k} \sum_{w||w|=k} \langle y^1 \uparrow \ldots \uparrow y^t, w \rangle + \sum_{k=0}^{n} \sum_{j=1}^{k} 2^{n-k} \binom{i-1}{j-1} \binom{n-i}{k-j} \sum_{\substack{w||w|=k, \\ w_j=1}} \langle y^1 \uparrow \ldots \uparrow y^t, w \rangle \right],$$

$$c_2 = \left[\sum_{k=0}^{n} 2^{n-k} \binom{n}{k} \sum_{w||w|=k} \langle y^1 \uparrow \ldots \uparrow y^t, w \rangle \right].$$

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:

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

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 $\Pr(Y^1 = y^1, \ldots, Y^t = y^t) = \sum_{x||x|=n} \Pr(x) \Pr(Y^1 = y^1, \ldots, Y^t = y^t | X = x)$ we have,

$$\Pr(X_i = 1 | Y^1 = y^1, \ldots, Y^t = y^t) = \frac{\sum_{\substack{x||x|=n, \\ x_i=1}} \binom{x}{y^1} \ldots \binom{x}{y^t}}{\sum_{x||x|=n} \binom{x}{y^1} \ldots \binom{x}{y^t}}. \tag{15}$$

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

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

where (a) is due to Lemma 2 and (b) due to Lemma 3 (both introduced in [1]); see Appendix A3 and Appendix A4 for the statement and proof.

Therefore we have,

$$\begin{aligned}
&\sum_{\substack{x||x|=n, \\ x_i=1}} \binom{x}{y^1} \cdots \binom{x}{y^t} \\
&\overset{(a)}{=} \sum_{k=0}^{\infty} 2^{n-k-1} \binom{n-1}{k} \sum_{w||w|=k} \langle y^1 \uparrow \ldots \uparrow y^t, w \rangle \\
&\quad + \sum_{k=0}^{\infty} \sum_{j=1}^{k} 2^{n-k} \binom{i-1}{j-1} \binom{n-i}{k-j} \sum_{\substack{w||w|=k, \\ w_j=1}} \langle y^1 \uparrow \ldots \uparrow y^t, w \rangle \\
&\overset{(b)}{=} \sum_{k=0}^{n} 2^{n-k-1} \binom{n-1}{k} \sum_{w||w|=k} \langle y^1 \uparrow \ldots \uparrow y^t, w \rangle \\
&\quad + \sum_{k=0}^{n} \sum_{j=1}^{k} 2^{n-k} \binom{i-1}{j-1} \binom{n-i}{k-j} \sum_{\substack{w||w|=k, \\ w_j=1}} \langle y^1 \uparrow \ldots \uparrow y^t, w \rangle,
\end{aligned} \tag{16}$$

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

$$\sum_{x||x|=n} \binom{x}{y^1} \cdots \binom{x}{y^t} = \sum_{k=0}^{n} 2^{n-k} \binom{n}{k} \sum_{w||w|=k} \langle y^1 \uparrow \ldots \uparrow y^t, w \rangle. \tag{17}$$

Plugging (16) and (17) in (15), we get the expression in Theorem 7,

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

□

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

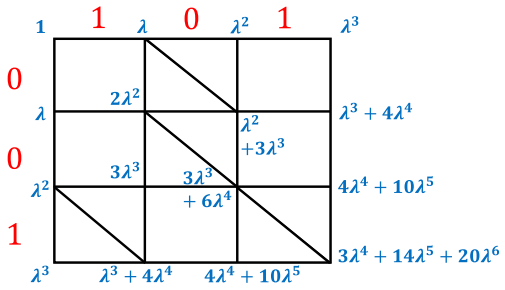

Fig. 5. The forward-potential $p_v^{for}(\lambda)$ at each vertex.

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

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

(b) $\sum_{\substack{w||w|=k,\\ w_j=1}} \langle y^1 \uparrow \ldots \uparrow y^t, w\rangle$ 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:

$$p_v^{for}(\lambda) = \sum_{u|u\to v} \lambda p_u^{for}(\lambda). \tag{18}$$

With this definition, we have

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

In the example in Fig. 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. 4, the forward-potentials are shown in Fig. 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 $O(n)$.

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 $(u\to v)$.

Algorithm 5 Computing the Forward-Potentials $p_u^{for}(\lambda)$

1: Input: Edit graph $\mathcal{G}(y^1,\ldots,y^t)$
2: Outputs: $p_v^{for}(\lambda)\ \forall\ v$
3: Order the vertices from $(0,0,\ldots,0)$ to $(|y^1|,|y^2|,\ldots,|y^t|)$ lexicogaphically; let the ordered list be $\mathcal{V}$
4: Initialise $p_{(0,\ldots,0)}^{for}(\lambda) \leftarrow 1$
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 6 Computing the Reverse-Potentials $p_u^{rev}(\lambda)$

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

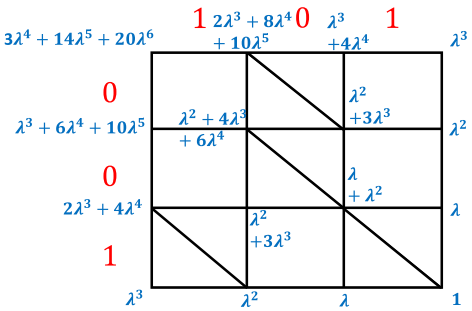

Fig. 6. The reverse-potential $p_v^{rev}(\lambda)$ at each vertex.

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:

$$p_v^{rev}(\lambda) = \sum_{u|v\to u} \lambda p_u^{rev}(\lambda), \tag{19}$$

with $p_{destination}^{rev}(\lambda) = 1$. The reverse potentials for the example in Fig. 4 is shown in Fig. 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:

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

Alg. 7 now summarizes the computation of the posterior probabilities. This algorithm iterates over all the edges

Algorithm 7 Symbolwise MAP With t Traces

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

Algorithm 8 Trace Reconstruction via Independent Posterior Combination

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

Algorithm 9 Bitwise Majority Alignment

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

(we have $O((2n)^t)$ of these), and also k, j ($O(n)$ each). The time complexity of Alg. 7 hence is $O(2^t n^{t+2})$.

VI. 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 F for a proof). We also use edit error rate as it is a typical metric used in the context of insertion/deletion channels.

