# Symbolwise MAP for Multiple Deletion Channels

Sundara R. Srinivasavaradhan, Michelle Du, Suhas Diggavi and Christina Fragouli
University of California, Los Angeles
Email: {sundar, michelleruodu, suhas.diggavi, christina.fragouli}@ucla.edu

***Abstract*—We consider the problem of reconstructing a sequence from fixed number of deleted versions of itself (also called traces). The problem is motivated from recent developments in *de novo* DNA sequencing technologies. The main contribution of this work is to provide a polynomial time algorithm for symbolwise MAP decoding with multiple traces. The algorithm leverages a dynamic program on the *edit graph*. We also develop a heuristic with reduced time complexity using similar ideas and provide preliminary numerical evaluations.**

***Index Terms*—Deletion channels, Trace reconstruction, symbolwise MAP, Edit graph, Dynamic programming**

## I. Introduction

Given an input sequence of length $n$ (known apriori), the independently and identically distributed (i.i.d.) deletion channel deletes each input symbol independently with probability $\delta$, 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. This is a simple model that approximately captures the process of a DNA sequence passed through a nanopore sequencer[1]. We call this the $t$-deletion channel model[2]. The goal is to estimate $X$ from the *traces* $\{Y_i\}$s. A similar problem has received considerable attention in CS theory and discrete mathematics, under the name "trace-reconstruction" (see [4], [5], [6], [7], [8], [9]). Trace reconstruction, however, addresses the complementary question of how many traces are needed for *perfect* reconstruction of the input sequence. A variety of upper and lower bounds have been derived for worst case as well as average case reconstruction. In contrast, in this work we fix the number of traces (motivated by finite number of reads in DNA sequencing), and aim for a symbolwise maximum aposteriori (MAP) estimate of the sequence, which is not necessarily a perfect reconstruction. Nonetheless, the symbolwise MAP estimate is optimal for minimizing the bit error rate.

There are other works on sequence assembly (see for example, [10], [11]), where multiple short reads (from different segments of a sequence) are used to reconstruct the bigger sequence. To the best of our knowledge, these works have not looked at symbolwise MAP decoding.

Deletion channels are known to be notoriously difficult to analyze. In fact, the capacity of a single deletion channel is still unknown ([12], [13], [14], [15]); as are optimal coding schemes. Recent works have looked at the design of codes for deletion channels ([16], [17], [18]); these works consider use of a codebook (we do not), and a single channel. As a result, statistical estimation over deletion channels is also a difficult problem 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; not even for maximum likelihood over a single deletion channel (see [12]). Our prior work in [19] was a first attempt to address this question. [19] solves the symbolwise MAP decoding with one trace in polynomial time, but the case of multiple traces was left open. Note that unlike in the case of memoryless channels, for deletion channels the posterior probabilities obtained from each of the traces $\Pr(X_i|Y_j)$ cannot be combined since the Markov chain property $Y_j - X_i - Y_k$ does not hold for individual symbols $X_i$ even though $Y_j - X - Y_k$ holds. Hence, the symbolwise MAP with multiple traces is not trivial and we completely solve it here in polynomial (in $n$) time.

In terms of theoretical tools, [20] extensively uses algebraic tools for problems in the combinatorics of sequences (or *words*), and our work is partly inspired by such techniques.

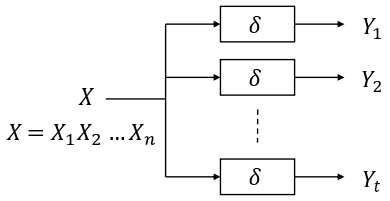


Fig. 1: $t$-deletion channel model: sequence $X$ passed through $t$ independent deletion channels. We aim to estimate $X$ from the $Y_i$s.

