Trace Reconstruction Algorithms

Anton Sazonov, Adrian Ciotinga

April 23, 2026

Abstract

We give an overview of the major results in the field of trace reconstruction algorithms. Specifically, we examine two algorithms: the first requires exponentially many traces to reconstruct the target string but can be derived using only tools taught in this course. The second requires sub-exponentially many traces to reconstruct the target string but requires tools from complex analysis to prove its correctness.

1 Introduction

1.1 Problem Statement

Trace reconstruction is the problem of inferring some unknown underlying data object from a series of local or partial random samples. The canonical and most well-known instance of this problem focuses on reconstructing a string $\mathbf{x} = (x_1...x_n)$ given independent *traces* of $\mathbf{x}$, or instances of $\mathbf{x}$ after being passed through a randomized deletion channel.

Definition 1.1 (Trace)**.** A trace is a string $\mathbf{y} = y_1...y_k$ of length $k \leq n$ that is a subsequence of $\mathbf{x}$ where each character of $\mathbf{x}$ is absent in $\mathbf{y}$ with probability p and kept with probability $1 - p$.

Traces are not necessarily contiguous. For example, let $\mathbf{x} = 01001$, and assume the second and fourth bits are randomly chosen for deletion: then $\mathbf{y} = 001$. y_j thus denotes the index-j symbol of the trace. It is therefore not trivial to attribute a specific character in $\mathbf{y}$ to a specific index in $\mathbf{x}$, as we are not given any information on which characters were deleted. Formally, we define the problem as follows:

String trace reconstruction problem
Input: An unknown string $\mathbf{x} = x_1...x_n \in \Sigma^n$ over a finite alphabet Σ, a collection of traces $(\mathbf{y}^{(1)}, \mathbf{y}^{(2)}, ..., \mathbf{y}^{(m)})$ of $\mathbf{x}$, length n of the original string, and the fixed deletion probability p. **Output:** The underlying string $\mathbf{x}$.

In the variation of trace reconstruction covered in this report, the length of the original string n and the deletion probability p are known, and the string $\mathbf{x}$ is arbitrary. This problem asks for the number of traces needed to reconstruct $\mathbf{x}$ with a bounded error probability. Without loss of generality, we can assume $\Sigma = \{0, 1\}$.

1.2 Motivation

Trace reconstruction problems arise naturally in computational biology, coding theory, statistical inference, network analysis, imaging, and other areas. Because the core challenge involves inferring an unknown underlying data object from a series of local or partial random samples, finding efficient reconstruction algorithms has direct applications in mitigating real-world data loss:

Computational Biology Trace reconstruction is a critical component of modern DNA storage systems. The process of DNA sequencing introduces deletion errors when converting the physical DNA strand into its digital representation, so reconstruction algorithms are required to reliably recover the original data.

Coding Theory Trace reconstruction directly models the problem of transmitting information across a binary deletion channel. When a binary string is broadcast, interference may cause bits to be independently deleted or lost. By analyzing the theoretical bounds of trace reconstruction, coding theorists design error-correcting codes that allow the original message to be efficiently recovered from a minimal number of received traces.

Statistical Inference Consider an array of independent sensors attempting to record a discrete sequence of events. If each sensor is imperfect and independently misses observing an event with a fixed probability, the central system receives a collection of incomplete subsequences. Reconstructing the true sequence of events from these individually inaccurate sensors maps exactly to the trace reconstruction problem.

2 An Influence Algorithm

2.1 Overview

Notation Each trace is an instance of a random process; thus, we denote $\mathbf{Y}$ as the random variable corresponding to the trace string, and Y_j as the random variable corresponding to the j'th symbol in the trace. $\mathbf{y}$ and y_j denote instances of these random variables respectively.

We outline an approach proposed by [5] that reconstructs $\mathbf{x}$ bit by bit—that is, we consider $x_1...x_{i-1}$ to be known for a recursive argument, and show that we can infer x_i. For each step in the recursion, we use the key observation that the bits in $\mathbf{x}$ do not have equal probability of ending up as bit y_j after deletion. The authors show that a threshold S exists such that $P(Y_j = 1) > S \iff x_i = 1$, where S is computed using only bits $x_1...x_{i-1}$. We can then empirically estimate $P(Y_j = 1)$ by averaging y_j across all provided traces; if this average is $> S$ then $x_i = 1$, otherwise $x_i = 0$. An error in this approach (i.e., wrong bit "reconstructed") comes from error in the estimation of $P(Y_j = 1)$, so the complexity in this algorithm comes from the number of samples of Y_j needed bound the total error.