Baseline algorithms:

1) **Independent posterior combination:** As pointed in the introduction, 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. 8. The complexity of this algorithm is $O(n^2 t)$ since computing the posteriors takes $O(n^2)$ and we compute posteriors for t traces.
2) **Bitwise Majority Alignment (introduced in [5]):** 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. 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 [6], [8], [10]). 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 [6]). 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. 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 [37]); 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

TABLE II

LIST OF TRACE RECONSTRUCTION ALGORITHMS COMPARED IN THE WORK

List of trace reconstruction algorithms compared in this work.		
Abbreviation	**Description**	**Complexity**
Ind. post. comb.	Independent posterior combination (Alg. 8)	$O(n^2t)$
BMA	Bitwise majority alignment of [5] (Alg. 9)	$O(nt)$
Trace stats.	Algorithm based on trace symbolwise statistics from [6] (Alg. 10)	$O(n^{3.37} + nt)$
Grad asc.	Projected gradient ascent (Alg. 2)	$O(n^2t)$
SMAP seq.	Sequential symbolwise MAP heuristic (Alg. 4)	$O(n^2t)$
SMAP exact	Exact symbolwise MAP (Alg. 7)	$O(n^{t+2}2^t)$

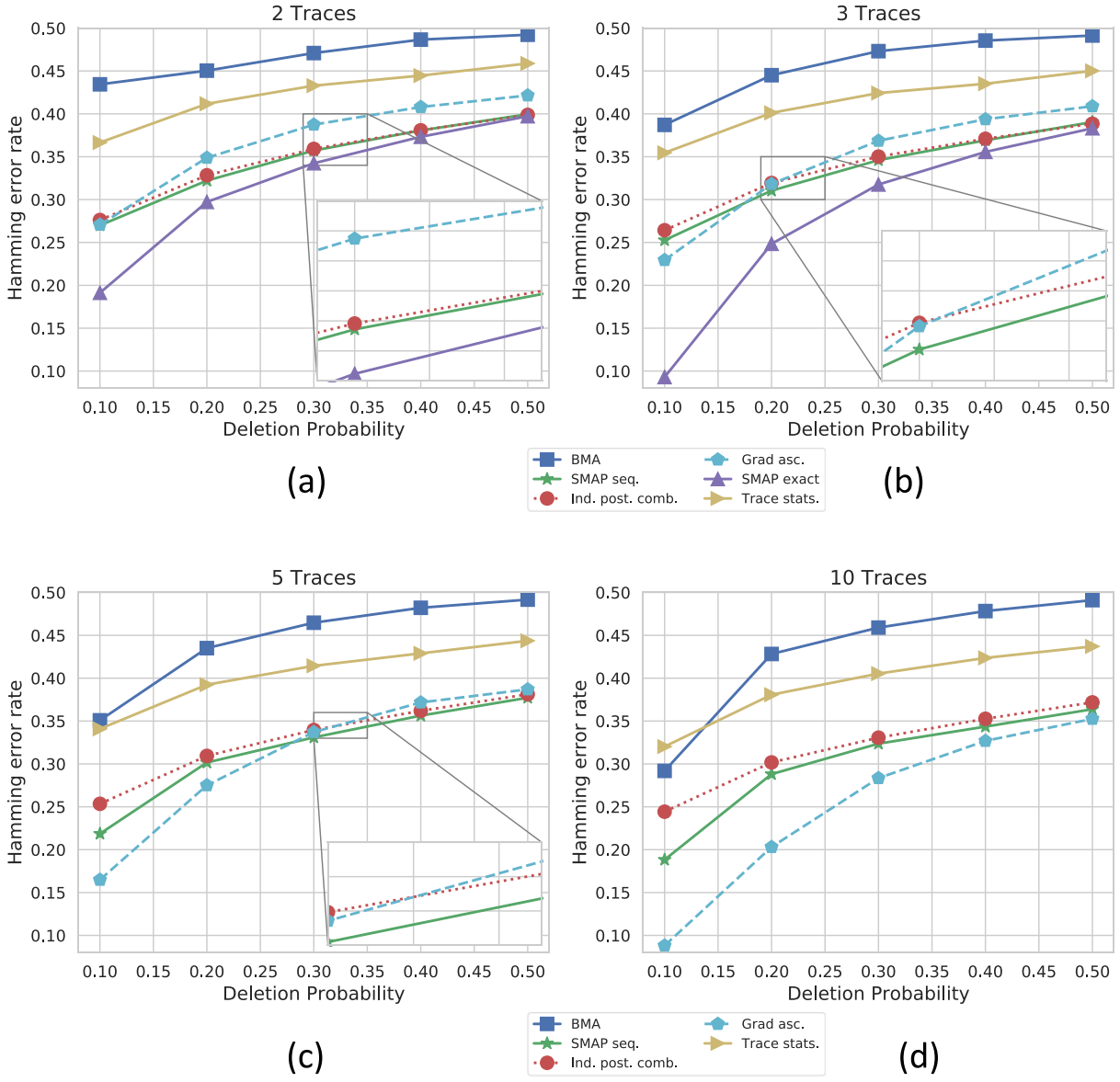

Fig. 7. Comparison of Hamming error rates for a blocklength $n = 100$ illustrated with 2, 3, 5 and 10 observed traces. Note that we do not run SMAP exact. for 5 and 10 traces since its complexity grows exponentially with the number of traces. All the subplots are plotted on the same scale to aid comparability across subplots. Few of the subplots which contain algorithms with similar error rates also contain a zoomed-in inset view.

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).