Our contributions are two-fold:

- In Section III, for the $t$-deletion channel model in Fig. 1, we give an $O(2^t n^{t+2})$ algorithm that exactly computes the symbolwise posterior probabilities $\Pr(X_i{=}1|Y_1, ..., Y_t)$ when $X_i \sim$ i.i.d. Ber$(0.5)$, i.e., each $n$-length binary sequence is equally likely apriori.
- Though, the symbolwise MAP is polynomial in $n$, it grows exponentially with $t$. In Section IV, we develop a symbolwise estimation heuristic which is linear in $t$ as follows: consider sequence $X$ where each symbol $X_i \sim$ ind. Ber$(p_i)$. Given a trace (say $Y$) of $X$, we provide a $O(n^2)$ algorithm to calculate $\Pr(X_i{=}1|Y)\ \forall\ i$, taking into account the prior distribution parameters $p_i$. We can then use this algorithm with one trace at a time to continually update $p_i$; we use each trace exactly once, thus making $t$ updates and estimating $X$ with $O(tn^2)$ time complexity. Note that the heuristic is not equivalent to the symbolwise MAP as $Y_j - X_i - Y_k$ does not hold.

This work was supported in part by NSF grants 1705077 and 1514531.

[1]As seen in [1],[2] there are more complicated effects of the nanopore reader not captured in this simple representation.

[2]As seen in the appendix of the longer version of this paper [3], this parallel channel model can be decomposed into a cascade of two other channels.

## II. Preliminaries

**Notation:** Calligraphic letters refer to sets and capitalized letters correspond to random variables. Let $\mathcal{A}$ be the set of all symbols. We will focus on the case where $\mathcal{A} = \{0, 1\}$, though our methods extend to larger sets. Define $\mathcal{A}^n$ to be the set of all $n$-length sequences, $\mathcal{A}^*$ to be the set of all finite length sequences with symbols in $\mathcal{A}$. For a sequence $f$, $|f|$ denotes the length of $f$. 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 $\binom{f}{g}$. 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.

**Edit graph** (as defined in [21]): We now define an *edit graph* where given two sequences $f$and $g$, every path connecting the origin to the destination on the edit graph yields a supersequence $h$ of $f, g$, where $h$ is "covered" by $f, g$ – each symbol of $h$ comes from either $f$ or $g$ or both. For $f$ and $g$ in $\mathcal{A}^*$, we form a graph $\mathcal{G}(f, g)$ with $(|f|+1)(|g|+1)$ vertices each labeled 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 $(|f|, |g|)$.

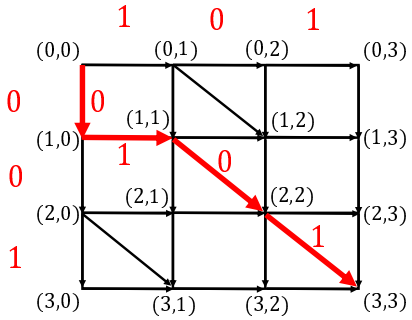


Fig. 2: Edit graph for sequences $f = 001$ and $g = 101$. Write down $f$ vertically with each symbol aligned to a vertical set of edges and $g$ horizontally likewise. A diagonal edge in a small square exists if the corresponding $f$ and $g$ symbols are equal. The thick red edges form a path from the origin to the destination; this path corresponds to the sequence '0101' – append the corresponding symbol at the left of an edge if it's vertical or diagonal, otherwise append the symbol at the top. It is also easy to verify that 0101 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.

Let $p = ((i_1, j_1), (i_2, j_2), ..., (i_m, j_m))$ be a path in $\mathcal{G}(f, g)$. We define $s(p)$ to be the sequence corresponding to the path. Intuitively, $s(p)$ is formed by appending symbols in the following way: append the corresponding $f$ symbol for a vertical edge, $g$ symbol for horizontal edge, and $f$ or $g$ symbol for diagonal edge (see example Fig. 2). Any path from $(0, 0)$ to $(|f|, |g|)$ corresponds to a supersequence of $f$ and $g$ and which is covered by $f$ and $g$. More formally, define $s(p) \triangleq x_1 x_2 ... x_{m-1}$ where

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

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

- $j_l = i_l$ or $j_l = i_l + 1 \ \forall \ l \in [t]$, i.e., $(i_1, ..., i_t)$ and $(j_1, ..., j_t)$ are vertices of a particular unit cube. Only these type of vertices can share an edge in the grid graph.
- Let $\mathcal{T} \subseteq [t]$ be the collection of indices where $j_l = i_l + 1$. Then $f_{i_l}$ is same $\forall \ l \in \mathcal{T}$. For instance, if $\mathcal{T} = \{1, 3, 5\}$, then $f_{i_1} = f_{i_3} = f_{i_5}$.