2.2 Algorithm

Define the influence of the bit x_i on the trace bit Y_j by:

$$\begin{aligned}\text{Inf}(i,j) &= P(x_i \text{ ends up as } Y_j \text{ in the trace}) \\ &= P(\text{exactly } j-1 \text{ of } i-1 \text{ preceding bits survive})P(x_i \text{ survives}) \\ &= \binom{i-1}{j-1}(1-p)^j(p)^{i-j}.\end{aligned}$$

Remark. Note that $\text{Inf}(i,j) > 0$ only if $i \geq j$, which gives us $P(Y_j = 1) = \sum_{i \geq j} \text{Inf}(i,j)x_i$.

We will need the following technical bounding lemma (Lemma 2.1). It is stated without proof, as the results are a consequence of algebraic manipulation of $\text{Inf}(i,j)$. We then provide the proof for a lemma that forms the basis for the entire algorithm (Lemma 2.2), which shows that we can relate the probability of a trace bit being greater than some threshold to a corresponding bit in the original string $\mathbf{x}$ taking a specific value.

Lemma 2.1. *Let $p < 1/3$.*

- *If $j \leq (1-3p)i$, then $Inf(i,j) \geq 2\sum_{k>i} Inf(k,j)$.*
- *If furthermore $j \geq (1-4p)i$, then $Inf(i,j) \geq e^{-7ip}$.*

Lemma 2.2. *Given that we know the value of $x_1...x_{i-1}$, there exists a threshold value S such that $x_i = 1 \iff P(Y_j = 1) > S$.*

Proof. This is the key result that will be used to distinguish x_i. Let $j = (1-3p)i$. We begin by breaking up the summation:

$$P(Y_j = 1) = \sum_{k=j}^{n} \text{Inf}(k,j)x_k = \sum_{k=j}^{i-1} \text{Inf}(k,j)x_k + \text{Inf}(i,j)x_i + \sum_{k=i+1}^{n} \text{Inf}(k,j)x_k.$$

Since $x_1...x_{i-1}$ is already known, the first term is directly computable; call it A. Lemma 2.1 yields $\sum_{k=i+1}^{n} \text{Inf}(k,j)x_k \in [0, \frac{1}{2}\text{Inf}(i,j)]$. Thus we see that $x_i = 1$ implies

$$A + \text{Inf}(i,j) \leq P(Y_j = 1) \leq A + \frac{3}{2}\text{Inf}(i,j),$$

and conversely $x_i = 0$ implies

$$A \leq P(Y_j = 1) \leq A + \frac{1}{2}\text{Inf}(i,j).$$

Since the lower bound in the first case is strictly greater than the upper bound in the second, we can let S be any value between them. □

For practicality, we pick S to be the midpoint between $A + \frac{1}{2}\text{Inf}(i,j)$ and $A + \text{Inf}(i,j)$. Crucially, S lies between the upper bound for $P(Y_j = 1 | x_i = 0)$ and the lower bound for $P(Y_j = 1 | x_i = 1)$. This gives us a direct correspondence between the value of x_i and whether $P(Y_j = 1) > S$: if $P(Y_j = 1) > S$ then $x_i = 1$, and if $P(Y_j = 1) < S$ then $x_i = 0$. Since we have already determined $x_1...x_{i-1}$ we can compute $A + \frac{3}{4}\text{Inf}(i,j) =$

$\sum_{k=j}^{i-1} \text{Inf}(k,j)x_k + \frac{3}{4}\text{Inf}(i,j)$ from the definition of $\text{Inf}(i,j)$ (which depends only on p, i, and j) and the values of x_k we know.

The next task is to estimate probability $P(Y_j = 1)$ from the traces. In particular, we want to ensure our prediction and the true value are "on the same side of S" (i.e., if $P(Y_j = 1) > S$ than so is our prediction) with a high probability, which is implied if the distance between them is at most $\frac{1}{4}\text{Inf}(i,j)$.