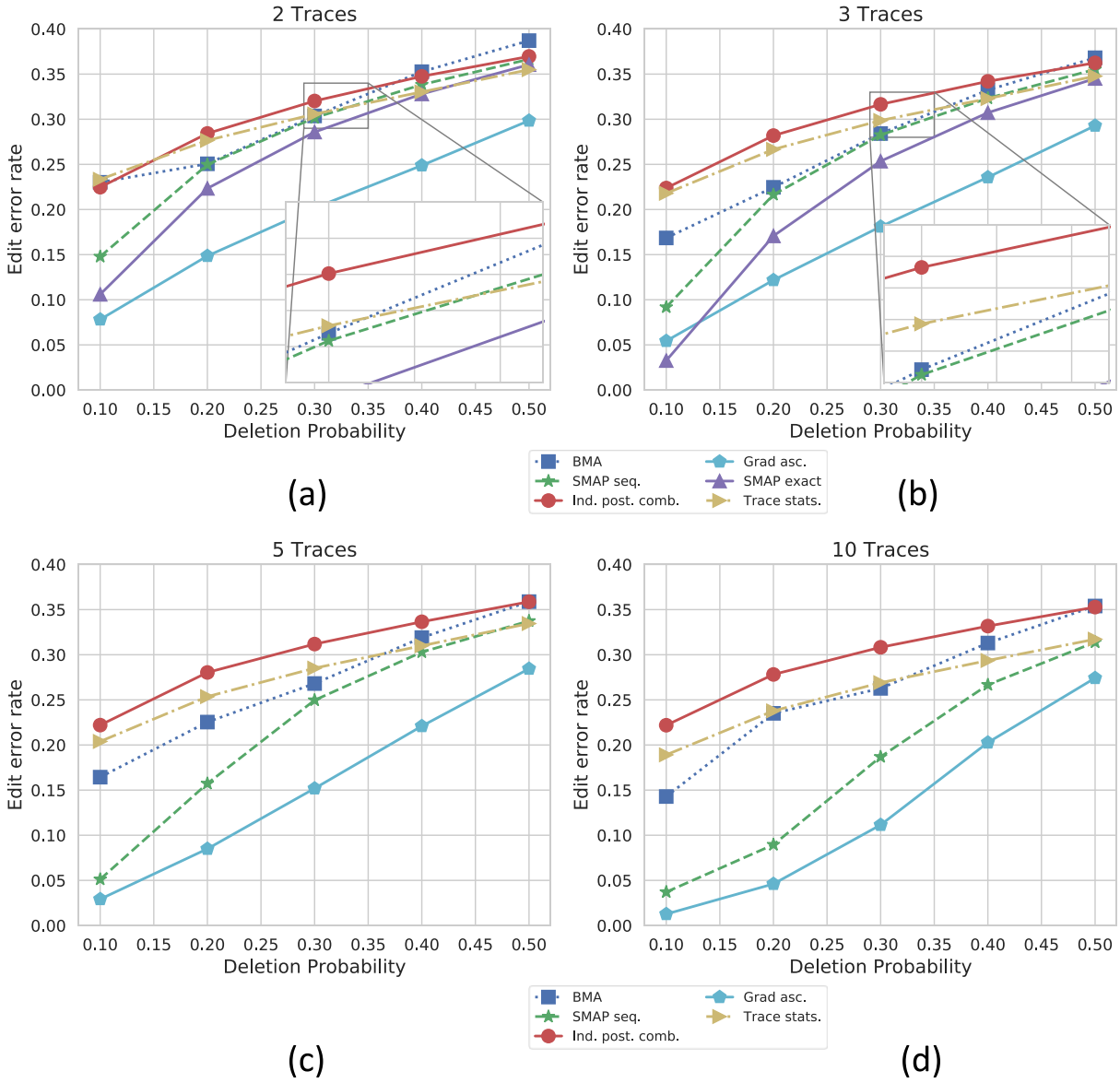

Fig. 8. Comparison of edit error rates for a blocklength $n = 100$ illustrated with 2, 3, 5 and 10 observed traces. Note that we do not run SMAP exact. for 5 and 10 traces since its complexity grows exponentially with the number of traces. All the subplots are plotted on the same scale to aid comparability across subplots. Few of the subplots which contain algorithms with similar error rates also contain a zoomed-in inset view.

Algorithm 10 Trace Statistics Heuristic

1: Input: Traces $Y^1 = y^1, \ldots, Y^t = y^t$, input length n
2: Output: estimate of input $\hat{X} = \hat{X}_1\hat{X}_2 \ldots \hat{X}_n$.
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 [6] 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 $\hat{X}_i = x_i \leftarrow 1$.

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

Algorithms introduced in this paper:

1) **Projected gradient ascent:** Alg. 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. 4 used as described.

3) **Exact symbolwise MAP:** Alg. 7 used as described.

Observations: In Fig. 7 and Fig. 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. 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. 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. 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. 8 (a) and (b).

VII. Conclusion

In this work 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 its relaxed version. 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.

Appendix

A. Proofs

1) Proof of Theorem 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.

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.,

$$\begin{aligned}\Pr(D_1 =&d_1, D_2 = d_2, \ldots, D_n = d_n)\\ &= \Pr(\widetilde{D}_1 = d_1, \widetilde{D}_2 = d_2, \ldots, \widetilde{D}_n = d_n),\end{aligned}$$

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., $D_i \perp\!\!\!\perp D_j$ for $i \neq j$. So it is sufficient to prove that the distributions of each D_i and $\widetilde{D}_i$ are the same.

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:

- $d_i = (-,-,-,\ldots,-)$, 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, $\Pr(\widetilde{D}_i = d_i) = \Pr(\breve{D}_i^0 = -) = \delta^t$.
- $d_i \neq (-,-,-,\ldots,-)$, 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,

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

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

2) Proof of Lemma 1:

Lemma 1: For $p = (p_1, p_2, .., p_i, \ldots, p_n)$ and $Y = y = y_1 \ldots y_m$ with $n \geq m > 0$, we have

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

Proof: The proof of this lemma uses a similar approach as the proof of Thm. 6. First, in the expression for $\mathbf{F}(\cdot)$, we separate out the subsets that contain index i:

$$\begin{aligned}\mathbf{F}(p, y) &= \sum_{\substack{\mathcal{S}|\mathcal{S}\subseteq[n],\\ |\mathcal{S}|=m}} \prod_{j=1}^{m} p_{\mathcal{S}_j}^{y_j}(1-p_{\mathcal{S}_j})^{1-y_j}\\ &= \sum_{\substack{\mathcal{S}|\mathcal{S}\subseteq[n],\\ |\mathcal{S}|=m,\\ i\notin\mathcal{S}}} \prod_{j=1}^{m} p_{\mathcal{S}_j}^{y_j}(1-p_{\mathcal{S}_j})^{1-y_j}\\ &+ \sum_{\substack{\mathcal{S}|\mathcal{S}\subseteq[n],\\ |\mathcal{S}|=m,\\ i\in\mathcal{S}}} \prod_{j=1}^{m} p_{\mathcal{S}_j}^{y_j}(1-p_{\mathcal{S}_j})^{1-y_j}\\ &= \mathbf{F}(p_{[n]\setminus\{i\}}, y) + \sum_{\substack{\mathcal{S}|\mathcal{S}\subseteq[n],\\ |\mathcal{S}|=m,\\ i\in\mathcal{S}}} \prod_{j=1}^{m} p_{\mathcal{S}_j}^{y_j}(1-p_{\mathcal{S}_j})^{1-y_j}.\end{aligned} \tag{20}$$

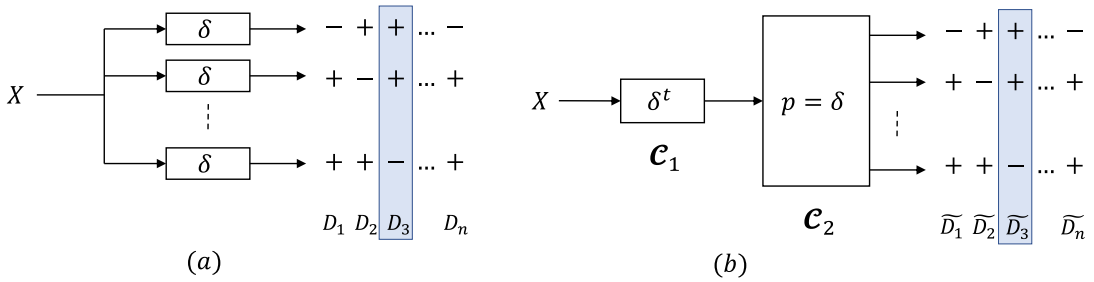

Fig. 9. The deletion error events occurring in the two channel models. Here '−' corresponds to a symbol being deleted and '+' corresponds to a transmission. The deletion pattern D_i corresponds to the input symbol X_i.