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

**Infiltration product** (introduced in [20]): The infiltration product has been extensively used in [20], as a tool in non-commutative algebra. Here we give an edit-graph interpretation of this tool. We also formally define it in the longer version of the paper [3] . Using the edit graph we can construct the set of possible supersequences $\mathcal{S}(f, g)$ of $f, g$ which are covered by it. Clearly multiple paths could yield the same supersequence and we can count the number of distinct ways $N(h; f, g)$ 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 as the supersequences $h$ in $\mathcal{S}(f, g)$ and coefficients $\langle f \uparrow g, h \rangle$ equal to $N(h; f, g)$. In Fig. 2, it is easy to verify that 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. More examples: 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 [20]. The definition of infiltration extends to two polynomials via distributivity (precisely defined in [3]), and consequently to multiple sequences as well. For multiple sequences, infiltration has the same edit graph interpretation: $\langle f_1 \uparrow f_2 \uparrow ... \uparrow f_t, w \rangle$ is the number of distinct ways of constructing $w$ as a supersequence of $f_1, f_2, ..., f_t$ so that the construction covers

$w$, i.e., construct the $t$-dimensional edit graph of $f_1, f_2, ..., f_t$ and count the number of paths corresponding to $w$.

## III. Symbolwise MAP estimate for the $t$-deletion channel model

Let $\mathcal{A} = \{0,1\}$, and for this section, assume $X \sim$ Uniform $\mathcal{A}^n$. The goal is to compute the symbolwise posterior probabilities $\Pr(X_i{=}1|Y_1, ..., Y_t)$[3] where $Y_i$ is the $i^{th}$ trace. Alg. 1 details the computation of the posterior probabilities. Using Alg. 1, the symbolwise MAP estimate is as follows: for each index $i$, compute $\Pr(X_i{=}1|Y_1...Y_t)$ and decide

$$\hat{X}_i = \begin{cases} 1, & \text{if } \Pr(X_i{=}1|Y_1...Y_t) \geq 0.5 \\ 0, & \text{else.} \end{cases}$$

Through the rest of this section, we compute $\Pr(X_i{=}1|Y_1, ..., Y_t)$ as follows: we first give an expression which sums over infiltration product terms and leverage a dynamic program on the edit graph to compute these.

### A. Step 1: An expression for $\Pr(X_i = 1|Y_1...Y_t)$

**Theorem 1.** *Assume $X \sim$ Uniform $\mathcal{A}^n$. The posterior probability of the $i^{th}$ bit given the traces can be expressed as*

$$\begin{aligned}
&\Pr(X_i = 1|Y_1...Y_t) \\
&= \Bigg[ \sum_{k=0}^{n} 2^{n-k-1} \binom{n-1}{k} \sum_{w||w|=k} \langle Y_1 \uparrow ... \uparrow Y_t, w \rangle \\
&\quad + \sum_{k=0}^{n} \sum_{j=1}^{k} 2^{n-k} \binom{i-1}{j-1} \binom{n-i}{k-j} \sum_{\substack{w||w|=k, \\ w_j=1}} \langle Y_1 \uparrow ... \uparrow Y_t, w \rangle \Bigg] \\
&\quad \Big/ \Bigg[ \sum_{k=0}^{n} 2^{n-k} \binom{n}{k} \sum_{w||w|=k} \langle Y_1 \uparrow ... \uparrow Y_t, w \rangle \Bigg]. \qquad (1)
\end{aligned}$$

Note that the summation index, $w||w|{=}k$ denotes all sequences $w$ of length $k$; this is an alternate for $w|w{\in}\mathcal{A}^k$. We follow this convention throughout the paper. We refer to [3] for the proof of Theorem 1. In short, the denominator of (1) counts the number of ways of obtaining $n$-length sequences as a supersequence of $Y_1, ..., Y_t$, and the numerator counts the number of such ways where the $i^{th}$ bit of the sequence is 1.

### B. Step 2: Dynamic program to compute terms in (1)

We compute $\sum_{w||w|=k} \langle Y_1 \uparrow ... \uparrow Y_t, w \rangle$ and $\sum_{\substack{w||w|=k, \\ w_j=1}} \langle Y_1 \uparrow ... \uparrow Y_t, w \rangle$, using a dynamic programming approach on the edit graph. Note that the number of sequences $w||w| = k$ is $O(2^k)$ so a naive evaluation is exponential in $n$. We can, however, exploit the edit graph to come up with a dynamic program resulting in an algorithm which is polynomial in $n$.