Theorem 2.3. *Given $p < 1/3$, we can reconstruct the first B bits of $\mathbf{x}$ using $O(cB\exp(Bp)\ln(\frac{2}{\varepsilon}))$ traces with error at most ε for some constant c.*

Proof. Suppose $x_1 ... x_{i-1}$ have already been determined and set $j = (1-3p)i$. Define

$$\hat{Z} := \frac{1}{m}\sum_{l\in[m]} \mathbf{1}_{\left\{Y_j^{(l)}=1\right\}}$$

to be the sum of indicators on the j-th bit of $\mathbf{Y}^{(1)}, ..., \mathbf{Y}^{(m)}$ being 1. Then $E[\hat{Z}] = P(Y_j = 1)$, since all $Y_j^{(l)}$ are identical in distribution. Write $q := P(Y_j = 1)$ and let the threshold $S = \frac{3}{4}\text{Inf}(i,j) + A$, so q is at least $\frac{1}{4}\text{Inf(i,j)}$ away from S. To guarantee that our estimate $\hat{Z}$, is "on the same side of S" as q, we must ensure that $|\hat{Z} - q| < \frac{1}{4}\text{Inf}(i,j)$. With $q = E[\hat{Z}]$, second part of Lemma 2.1, and Hoeffding's concentration inequality, we bound the probability of error:

$$\begin{aligned} P\left(|\hat{Z} - q| \geq \frac{1}{4}\text{Inf}(i,j)\right) &\leq P\left(|\hat{Z} - q| \geq \frac{1}{4}e^{-7ip}\right) \\ &\leq 2\exp(-2m(1/4e^{-7ip})^2) = 2\exp(-\frac{1}{8}me^{-14ip}). \end{aligned}$$

The first inequality comes as a consequence of the second part of Lemma 2.1, and the second line is a consequence of Hoeffding's consequence inequality since q is the expected value of $\hat{Z}$. Both ε and $P\big(|\hat{Z} - q| \geq \frac{1}{4}\text{Inf}(i,j)\big)$ represent the probability of error, so they must be equivalent. By the above inequality, $\varepsilon = P\big(|\hat{Z} - q| \geq \frac{1}{4}\text{Inf}(i,j)\big) \leq 2\exp(-\frac{1}{8}me^{-14ip})$, and solving for m yields $m \leq 8e^{14ip}\ln(2/\varepsilon)$, which gives us an upper-bound on the number of traces needed to reconstruct x_i. To reconstruct the first B bits we need to perform this procedure from $i \in \{1, ..., B\}$, so since $i \leq B$ the final bound on m becomes $m \leq 8Be^{14Bp}\ln(2/\varepsilon)$ □

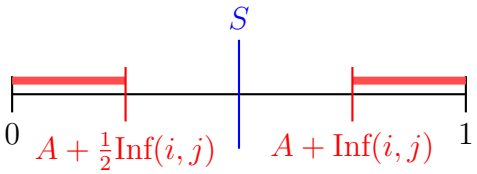

Figure 1: For the estimate $\hat{Z}$ to be accurate, it must lie on the same side of threshold S as the true probability.

The algorithm determines each bit b sequentially by calculating S (for which it needs the preceeding $b-1$ bits) and $\hat{Z}$, which is the average bit at position j in the trace, and checking whether $\hat{Z} > S$ or not. If it is then we set $x_i = 1$, otherwise $x_i = 0$.

Corollary 2.3.1. *Given $p < 1/3$ and c arbitrary, the first $c\log n$ bits of $\mathbf{x}$ can be determined with high probability given poly(c) traces.*

3 A Subexponential Algorithm

3.1 Overview

Next, we consider a more recent improvement introduced in [6] and [4] which yield a $O(\exp(n^{1/3}))$ algorithm for estimating the original string. The strategy is the following: given the list of traces $\mathbf{Y}^{(1)}, ..., \mathbf{Y}^{(m)}$, and considering two candidate strings $\mathbf{a}, \mathbf{b} \in \{0,1\}^n$, we quantify how close the average trace is to the expected random trace $\tilde{a}$ taken from the distribution of possible traces of $\mathbf{a}$ under some function f. We then do the same for $\mathbf{b}$, and we say that $\mathbf{b}$ beats $\mathbf{a}$ if its expected trace is determined to be closer than that of $\mathbf{a}$ to the average of $\mathbf{Y}^{(1)}, ..., \mathbf{Y}^{(m)}$. This allows us to define a (transitive) binary relation on $\{0,1\}^n$, and we output the string which beats every other string.