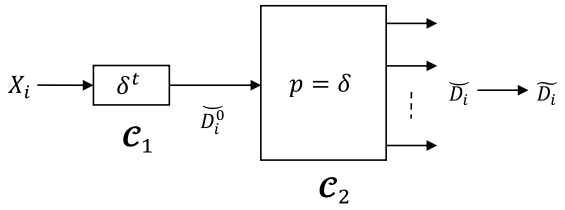

Fig. 10. The error events of the cascade model, expressed in terms of the error events of its components.

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

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

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,

$$\begin{aligned}
&\sum_{\substack{\mathcal{S}|\mathcal{S}\subseteq[n],\\ |\mathcal{S}|=m,\\ i\in\mathcal{S}}} \prod_{j=1}^{m} p_{\mathcal{S}_j}^{y_j}(1-p_{\mathcal{S}_j})^{1-y_j} \\
&= (1-p_i) \sum_{k|\,y_k=0} \mathbf{F}(p_{[i-1]}, y_{[k-1]}) \mathbf{F}(p_{[i+1:n]}, y_{[k+1:m]}) \\
&\quad + p_i \sum_{k|\,y_k=1} \mathbf{F}(p_{[i-1]}, y_{[k-1]}) \mathbf{F}(p_{[i+1:n]}, y_{[k+1:m]}). \qquad (21)
\end{aligned}$$

Plugging (21) in (20) concludes the proof of the Lemma. □

3) Proof of Lemma 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 3.

Lemma 2: For $h, f_1, f_2, \ldots, f_m \in \mathcal{A}^*$,

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

Proof: The channel equivalence can essentially be tied to this lemma as follows: consider the two channel models in Fig. 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. 3 (a), the number of ways to obtain the traces given the input is equal to $\binom{X}{Y^1}\binom{X}{Y^2}\cdots\binom{X}{Y^t}$.
- For the cascade model in Fig. 3 (b), the number of ways to obtain the traces given the input is equal to $\sum_z \binom{X}{z}\langle \tilde{Y}^1 \uparrow \tilde{Y}^2 \uparrow \ldots \uparrow \tilde{Y}^t, z\rangle$, 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 $\Pr(\tilde{Y}^1, \tilde{Y}^2, \ldots, \tilde{Y}^t | Z)$. First, note that there can be multiple deletion patterns corresponding to outputs $\tilde{Y}^1, \tilde{Y}^2, \ldots, \tilde{Y}^t$ resulting from a given input Z. The number of such patterns is equal to $\langle \tilde{Y}^1 \uparrow \tilde{Y}^2 \uparrow \ldots \uparrow \tilde{Y}^t, Z\rangle$, 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 $\tilde{Y}^1, \tilde{Y}^2, \ldots, \tilde{Y}^t$: $\mathcal{D} = \{(1, S_1), (2, S_2), \ldots, (|Z|, S_{|Z|})\}$. 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,

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

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

$$\begin{aligned}
&\Pr(\tilde{Y}^1, \tilde{Y}^2, \ldots, \tilde{Y}^t | X) = \sum_z \Pr(\tilde{Y}^1, \tilde{Y}^2, \ldots, \tilde{Y}^t, Z = z | X) \\
&= \sum_z \Pr(Z = z | X) \Pr(\tilde{Y}^1, \tilde{Y}^2, \ldots, \tilde{Y}^t | Z = z) \\
&= \sum_z \left[\binom{X}{z} \delta^{t(|X|-|z|)} (1-\delta^t)^{|z|} \langle \tilde{Y}^1 \uparrow \tilde{Y}^2 \uparrow \ldots \uparrow \tilde{Y}^t, z \rangle \right. \\
&\qquad \left. \frac{(1-\delta)^{\sum |\tilde{Y}^j|} \delta^{|z|t - \sum |\tilde{Y}^j|}}{(1-\delta^t)^{|z|}} \right] \\
&= \delta^{t|X| - \sum |\tilde{Y}^j|} (1-\delta)^{\sum |\tilde{Y}^j|} \sum_z \binom{X}{z} \langle \tilde{Y}^1 \uparrow \tilde{Y}^2 \uparrow \ldots \uparrow \tilde{Y}^t, z \rangle.
\end{aligned} \tag{22}$$

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

$$\begin{aligned}
\Pr(Y^1, Y^2, \ldots, Y^t | X) &= \prod_{j=1}^{t} \binom{X}{Y^j} \delta^{|X| - |Y^j|} (1-\delta)^{|Y_j|} \\
&= \binom{X}{Y^1} \binom{X}{Y^2} \cdots \binom{X}{Y^t} \delta^{t|X| - \sum |Y^j|} (1-\delta)^{\sum |Y^j|}.
\end{aligned} \tag{23}$$

Equating (22) and (23) with $X = h$ and traces as $Y^j = \tilde{Y}^j = f_j$ proves the Lemma.

Alternatively, we also use 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 [13] 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

$$\begin{aligned}
&\binom{h}{f_1} \binom{h}{f_2} \cdots \binom{h}{f_k} \binom{h}{f_{k+1}} \\
&= \sum_w \binom{h}{w} \langle f_1 \uparrow f_2 \uparrow \ldots \uparrow f_k, w \rangle \binom{h}{f_{k+1}} \\
&= \sum_w \left[\binom{h}{w} \binom{h}{f_{k+1}} \right] \langle f_1 \uparrow f_2 \uparrow \ldots \uparrow f_k, w \rangle \\
&= \sum_w \left[\sum_v \langle w \uparrow f_{k+1}, v \rangle \binom{h}{v} \right] \langle f_1 \uparrow f_2 \uparrow \ldots \uparrow f_k, w \rangle \\
&= \sum_v \binom{h}{v} \left[\sum_w \langle w \uparrow f_{k+1}, v \rangle \langle f_1 \uparrow f_2 \uparrow \ldots \uparrow f_k, w \rangle \right].
\end{aligned} \tag{24}$$

$$\tag{25}$$

To evaluate the term in the square bracket, we use (34). For the case where $\tau \in \mathcal{A}^*, \sigma \in \mathbb{Z}\langle\mathcal{A}\rangle$ in (34), we have

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

and thus

$$\langle \sigma \uparrow \tau, u \rangle = \sum_{f \in \mathcal{A}^*} \langle \sigma, f \rangle \langle f \uparrow \tau, u \rangle. \tag{26}$$

We use (26) to replace the term in the square bracket in (25), i.e.,

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

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

4) Proof of Lemma 3:

Lemma 3:

