Approximate Trace Reconstruction via Median String (In Average-Case)

Diptarka Chakraborty
National University of Singapore, Singapore

Debarati Das
Basic Algorithm Research Copenhagen (BARC), University of Copenhagen, Denmark

Robert Krauthgamer
Weizmann Institute of Science, Rehovot, Israel

Abstract

We consider an *approximate* version of the trace reconstruction problem, where the goal is to recover an unknown string $s \in \{0,1\}^n$ from m traces (each trace is generated independently by passing s through a probabilistic insertion-deletion channel with rate p). We present a deterministic near-linear time algorithm for the average-case model, where s is random, that uses only *three* traces. It runs in near-linear time $\tilde{O}(n)$ and with high probability reports a string within edit distance $\tilde{O}(p^2 n)$ from s, which significantly improves over the straightforward bound of $O(pn)$.

Technically, our algorithm computes a $(1+\epsilon)$-approximate median of the three input traces. To prove its correctness, our probabilistic analysis shows that an approximate median is indeed close to the unknown s. To achieve a near-linear time bound, we have to bypass the well-known dynamic programming algorithm that computes an optimal median in time $O(n^3)$.

2012 ACM Subject Classification Theory of computation → Approximation algorithms analysis

Keywords and phrases Trace Reconstruction, Approximation Algorithms, Edit Distance, String Median

Digital Object Identifier 10.4230/LIPIcs.FSTTCS.2021.11

Related Version *Full Version*: `https://arxiv.org/abs/2107.09497`

Funding *Diptarka Chakraborty*: Work partially supported by NUS ODPRT Grant, WBS No. R-252-000-A94-133.
Robert Krauthgamer: Work partially supported by ONR Award N00014-18-1-2364, the Israel Science Foundation grant #1086/18, and a Minerva Foundation grant, and by the Israeli Council for Higher Education (CHE) via the Weizmann Data Science Research Center.

1 Introduction

Trace Reconstruction. One of the most common problems in statistics is to estimate an unknown parameter from a set of noisy observations (or samples). The main objectives are (1) to use as few samples as possible, (2) to minimize the estimation error, and (3) to design an efficient estimation algorithm. One such parameter-estimation problem is *trace reconstruction*, where the unknown quantity is a string $s \in \Sigma^n$, and the observations are independent *traces*, where a trace is a string that results from s passing through some noise channel. The goal is to reconstruct s using a few traces. (Unless otherwise specified, in this paper we consider $\Sigma = \{0,1\}$.) Various noise channels have been considered so far. The most basic one only performs substitutions. A more challenging channel performs deletions. Even more challenging is the *insertion-deletion* channel, which scans the string s and keeps the next character with probability $1-p$, deletes it with probability $p/2$, or inserts a uniformly

randomly chosen symbol (without processing the next character) with probability $p/2$, for some noise-rate parameter $p \in [0, 1)$. We denote this insertion-deletion channel by $R_p(s)$, see Section 2 for a formal definition.[1]

The literature studies mostly two variants of trace reconstruction. In the *worst-case* variant, the unknown string s is an arbitrary string from Σ^n, while in the *average-case* variant, s is assumed to be drawn uniformly at random from Σ^n. The trace reconstruction problem finds numerous applications in computational biology, DNA storage systems, coding theory, etc. Starting from early 1970s [30], various other versions of this problem have been studied, including combinatorial channels [37, 38], smoothed complexity [12], coded trace reconstruction [13], and population recovery [3, 44].

We focus on the average-case variant, where it is known that $\exp(O(\log^{1/3} n))$ samples suffice to reconstruct a (random) unknown string s over the insertion-deletion channel [28]. On the other hand, a recent result [9] showed that $\tilde{\Omega}(\log^{5/2} n)$ samples are necessary, improving upon the previous best lower bound of $\tilde{\Omega}(\log^{9/4} n)$ [27]. We emphasize that all these upper and lower bounds are for *exact* trace reconstruction, i.e., for recovering the unknown string x perfectly (with no errors). A natural question proposed by Mitzenmacher [43] is whether such a lower bound on the sample complexity can be bypassed by allowing approximation, i.e., by finding a string z that is "close" to the unknown string s. One of the most fundamental measures of closeness between a pair of strings z and z', is their *edit distance*, denoted by $\mathrm{ED}(z, z')$ and defined as the minimum number of insertion, deletion, and substitution operations needed to transform z into z'. Observe that a trace generated from s via an insertion-deletion channel G_p has expected edit distance about pn from the unknown string s (see Section 3). We ask how many traces (or samples) are required to construct a string z at a much smaller edit distance from the unknown s. (Since the insertion-deletion channel has no substitutions, we also do not consider substitutions in our analysis of the edit distance.)