3.2 Main Results

Given a trace $\mathbf{Y}$, we define the (random) complex polynomial $f_Y(z) = \sum_{j\geq 0} Y_j z^j = Y_0 + Y_1 z + ... + Y_n z^n$. This transform will allow us to leverage complex-analytic tools for this analysis. We begin with the following result:

Lemma 3.1. *If* $\mathbf{Y}$ *is a random trace of* $\mathbf{x}$*, then* $E_Y[f_Y(z)] = (1-p)\sum_{j\geq 0} Y_j(p+(1-p)z)^j$.

We will similarly define $f_{a-b}(z)$ for two strings $\mathbf{a}, \mathbf{b}$ to be $\sum_{j\geq 0}(a_j - b_j)z^j$. Observe that this polynomial has coefficients in $\{-1, 0, 1\}$; it is known as a Littlewood polynomial. This enables the use of the following lemma, stated without proof:

Lemma 3.2 (Borwein, Erdelyi)**.** *Let* $Arc_L = \left\{e^{i\theta} \mid \theta \in [-\pi/L, \pi/L]\right\}$ *be the set of points on the arc of the complex unit circle with length* L *and let* f *be a Littlewood polynomial. Then* $\max_{z\in A_L} |f(z)| \geq \exp(-cL)$ *for some constant* c*.*

Finally, we need to define the selection metric: we will say that $\mathbf{a}$ beats $\mathbf{b}$ on list of traces $\mathbf{Y}^{(1)}, ..., \mathbf{Y}^{(m)}$ if for a specific $j = j(\mathbf{a}, \mathbf{b})$, the average $\frac{1}{m}\sum_{l\in[m]} Y_j^{(l)}$ is closer to $E_a[A_j]$ than to $E_b[B_j]$, where we take the expectations over the distributions of possible traces of $\mathbf{a}, \mathbf{b}$.

Theorem 3.3. *With probability deletion* p *and given independent traces of unknown string* $\mathbf{x}$*, two candidate strings* $\mathbf{a}, \mathbf{b} \in \{0,1\}^n$*, and error* ε*, we can determine whether* $\mathbf{a}$ *beats* $\mathbf{b}$ *from* $O(\exp(n^{1/3}))$ *samples w.p. at least* $1 - \varepsilon$*.*

Proof. Let $z \in \text{Arc}_L$ such that $|f_{a-b}(z)| \geq \exp(-cL)$, applying the Borwein-Erdelyi lemma. Note also that if $z \in \text{Arc}_L$, then $w := \frac{z-p}{1-p}$ has magnitude bounded by $O(\exp(L^{-2}))$.

$$(1-p)f_{a-b}(z) = (1-p)(f_a(z) - f_b(z)) = E[f_A(w)] - E[f_B(w)] = E\left[\sum_{j\geq 0} A_j w^j\right] - E\left[\sum_{j\geq 0} B_j w^j\right]$$

$$\exp(-cL) \leq |f_{a-b}(z)| \leq \sum_{j\geq 0} |E[A_j - B_j]| \cdot |w|^j$$

We combine this with the bound on $|w|$, noting that it is in the sum n times:

$$\sum_{j\geq 0} |E[A_j - B_j]| \geq \exp(-c_1 n L^{-2}) \exp(-c_2 L)$$

Next, let $L = n^{1/3}$, which yields:

$$\sum_{j \geq 0} |E[A_j - B_j]| \geq \exp(-cn^{1/3})$$

Then by the probabilistic method there is some j for which

$$|E[A_j] - E[B_j]| = |E[A_j - B_j]| \geq \frac{1}{n} \exp(-cn^{1/3})$$

holds; we can pick such an index arbitrarily. We will use this index j to determine whether $\mathbf{a}$ beats $\mathbf{b}$. Let $\hat{Z}_j = \frac{1}{m} \sum_{l \in [m]} Y_j^{(l)}$ and $d = \frac{1}{n} \exp(-cn^{1/3})$, where the latter is a lower bound on $|E[A_j] - E[B_j]|$. Then if $\mathbf{b}$ beats $\mathbf{a}$, we know that $E[B_j]$ is closer to $\hat{Z}_j$ than $E[A_j]$. Since $|E[A_j] - E[B_j]| \geq d$, then $|\hat{Z}_j - E[A_j]| \geq d/2$; note that if this deviation were smaller than half the deviation $|E[A_j] - E[B_j]|$, then $E[A_j]$ would be closer to $\hat{Z}_j$ and instead beat $\mathbf{b}$. Thus, we can represent the event that $\mathbf{b}$ beats $\mathbf{a}$ as $|\hat{Z}_j - E[A_j]| \geq d/2$. Now:

$$P(\mathbf{b} \text{ beats } \mathbf{a}) = P(|\hat{Z}_j - E[A_j]| \geq d/2) \leq \exp(-2md(d/2)^2/2) = \exp(-\frac{1}{2n^2} m \exp(-2cn^{1/3}))$$

by Hoeffding's inequality.

We are interested in the string which beats every other string in this manner. Since there are only finitely many strings of length n, such a string must exist due to the ordering imposed by "$\cdot$ beats $\cdot$", and be unique by the identifiability of Littlewood polynomials. Let $\hat{x}$ be this such string. Now, we apply the union bound to determine the probability that $\hat{x}$ beats every other string:

$$P(\hat{\mathbf{x}} \text{ beats all other } \mathbf{x}) \leq \sum_{\mathbf{x} \neq \hat{\mathbf{x}}} P(\hat{\mathbf{x}} \text{ beats } \mathbf{x}) \leq 2^n \exp(-\frac{1}{2n^2} m \exp(-2cn^{1/3}))$$

Setting $m \geq c' \exp(n^{1/3})$ for a large enough constant c' forces the right side to 0, implying that we can determine whether a string beats all other strings with high probability given that many traces. □

Theorem 3.3 shows that with a subexponential number of samples, we can perform a search across all potential strings to determine the string that beats every other string, giving us a reconstruction of $\mathbf{x}$.

4 Conclusion

The algorithm derived in [6] and [4] is considered state of the art for reconstruction of arbitrary strings, though there has been an improvement [3] in 2022 to $O(\exp(n^{1/5}))$.

Various relaxations exist, and for many of them polynomial time algorithms have been developed. In particular, [5] additionally develops such an algorithm for random strings. [1] developed a polylog-time algorithm for strings which are not random, but any two ones are separated by $\Omega(\text{polylog}(n))$ zeroes. [2] considers a variation where the string undergoes circular shift operations, in addition to deletions. Generalizations to non-string discrete structures such as trees and graphs exists, though for many such objects analysis is much harder.

References

[1] Anders Aamand, Allen Liu, and Shyam Narayanan. "Near-Optimal Trace Reconstruction for Mildly Separated Strings". en. In: vol. 334. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2025, 3:1–3:20. DOI: 10.4230/LIPICS.ICALP.2025.3. URL: https://drops.dagstuhl.de/entities/document/10.4230/LIPIcs.ICALP.2025.3.

[2] Arnav Burudgunte, Paul Valiant, and Hongao Wang. "New Bounds for Circular Trace Reconstruction". en. In: vol. 362. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2026, 30:1–30:23. DOI: 10.4230/LIPICS.ITCS.2026.30. URL: https://drops.dagstuhl.de/entities/document/10.4230/LIPIcs.ITCS.2026.30.

[3] Zachary Chase. *New upper bounds for trace reconstruction*. 2020. eprint: arXiv:2009.03296.

[4] Anindya De, Ryan O'Donnell, and Rocco Servedio. *Optimal mean-based algorithms for trace reconstruction*. 2016. eprint: arXiv:1612.03148.

[5] Thomas Holenstein et al. "Trace reconstruction with constant deletion probability and related results". In: *Proceedings of the Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms*. SODA '08. San Francisco, California: Society for Industrial and Applied Mathematics, 2008, pp. 389–398.

[6] Fedor Nazarov and Yuval Peres. "Trace reconstruction with exp(O(n 1/3)) samples". In: *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*. STOC '17. ACM, June 2017, pp. 1042–1046. DOI: 10.1145/3055399.3055494. URL: http://dx.doi.org/10.1145/3055399.3055494.