Recall that in the edit graph, $\langle Y_1 \uparrow ... \uparrow Y_t, w \rangle$ is equal to the number of distinct paths from the origin $(0, ..., 0)$ to the destination $(|Y_1|, ..., |Y_t|)$ and which correspond to $w$. Thus,

(a) $\sum_{w||w|=k} \langle Y_1 \uparrow ... \uparrow Y_t, w \rangle$ is the number of distinct paths of length $k$ from origin to destination.

(b) $\sum_{\substack{w||w|=k, \\ w_j=1}} \langle Y_1 \uparrow ... \uparrow Y_t, w \rangle$ is the number of distinct paths of length $k$ from origin to destination such that the $j^{th}$ edge of the path corresponds to 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 $\lambda$) for each vertex, such that the coefficient of $\lambda^k$ 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 $\lambda^k$ as earlier is denoted by $\langle p_v^{for}(\lambda), \lambda^k \rangle$. The dynamic program to compute $p_v^{for}(\lambda)$ for all $v$:

$$p_v^{for}(\lambda) = \sum_{u|u \to v} \lambda p_u^{for}(\lambda). \qquad (2)$$

Thus, $p_{destination}^{for}(\lambda)$ has all the information needed for (a). In the example in Fig. 2, one could do the following: order the vertices $(0,0)$ to $(3,3)$ lexicographically and then compute $p_v^{for}(\lambda)$ in the same order. Because of the directed grid nature of the edit graph, every vertex has incoming neighbors which are lexicographically ahead of itself. Also we initialize $p_{(0,0)}^{for}(\lambda) = 1$. For the example in Fig. 2, the forward-potentials are shown in Fig. 3; see [3] for the description of algorithm to compute the forward potentials. 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 $O(n)$.

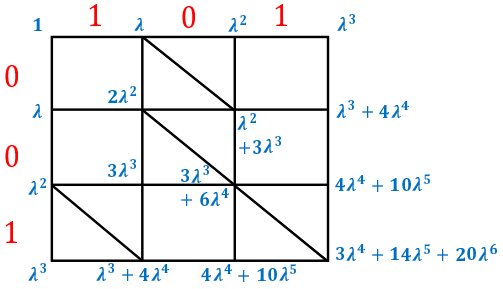


Fig. 3: The forward-potential $p_v^{for}(\lambda)$ at each vertex.

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

$$p_v^{rev}(\lambda) = \sum_{u|v \to u} \lambda p_u^{rev}(\lambda), \qquad (3)$$

[3] Symbolwise MAP with non-uniform priors is a part of on-going work.

**Algorithm 1** Computing the posterior probabilities

Input: Traces $Y_1, ..., Y_t$, index $i$
Output: $\Pr(X_i = 1|Y_1, ...Y_t)$
Construct edit graph with the traces $\mathcal{G}(Y_1, ..., Y_t)$
Compute the forward and reverse potentials $p_v^{for}(\lambda)$ and $p_v^{rev}(\lambda)\ \forall\ v \in \mathcal{G}$
**assign** $\sum_{w||w|=k} \langle Y_1\uparrow ...\uparrow Y_t, w\rangle \leftarrow \langle p^{rev}_{(0,0,..,0)}, \lambda^k\rangle\ \forall k$
**for** each $k, j$ **do**
    Initialize $temp \leftarrow 0$
    **for** each edge $u \rightarrow v\ \in\ \mathcal{G}$ **do**
        **if** $s(u{\rightarrow}v) = 1$ **then**
            $temp\ +{=} \langle p_u^{for}(\lambda), \lambda^{j-1}\rangle\langle p_v^{rev}(\lambda), \lambda^{k-j}\rangle$
    **assign** $\sum_{w||w|=k, w_j=1} \langle Y_1\uparrow ...\uparrow Y_t, w\rangle \leftarrow temp$
Use (1) to compute $\Pr(X_i = 1|Y_1, ...Y_t)$

with $p^{rev}_{destination}(\lambda) = 1$. With this,

$$\sum_{\substack{w||w|=k,\\ w_j=1}} \langle Y_1\uparrow ...\uparrow Y_t, w\rangle = \sum_{\substack{(u\rightarrow v)\\ s(u,v)=1}} \langle p_u^{for}(\lambda), \lambda^{j-1}\rangle\langle p_v^{rev}(\lambda), \lambda^{k-j}\rangle.$$

Alg. 1 details the computation of the posterior probabilities. This algorithm iterates over all the edges ($O((2n)^t)$ edges exist), and also $k, j$ which are $O(n)$ each. The time complexity of Alg. 1 hence is $O(2^t n^{t+2})$, resulting in $O(2^t n^{t+3})$ for the symbolwise MAP described at the beginning of the section.