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

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

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,

$$\begin{aligned}
\sum_{\substack{f \in \mathcal{A}^n | \\ f_i = a}} \binom{f}{g} &= \sum_{\substack{f \in \mathcal{A}^n | \\ f_i = a}} \sum_{\substack{S \subseteq [n]| \\ |S| = |g|}} \mathbb{1}_{f_S = g} = \sum_{\substack{S \subseteq [n]| \\ |S| = |g|}} \sum_{\substack{f \in \mathcal{A}^n | \\ f_i = a}} \mathbb{1}_{f_S = g} \\
&= \sum_{\substack{S \subseteq [n]| \\ |S| = |g| \\ i \notin S}} \sum_{\substack{f \in \mathcal{A}^n | \\ f_i = a}} \mathbb{1}_{f_S = g} + \sum_{\substack{S \subseteq [n]| \\ |S| = |g| \\ i \in S}} \sum_{\substack{f \in \mathcal{A}^n | \\ f_i = a}} \mathbb{1}_{f_S = g} \\
&= \sum_{\substack{S \subseteq [n]| \\ |S| = |g| \\ i \notin S}} \sum_{\substack{f \in \mathcal{A}^n | \\ f_i = a}} \mathbb{1}_{f_S = g} + \sum_{j=1}^{m} \sum_{\substack{S \subseteq [n]| \\ |S| = |g| \\ S_j = i}} \sum_{\substack{f \in \mathcal{A}^n | \\ f_i = a}} \mathbb{1}_{f_S = g}.
\end{aligned} \tag{27}$$

The two terms in (27) can be visualized as the number of ways to fill up the blank spaces (spaces without arrows pointing to

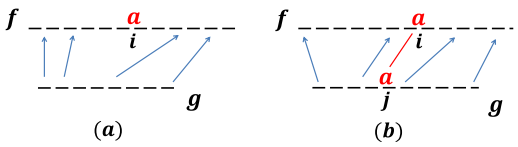

Fig. 11. Figure illustrating proof of Lemma 3.

it in f) in Fig. 11(a) and (b) respectively. Solving this counting problem, we get

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

□

B. Dynamic Program to Compute $\mathbf{F}(\cdot)$ and $\nabla\mathbf{F}(\cdot)$

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

$$\mathbf{G}^{for}(k, j) \triangleq \mathbf{F}(p_{[1:k]}, v_{[1:j]}). \tag{28}$$

Using Lemma 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 $\mathbf{G}^{for}$:

$$\mathbf{G}^{for}(k, j) = \mathbf{G}^{for}(k-1, j) + p_k^{v_j}(1-p_k)^{1-v_j}\mathbf{G}^{for}(k-1, j-1), \tag{29}$$

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. 11.

Algorithm 11 Computing $\mathbf{F}(p, v)$

Inputs: $p \in [0, 1]^n$, $v \in \{0, 1\}^m$
Outputs: $\mathbf{F}(p_{[1:k]}, v_{[1:j]})$ for all $k \in [n]$ and $j \in [m]$
Initialize $\mathbf{G}^{for}(k, 0) = 1\ \forall\ k$ and $\mathbf{G}^{for}(k, j) = 0\ \forall\ k < j$
for $k = 1 : n$ and $j = 1 : m$ **do**
    Use (29) to update $\mathbf{G}^{for}(k, j)$
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 $\mathbf{G}^{rev}$:

$$\mathbf{G}^{rev}(k, j) = \mathbf{G}^{rev}(k+1, j) + p_{k+1}^{v_{j+1}}(1-p_{k+1})^{1-v_{j+1}}\mathbf{G}^{rev}(k+1, j+1), \tag{30}$$

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) Computation of $\nabla_p\mathbf{F}(p, v)$: First, from Lemma 1, we have

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

Differentiating with respect to p_i, we get

$$\begin{aligned}\frac{\partial\mathbf{F}(p, v)}{\partial p_i} &= \sum_{k|v_k=1}\mathbf{F}(p_{[i-1]}, v_{[k-1]})\mathbf{F}(p_{[i+1:n]}, v_{[k+1:m]})\\ &\quad - \sum_{k|v_k=0}\mathbf{F}(p_{[i-1]}, v_{[k-1]})\mathbf{F}(p_{[i+1:n]}, v_{[k+1:m]})\\ &= \sum_{k|v_k=1}\mathbf{G}^{for}(i-1, k-1)\mathbf{G}^{rev}(i, k)\\ &\quad - \sum_{k|v_k=0}\mathbf{G}^{for}(i-1, k-1)\mathbf{G}^{rev}(i, k).\end{aligned} \tag{31}$$

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, $\nabla_p\mathbf{F}(p, v)$ can be computed in $O(nm)$ time and space complexity.

Algorithm 12 Computing $\nabla_p\mathbf{F}(p, v)$

Inputs: $p \in [0, 1]^n$, $v \in \{0, 1\}^m$
Outputs: $\nabla_p\mathbf{F}(p, v)$
Initialize $\mathbf{G}^{for}(k, 0) = 1\ \forall\ k$ and $\mathbf{G}^{for}(k, j) = 0\ \forall\ k < j$
Initialize $\mathbf{G}^{rev}(k, m) = 1\ \forall\ k$ and $\mathbf{G}^{rev}(k, j) = 0\ \forall\ k, j : n-k < m-j$
for $k = 1 : n$ and $j = 1 : m$ **do**
    Use (29) and (30) to compute $\mathbf{G}^{for}(k, j)$ and $\mathbf{G}^{rev}(k, j)$
for $i = 1 : n$ **do**
    Use (31) to compute $\frac{\partial\mathbf{F}(p,v)}{\partial p_i}$
return $\nabla_p\mathbf{F}(p, v)$

C. An Algebraic Definition of the Infiltration Product

For completeness, we reproduce the formal definition of the infiltration product from Section 6.3 of [13] (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.

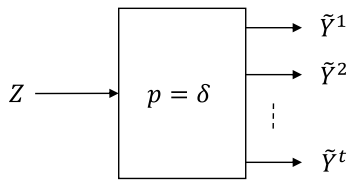

Fig. 12. The remnant channel.

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:

$$\langle \sigma \oplus \tau, w \rangle \triangleq \langle \sigma, w \rangle + \langle \tau, w \rangle \quad \forall w \in \mathcal{A}^*, \tag{32}$$

$$\langle \sigma \times \tau, w \rangle \triangleq \sum_{\substack{f,g \in \mathcal{A}^*: \\ f.g = w}} \langle \sigma, f \rangle \langle \tau, g \rangle \quad \forall w \in \mathcal{A}^*. \tag{33}$$

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 (32) and (33) are similar to the operations defined on commutative polynomials, except that the multiplication operation under the summation in (33) ($f.g = w$) is actually concatenation and is non-commutative. The multiplication inside the summation in (33) is multiplication in the real field and hence commutative. The multiplication defined in (33) distributes over addition defined in (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:

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

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

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

$$\forall \sigma, \tau \in \mathbb{Z}\langle \mathcal{A} \rangle, \quad \sigma \uparrow \tau = \sum_{f,g \in \mathcal{A}^*} \langle \sigma, f \rangle \langle \tau, g \rangle (f \uparrow g). \tag{34}$$