## IV. A heuristic based on symbolwise MAP

Given that the exact symbolwise MAP proposed earlier is exponential in the number of traces $t$, we here develop a symbolwise estimation heuristic which is, in fact, linear in $t$. Let $p \triangleq (p_1, p_2, ..., p_n)$ and $X = X_1...X_n$. We assume the $X_i$'s are distributed as $X_i \sim$ ind. Ber $(p_i)$, and give a $O(n^2)$ algorithm to calculate $\Pr(X_i = 1|Y)$ given one trace $Y$. We can then use this algorithm and use one trace at a time to continually update the posterior probabilities; we use each trace exactly once, thus making $t$ updates, and finally use a thresholding criteria to decide if $X_i$ is 0 or 1. The time complexity of the algorithm is $O(tn^2)$.

### A. Step 1: Expression for $\Pr(X_i = 1|Y)$

To simplify notation, we here assume $\Pr(X_i = 1) = p_i$ and let $m \triangleq |Y|$. As in the previous section, by Bayes' rule,

$$\Pr(X_i = 1|Y) = \frac{\sum_{x|x_i=1} \Pr(X = x)\binom{x}{Y}}{\sum_x \Pr(X = x)\binom{x}{Y}}, \tag{4}$$

and as in the previous section, we are interested to simplify the summation in the numerator and the denominator. Theorem 2 gives a simplified expression for (4) in terms of a function that we define first. See longer version [3] for the proof of Theorem 2. The function denoted by $F()$ is defined as follows.

**Definition 1.**

$$F : \mathbb{R}^k \times \{0,1\}^l \rightarrow \mathbb{R},$$

$$F(q, v) \triangleq \begin{cases} \sum_{S|S\subseteq[k],|S|=l} \prod_{i=1}^{l} q_{S_i}^{v_i}(1 - q_{S_i})^{1-v_i} & 1 \le l \le k \\ 1 & 0 = l \le k \\ 0 & \text{else.} \end{cases}$$

An alternate definition is as follows: consider a random vector $Z \in \{0,1\}^k$ such that $Z_i \sim$ ind. Ber$(q_i)$, let $q$ be the vector of probabilities of length $k$, $S \subseteq [k]$ a subset of the indices of size $l$, and $v$ a binary vector of length $l$. Then,

$$F(q, v) = \sum_{S||S|=l} \Pr(Z_S = v).$$

Note that if $q$ is a vector of 0's and 1's, then $F(q, v) = \binom{q}{v}$, the binomial coefficient of $v$ in $q$. Thus, $F()$ could be interpreted as a weighted version of the binomial coefficient.

**Theorem 2.**

$$\Pr(X_i = 1|Y) = \frac{p_i F(p_{[n]\setminus\{i\}}, Y)}{F(p_{[n]}, Y)} + \frac{p_i \sum_{k|Y_k=1} F(p_{[1:i-1]}, Y_{[1:k-1]})F(p_{[i+1:n]}, Y_{[k+1,m]})}{F(p_{[n]}, Y)}$$

*where $F(\cdot,\cdot)$ is as in Definition 1.*

The intuition here is similar to that of Theorem 1 - the denominator term corresponds to the total "weight" of the ways of obtaining a $n$-length supersequence of $Y$, and the numerator corresponds to the total weight of obtaining such supersequences where the $i^{th}$ bit is 1.