D. Symbolwise Posterior Probabilities for the Remnant Channel

Consider the remnant channel shown below, and let $Z = Z_1 Z_2 \dots Z_n$. Also let $Z_i \sim \text{Ber}(0.5)$. We aim to compute $\Pr(Z_i = 1 | \tilde{Y}^1 = y^1, \tilde{Y}^2 = y^2, \dots, \tilde{Y}^t = y^t)$. From the definition of the infiltration product, the input-output relation for this channel can be derived to be:

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

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

$$\begin{aligned}
&\Pr(Z_i = 1 | \tilde{Y}^1 = y^1, \tilde{Y}^2 = y^2, \dots, \tilde{Y}^t = y^t) \\
&= \sum_{\substack{z || z|=n, \\ z_i = 1}} \Pr(z | \tilde{Y}^1 = y^1, \tilde{Y}^2 = y^2, \dots, \tilde{Y}^t = y^t) \\
&= \frac{\sum_{\substack{z || z|=n, \\ z_i = 1}} \Pr(\tilde{Y}^1 = y^1, \tilde{Y}^2 = y^2, \dots, \tilde{Y}^t = y^t | z)}{2^n \Pr(\tilde{Y}^1 = y^1, \tilde{Y}^2 = y^2, \dots, \tilde{Y}^t = y^t)} \\
&= \frac{(1-\delta)^{\sum |y^j|} \delta^{nt - \sum |y^j|} \sum_{\substack{z || z|=n, \\ z_i = 1}} \langle y^1 \uparrow y^2 \uparrow \dots \uparrow y^t, z \rangle}{(1-\delta^t)^n 2^n \Pr(\tilde{Y}^1 = y^1, \tilde{Y}^2 = y^2, \dots, \tilde{Y}^t = y^t)}.
\end{aligned} \tag{35}$$

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

$$\begin{aligned}
&\Pr(Z_i = 0 | \tilde{Y}^1 = y^1, \tilde{Y}^2 = y^2, \dots, \tilde{Y}^t = y^t) \\
&= \frac{(1-\delta)^{\sum |y^j|} \delta^{nt - \sum |y^j|} \sum_{\substack{z || z|=n, \\ z_i = 0}} \langle y^1 \uparrow y^2 \uparrow \dots \uparrow y^t, z \rangle}{(1-\delta^t)^n 2^n \Pr(\tilde{Y}^1 = y^1, \tilde{Y}^2 = y^2, \dots, \tilde{Y}^t = y^t)}.
\end{aligned} \tag{36}$$

We could further simplify (35) and (36) using the fact that the expressions in (35) and (36) must sum to 1, leading us to

$$\Pr(Z_i = 1 | \tilde{Y}^1 = y^1, \tilde{Y}^2 = y^2, \dots, \tilde{Y}^t = y^t) = \frac{\sum_{\substack{z || z|=n, \\ z_i = 1}} \langle y^1 \uparrow y^2 \uparrow \dots \uparrow y^t, z \rangle}{\sum_{z || z| = n} \langle y^1 \uparrow y^2 \uparrow \dots \uparrow y^t, z \rangle}. \tag{37}$$

We precisely describe the algorithm which computes the terms in (37) in section V, 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 $\langle y^1 \uparrow y^2 \uparrow \dots \uparrow y^t, z \rangle$ is the number of paths from origin to destination of the edit graph $\mathcal{G}(y^1, y^2, \dots, y^t)$ which correspond to z. Therefore, $\sum_{z||z|=n} \langle y^1 \uparrow y^2 \uparrow \dots \uparrow y^t, z \rangle$ is equal to the number of n-length paths in $\mathcal{G}(y^1, y^2, \dots, y^t)$ 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 $\mathcal{G}(y^1, y^2, \dots, y^t)$ exactly once.

The numerator term $\sum_{z||z|=n \atop z_i=1}\langle y^1 \uparrow y^2 \uparrow \ldots \uparrow y^t, z\rangle$ can be interpreted in a similar way: it is equal to the number of n-length paths in $\mathcal{G}(y^1, y^2, \ldots, y^t)$ 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 $0, 1, \ldots, n$,
- the number of paths from v to the destination of length $0, 1, \ldots, n$.

Next we iterate over all edges in $\mathcal{G}(y^1, y^2, \ldots, y^t)$ 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 $\mathcal{G}(y^1, y^2, \ldots, y^t)$ once.

Algorithm 13 Coordinate Switch ML Heuristic

Input: Blocklength n, Trace $Y = y$, Initial point $p = (p_1, p_2, \ldots, p_n)$
Outputs: Estimated sequence $\hat{X}$
Initialize visited set $\mathcal{V} = \emptyset$
while True **do**
  Compute $\mathcal{F}_i = |\mathbf{F}(p^{(i\to 1)}, y) - \mathbf{F}(p^{(i\to 0)}, y)|\ \forall\ i$ and let $\mathcal{F} = (\mathcal{F}_1, \mathcal{F}_2, \ldots, \mathcal{F}_n)$.
  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., $\mathcal{F}_{\mathcal{S}_1} \geq \mathcal{F}_{\mathcal{S}_2} \geq \ldots \geq \mathcal{F}_{\mathcal{S}_n}$.
  for $i \in \mathcal{S}$ (ordered traversal) **do**
    if $\mathbf{F}(p^{(i\to 1)}, y) - \mathbf{F}(p^{(i\to 0)}, y) \geq 0$ **then**
      update $p \leftarrow p^{(i\to 1)}$
    else
      update $p \leftarrow p^{(i\to 0)}$
  if $p \in \mathcal{V}$ **then** break
  $\mathcal{V} = \mathcal{V} \cup \{p\}$
return $\hat{X} = p$

E. A Heuristic for ML Optimization With a Single Trace

The proof of Theorem 4 inspires a heuristic for sequence reconstruction (see Alg. 13):

- Start from a given point $p = (p_1, \ldots, p_n) \in [0,1]^n$.
- One round of iteration is defined as follows: fix a traversal order for the indices $\{1, 2, \ldots, n\}$. 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. 13 starts from an interior point ($p = (p_1, \ldots, p_n) \in [0,1]^n$ where $\exists\ p_i \in (0,1)$) as compared to starting from a lattice point (for instance, we could start from $p = (y, 0, \ldots, 0) \in \{0,1\}^n$) which is the n-length sequence 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. 13, we compare the performance of Coordinate switch heuristic with the other trace reconstruction heuristics in Section VI. We see that the coordinate switch with interior point initialization performs very similarly 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.