### B. Step 2: Computing $F(\cdot,\cdot)$

At first sight, although $F()$ sums over all subsets of size $m$, it is still possible to efficiently compute it via a dynamic programming approach as in the previous section. We will now describe how to compute $F(p, Y)$ where $p = (p_1, ..., p_n)$ and $Y = Y_1...Y_m$, and doing so aids in computing the other terms in Theorem 2 too. We first define

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

The following dynamic program follows from the definitions:

$$G^{for}(k, j) = p_{S_j}^{Y_j}(1 - p_{S_j})^{1-Y_j} G^{for}(k-1, j-1) + G^{for}(k-1, j), \tag{5}$$

with the boundary conditions $G^{for}(k, 0) = 1\ \forall\ k$ and $G^{for}(k, j) = 0\ \forall\ k < j$. This is easily seen by splitting the summation into two cases: 1. When $k \in S$ and $k \notin S$, the first case gives the first term and the second case the second term. An equivalent argument also gives,

$$G^{rev}(k, j) = p_{S_j}^{Y_j}(1-p_{S_j})^{1-Y_j} G^{rev}(k+1, j+1) + G^{rev}(k+1, j) \tag{6}$$

with boundary conditions $G^{rev}(k, m) = 1\ \forall\ k$ and $G^{rev}(k, j) = 0\ \forall\ k, j|n - k > m - j$. The description of

**Algorithm 2** Approximate Symbolwise MAP with $t$ traces

Input: Traces $Y_1, ..., Y_t$
Outputs: Input estimate $\tilde{X}$
Initialize priors $p^{old} = p^{new} \leftarrow (0.5, 0.5, ..., 0.5)$
**for** $l = 1 : t$ **do**
    Use $p^{old}$ and $Y_l$ to compute $G^{for}$ and $G^{rev}$
    **for** $i = 1 : n$ **do**
        Use (7) to update $p_i^{new}$ with $p^{old}$ as input priors
    $p^{old} \leftarrow p^{new}$
**for** $i = 1 : n$ **do**
    **if** $p_i^{new} \geq 0.5$ **then** $\tilde{X}_i \leftarrow 1$
    **else** $\tilde{X}_i \leftarrow 0$
**return** $\tilde{X}_1\tilde{X}_2...\tilde{X}_n$

the algorithm to compute $G^{for}(k,j), G^{rev}(k,j)\ \forall\ k,j$ can be found in the [3]. The complexity of such an algorithm is $O(n^2)$ as one needs to compute $O(n^2)$ values; note that $m = O(n)$ since $Y$ is a deleted version of the input.

### C. Step 3: The overall algorithm

First note that, the following relation can be derived from the definitions of $F(), G^{for}()$ and $G^{rev}()$:

$$F(p_{[n]\setminus\{i\}}, Y) \quad = \sum_{k\in[1:i-1]} G^{for}(i-1,k)G^{rev}(i+1,k+1).$$

Thus, we can transform Theorem 2 in terms of $G()$ as

$$\Pr(X_i = 1|Y) = \frac{p_i \sum\limits_{k\leq i-1} G^{for}(i-1,k)G^{rev}(i+1,k+1)}{G^{for}(n,m)} + \frac{p_i \sum\limits_{k|Y_k=1} G^{for}(i-1,k-1)G^{rev}(i+1,k+1)}{G^{for}(n,m)}, \quad (7)$$

enabling us to compute the posterior probabilities given a trace and arbitrary priors in $O(n^2)$. The overall heuristic with $t$ traces is now detailed in Alg. 2. The complexity of the algorithm is $O(tn^2)$; $t$ loops for the traces, and each loop computes $O(n^2)$ $G^{for}$ and $G^{rev}$ terms.

## V. Numerical results

We show some preliminary numerical results for a block-length of 20 and for $2, 3, 4$ traces. We use hamming distance to measure the performance since we would like to reconstruct the (known length) entire input sequence. As seen, both the algorithms perform better as the number of traces increases, and the exact MAP outperforms the heuristic as expected. These results are for the case when each $X_i \sim$i.i.d Ber$(0.5)$. For $X_i \sim$i.i.d Ber$(p)$ $(p \neq 0.5)$, Alg.1 (which is the MAP for uniform priors) and Alg.2 give lower error rates (plots not shown here), but the estimate from Alg.1 may no longer be the symbolwise MAP estimate. The improvement in performance is expected since it is easier to "guess" each bit when the priors are not uniform. For $X_i \sim$Ber$(p_i)$, where $p_i \sim$Uniform$[0,1]$, the error rates for both algorithms approximately the same as in Fig.4.

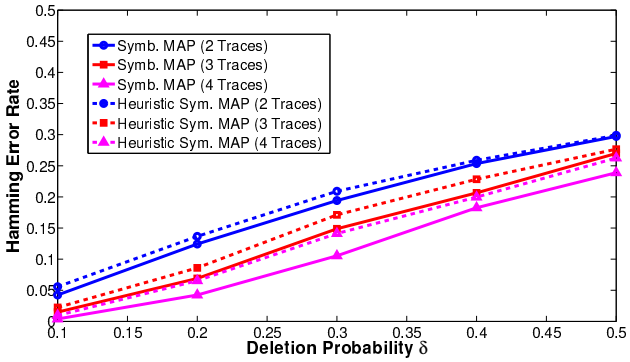


Fig. 4: Hamming error rate between the actual and estimated sequence. The exact symbolwise MAP (Alg. 1) is plotted in solid lines while the heuristic (Alg. 2) is plotted in dotted lines.

## References


[1] W. Mao, S. N. Diggavi, and S. Kannan, "Models and information-theoretic bounds for nanopore sequencing," *2017 ISIT*, 2017.
[2] ——, "Models and information-theoretic bounds for nanopore sequencing," *IEEE Transactions on Information Theory*, 2018.
[3] S. Srinivasavaradhan et al., "Symbolwise map for multiple deletion channels," Longer version. [Online]. Available: http://eeucla.com/arni/bibliography/index.php/attachments/single/147
[4] V. I. Levenshtein, "Efficient reconstruction of sequences," *IEEE Transactions on Information Theory*, Jan 2001.
[5] T. Batu, S. Kannan, S. Khanna, and A. McGregor, "Reconstructing strings from random traces," in *SODA '04*, 2004, pp. 910–918.
[6] T. Holenstein, M. Mitzenmacher, R. Panigrahy, and U. Wieder, "Trace reconstruction with constant deletion probability and related results," in *ACM-SIAM SODA '08*, 2008, pp. 389–398.
[7] A. De, R. O'Donnell, and R. A. Servedio, "Optimal mean-based algorithms for trace reconstruction," in *STOC 2017*.
[8] Y. Peres and A. Zhai, "Average-case reconstruction for the deletion channel: subpolynomially many traces suffice," *CoRR*, vol. abs/1708.00854, 2017.
[9] N. Holden, R. Pemantle, and Y. Peres, "Subpolynomial trace reconstruction for random strings and arbitrary deletion probability," in *Proceedings of the 31st Conference On Learning Theory*, 2018.
[10] H. Li and R. Durbin, "Fast and accurate short read alignment with burrows-wheeler transform," 2009.
[11] 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, 2016.
[12] M. Mitzenmacher, "A survey of results for deletion channels and related synchronization channels," *Probability Surveys*, vol. 6, pp. 1–33, 2009.
[13] S. Diggavi, M. Mitzenmacher, and H. Pfister, "Capacity upper bounds for deletion channels," in *2007 ISIT*.
[14] S. Diggavi and M. Grossglauser, "On information transmission over a finite buffer channel," *IEEE Transactions on Information Theory*, 2006.
[15] S. N. Diggavi and M. Grossglauser, "On transmission over deletion channels," in *Proceedings of the Annual Allerton Conference on Communication Control and Computing*, 2001.
[16] E. A. Ratzer, "Marker codes for channels with insertions and deletions," in *Annales des télécommunications*. Springer, 2005.
[17] E. A. Ratzer and D. J. MacKay, "Codes for channels with insertions, deletions and substitutions," in *In 2nd International Symposium on Turbo Codes and Related Topics*. Citeseer, 2000.
[18] E. K. Thomas, V. Y. Tan, A. Vardy, and M. Motani, "Polar coding for the binary erasure channel with deletions," *IEEE Communications Letters*, vol. 21, no. 4, pp. 710–713, 2017.
[19] S. R. Srinivasavaradhan, M. Du, S. Diggavi, and C. Fragouli, "On maximum likelihood reconstruction over multiple deletion channels," in *2018 IEEE International Symposium on Information Theory (ISIT)*.
[20] M. Lothaire, *Combinatorics on Words*, ser. Cambridge Mathematical Library. Cambridge University Press, 1997.
[21] D. Gusfield, *Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology*. New York, NY, USA: Cambridge University Press, 1997.