F. 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.,

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

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

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

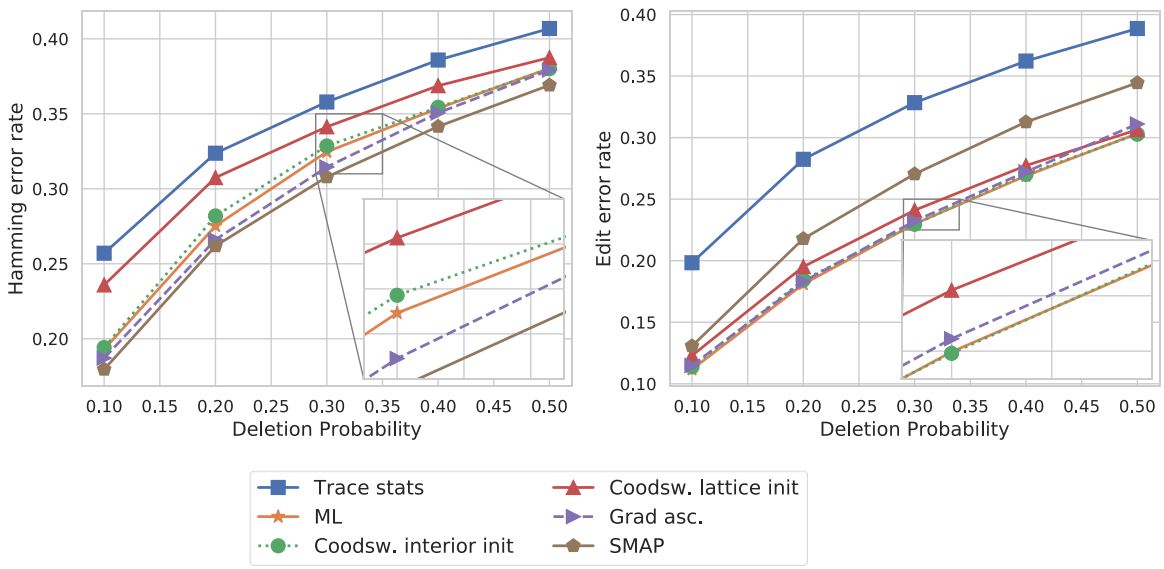

Fig. 13. Numerics for reconstruction from a single trace for a blocklength $n = 20$. This plot compares the performance of coordinate switch heuristic (abbreviated "Coodsw. interior init." and "Coodsw. lattice init.") with other trace reconstruction algorithms from Section VI. "ML" refers to the true ML sequence obtained via an exhaustive search on all 20 length binary sequences. The interior point initialization initializes $p = (0.5, 0.5, \ldots, 0.5)$ while the lattice point initialization appends the trace y with zeros to obtain an n-length vector $p = (y, 0, \ldots, 0)$.

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

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

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.

ACKNOWLEDGMENT

The authors would like to thank the anonymous reviewers and the associate editor for their numerous insightful and useful comments and suggestions.

REFERENCES

[1] S. R. Srinivasavaradhan, M. Du, S. Diggavi, and C. Fragouli, "On maximum likelihood reconstruction over multiple deletion channels," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jun. 2018, pp. 436–440.

[2] S. R. Srinivasavaradhan, M. Du, S. Diggavi, and C. Fragouli, "Symbolwise MAP for multiple deletion channels," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jul. 2019, pp. 181–185.

[3] S. R. Srinivasavaradhan, S. Diggavi, and C. Fragouli, "Equivalence of ML decoding to a continuous optimization problem," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jun. 2020, pp. 343–348.

[4] M. Mitzenmacher, "A survey of results for deletion channels and related synchronization channels," *Probab. Surv.*, vol. 6, pp. 1–33, 2009.

[5] T. Batu, S. Kannan, S. Khanna, and A. McGregor, "Reconstructing strings from random traces," in *Proc. SODA*, 2004, pp. 910–918.

[6] T. Holenstein, M. Mitzenmacher, R. Panigrahy, and U. Wieder, "Trace reconstruction with constant deletion probability and related results," in *Proc. ACM-SIAM SODA*, 2008, pp. 389–398.

[7] Y. Peres and A. Zhai, "Average-case reconstruction for the deletion channel: Subpolynomially many traces suffice," in *Proc. IEEE 58th Annu. Symp. Found. Comput. Sci. (FOCS)*, Oct. 2017, pp. 228–239.

[8] A. De, R. O'Donnell, and R. A. Servedio, "Optimal mean-based algorithms for trace reconstruction," in *Proc. 49th Annu. ACM SIGACT Symp. Theory Comput. (STOC)*, 2017, pp. 1047–1056.

[9] N. Holden, R. Pemantle, and Y. Peres, "Subpolynomial trace reconstruction for random strings and arbitrary deletion probability," in *Proc. 31st Conf. Learn. Theory*, 2018, pp. 1799–1840.

[10] F. Nazarov and Y. Peres, "Trace reconstruction with $\exp(O(n^{1/3}))$ samples," in *Proc. 49th Annu. ACM SIGACT Symp. Theory Comput. (STOC)*, 2017, pp. 1042–1046.

[11] N. Holden and R. Lyons, "Lower bounds for trace reconstruction," *Ann. Appl. Probab.*, vol. 30, no. 2, pp. 503–525, Apr. 2020.

[12] Z. Chase, "New lower bounds for trace reconstruction," 2019, *arXiv:1905.03031*. [Online]. Available: http://arxiv.org/abs/1905.03031

[13] M. Lothaire, *Combinatorics Words* (Cambridge Mathematical Library). Cambridge, U.K.: Cambridge Univ. Press, 1997.

[14] M. Lothaire and M. Lothaire, *Algebraic Combinatorics on Words*. vol. 90. Cambridge, U.K.: Cambridge Univ. Press, 2002.

[15] M. Lothaire, *Applied combinatorics on words*, vol. 105. Cambridge, U.K.: Cambridge Univ. Press, 2005.

[16] W. Mao, S. Diggavi, and S. Kannan, "Models and information-theoretic bounds for nanopore sequencing," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jun. 2017, pp. 2458–2462.

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

[18] V. I. Levenshtein, "Efficient reconstruction of sequences," *IEEE Trans. Inf. Theory*, vol. 47, no. 1, pp. 2–22, Jan. 2001.

[19] S. Diggavi, M. Mitzenmacher, and H. Pfister, "Capacity upper bounds for the deletion channel," in *Proc. IEEE Int. Symp. Inf. Theory*, Jun. 2007, pp. 1716–1720.

[20] S. Diggavi and M. Grossglauser, "On information transmission over a finite buffer channel," *IEEE Trans. Inf. Theory*, vol. 52, no. 3, pp. 1226–1237, Mar. 2006.

[21] S. N. Diggavi and M. Grossglauser, "On transmission over deletion channels," in *Proc. Annu. Allerton Conf. Commun. Control Comput.*, 2001, pp. 1–10.

[22] E. A. Ratzer, "Marker codes for channels with insertions and deletions," in *Annales des Télécommunications*. Cham, Switzerland: Springer, 2005.

[23] E. A. Ratzer and D. J. MacKay, "Codes for channels with insertions, deletions and substitutions," in *2nd Int. Symp. Turbo Codes Rel. Topics*, 2000, pp. 149–156.
[24] E. K. Thomas, V. Y. F. Tan, A. Vardy, and M. Motani, "Polar coding for the binary erasure channel with deletions," *IEEE Commun. Lett.*, vol. 21, no. 4, pp. 710–713, Apr. 2017.
[25] M. Abroshan, R. Venkataramanan, L. Dolecek, and A. G. i Fabregas, "Coding for deletion channels with multiple traces," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jul. 2019, pp. 1372–1376.
[26] M. Cheraghchi, R. Gabrys, O. Milenkovic, and J. Ribeiro, "Coded trace reconstruction," *IEEE Trans. Inf. Theory*, vol. 66, no. 10, pp. 6084–6103, Oct. 2020.
[27] J. Brakensiek, R. Li, and B. Spang, "Coded trace reconstruction in a constant number of traces," 2019, *arXiv:1908.03996*. [Online]. Available: http://arxiv.org/abs/1908.03996
[28] H. Li and R. Durbin, "Fast and accurate short read alignment with burrows-wheeler transform," *Bioinformatics*, vol. 25, no. 14, pp. 1754–1760, Jul. 2009.
[29] I. Shomorony, S. H. Kim, T. A. Courtade, and D. N. C. Tse, "Information-optimal genome assembly via sparse read-overlap graphs," *Bioinformatics*, vol. 32, no. 17, pp. i494–i502, Sep. 2016.
[30] C. Elzinga, S. Rahmann, and H. Wang, "Algorithms for subsequence combinatorics," *Theor. Comput. Sci.*, vol. 409, no. 3, pp. 394–404, Dec. 2008.
[31] B. Haeupler and M. Mitzenmacher, "Repeated deletion channels," in *Proc. IEEE Inf. Theory Workshop (ITW)*, Nov. 2014, pp. 152–156.
[32] D. Gusfield, *Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology*. New York, NY, USA: Cambridge Univ. Press, 1997.
[33] G. Nicosia and G. Oriolo, "Solving the shortest common supersequence problem," in *Operations Research Proceedings*. Berlin, Germany: Springer, 2001, pp. 77–83.
[34] G. Xu, "Global optimization of signomial geometric programming problems," *Eur. J. Oper. Res.*, vol. 233, no. 3, pp. 500–510, Mar. 2014.
[35] V. Chandrasekaran and P. Shah, "Relative entropy relaxations for signomial optimization," *SIAM J. Optim.*, vol. 26, no. 2, pp. 1147–1173, Jan. 2016.
[36] P. H. Calamai and J. J. Moré, "Projected gradient methods for linearly constrained problems," *Math. Program.*, vol. 39, no. 1, pp. 93–116, Sep. 1987.
[37] M. B. Cohen, Y. T. Lee, and Z. Song, "Solving linear programs in the current matrix multiplication time," in *Proc. 51st Annu. ACM SIGACT Symp. Theory Comput. (STOC)*, 2019, pp. 938–942.

Sundara Rajan Srinivasavaradhan (Member, IEEE) received the B.Tech. and M.Tech. degrees in electrical engineering from the Indian Institute of Technology Madras, India, in 2016. He is currently pursuing the Ph.D. degree with the Department of ECE, University of California, Los Angeles, Los Angeles, CA, USA.

He was a Research Intern with Edwards Lifesciences in the summer of 2019, and a Research Intern with Microsoft Research in the summer of 2020. His research interests include information and coding theory and probabilistic inference applied to problems motivated by medicine and biotechnology.

Michelle Du received the B.S. degree (Hons.) in electrical engineering from the University of California, Los Angeles, Los Angeles, CA, USA, in 2020. She is currently a Software Engineer with Google, Seattle. She is broadly interested in algorithm and system design and software development. She was a recipient of the Christina Huang Memorial Prize in 2020.

Suhas N. Diggavi (Fellow, IEEE) received the B.Tech. degree in electrical engineering from the Indian Institute of Technology Delhi, India, and the Ph.D. degree in electrical engineering from Stanford University, Stanford, CA, USA, in 1998.

After completing his Ph.D., he was a Principal Member Technical Staff with the Information Sciences Center, AT&T Shannon Laboratories, Florham Park, NJ, USA. After that, he was on the faculty of the School of Computer and Communication Sciences, EPFL, where he directed the Laboratory for Information and Communication Systems (LICOS). He is currently a Professor with the Department of Electrical Engineering, University of California, Los Angeles, where he directs the Information Theory and Systems Laboratory. He has eight issued patents. His research interests include information theory and its applications to several areas, including learning, security and privacy, data compression, wireless networks, cyber-physical systems, genomics, and neuroscience. He received several recognitions for his research, including 2013 IEEE Information Theory Society & Communications Society Joint Paper Award, the 2013 ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc) Best Paper Award, the 2006 IEEE Donald Fink Prize Paper Award, and the 2019 Google Faculty Research Award. He has served as a Distinguished Lecturer and also serves on Board of Governors for the IEEE Information Theory Society. He has also helped organize IEEE and ACM conferences, including serving as the Technical Program Co-Chair for 2012 IEEE Information Theory Workshop (ITW), the Technical Program Co-Chair for the 2015 IEEE International Symposium on Information Theory (ISIT), and the General Co-Chair for Mobihoc 2018. He has been an Associate Editor of the IEEE TRANSACTIONS ON INFORMATION THEORY, ACM/IEEE TRANSACTIONS ON NETWORKING, and IEEE COMMUNICATION LETTERS, and a Guest Editor of the IEEE SELECTED TOPICS IN SIGNAL PROCESSING and in the program committees of several IEEE conferences.

Christina Fragouli (Fellow, IEEE) received the B.S. degree in electrical engineering from the National Technical University of Athens, Athens, Greece, and the M.Sc. and Ph.D. degrees in electrical engineering from the University of California, Los Angeles. She worked with the Information Sciences Center, AT&T Labs, Florham Park, NJ, USA, and the National University of Athens. She also visited Bell Laboratories, Murray Hill, NJ, USA, and DIMACS, Rutgers University. From 2006 to 2015, she was an Assistant Professor and an Associate Professor with the School of Computer and Communication Sciences, EPFL, Switzerland. She is currently a Professor with the Department of Electrical and Computer Engineering, UCLA. Her research interests include network coding, algorithms for networking, and network security and privacy. She has served in several IEEE committees, and received awards for her work. She has also served as an Information Theory Society Distinguished Lecturer, and as an Associate Editor of the IEEE COMMUNICATIONS LETTERS, Elsevier *Journal on Computer Communication*, IEEE TRANSACTIONS ON COMMUNICATIONS, IEEE TRANSACTIONS ON INFORMATION THEORY, and IEEE TRANSACTIONS ON MOBILE COMMUNICATIONS.