A practical application of average-case trace reconstruction is in the portable DNA-based data storage system. In the DNA storage system [23, 51], a file is preprocessed by encoding it into a DNA sequence. This encoded sequence is randomized using a pseudo-random sequence, and thus the final encoding sequence could be treated as a (pseudo-)random string. The stored (encoded) data is retrieved using next-generation sequencing (like single-molecule real-time sequencing (SMRT) [52] that involves $12 - 18\%$, which generates several noisy copies (traces) of the stored data via some insertion-deletion channel. The final step is to decode back the stored data with as few traces as possible. Currently, researchers use multiple sequence alignment algorithms to reconstruct the trace [55, 47]. Unfortunately, such heuristic algorithms are notoriously difficult to analyze rigorously to show a theoretical guarantee. However, the preprocessing step also involves error-correcting code to encode the strings. Thus it suffices to reconstruct the original string up to some small error (depending on the error-correcting codes used). This specific application gives one motivation to study approximate trace reconstruction.

Our main contribution is to show that it is sufficient to use only three traces to reconstruct the unknown string up to a small edit error.

[1] In the literature, an insertion-deletion channel with different probabilities for insertion and for deletion has been studied. For simplicity in exposition, we consider a single error probability throughout this paper, however our results can easily be generalized to different insertion and deletion probabilities. Another possible generalization is to allow substitutions along with insertions and deletions. Again, for simplicity, we do not consider substitutions, but with slightly more careful analysis our results could be extended.

▶ **Theorem 1.** *There is a constant $c_0 > 0$ and a deterministic algorithm that, given as input a noise parameter $p \in (0, c_0]$, and three traces from the insertion-deletion channel $R_p(s)$ for a uniformly random (but unknown) string $s \in \{0,1\}^n$, outputs in time $\tilde{O}(n)$ a string z that satisfies* $\Pr[\mathrm{ED}(s,z) \le O(p^2 \log(1/p) n)] \ge 1 - n^{-1}$.

The probability in this theorem is over the random choice of s and the randomness of the insertion-deletion channel R_p. We note that the term $\log(1/p)$ in the estimation error $\mathrm{ED}(s,z)$ can be shaved by increasing the alphabet size to $\mathrm{poly}(1/\epsilon)$. An edit error of $O(p^2 n)$ is optimal for three traces, because in expectation $O(p^2 n)$ characters of s are deleted in two of the three traces, and look as if they are inserted to s in one of the three traces (which occurs in expectation for even more characters).

Our theorem demonstrates that the number of required traces exhibits a sharp contrast between exact and approximate trace reconstruction. In fact, approximate reconstruction not only beats the $\Omega(\log^{5/2} n)$ lower bound for exact reconstruction, but surprisingly uses only *three* traces! We conjecture that the estimation error $\mathrm{ED}(s,z)$ can be reduced further using more than three traces. We believe that our technique can be useful here, but this is left open for future work.

▶ **Conjecture 2.** *The estimation error $\mathrm{ED}(s,z)$ in Theorem 1 can be reduced to $O(\epsilon p n)$ for arbitrarily small $\epsilon > 0$, using $\mathrm{poly}(1/\epsilon)$ traces.*

This conjecture holds for $\epsilon < 1/n$, as follows from known bounds for exact reconstruction [28], and perhaps suggests that a number of traces that is sub-polynomial in $1/\epsilon$ it suffices for all $\epsilon > 0$.

Median String. As mentioned earlier, a common heuristic to solve the trace reconstruction problem is multiple sequence alignment, which can be formulated equivalently (see [25] and the references therein) as the problem of finding a median under edit distance. For general context, the median problem is a classical aggregation task in data analysis; its input is a set S of points in a metric space relevant to the intended application, and the goal is to find a point (not necessarily from S) with the minimum sum of distances to points in S, i.e.,

$$\min_{y} \sum_{x \in S} d(y,x). \tag{1}$$

Such a point is called a *median* (or *geometric median* in a Euclidean space). For many applications, it suffices to find an *approximate median*, i.e., a point in the metric with approximately minimal objective value (1) . The problem of finding an (approximate) median has been studied extensively both in theory and in applied domains, over various metric spaces, including Euclidean [15] (see references therein for an overview), Hamming (folklore), the edit metric [53, 34, 46], rankings [19, 2, 32], Jaccard distance [14], Ulam [8], and many more [21, 41, 6].

The median problem over the *edit-distance metric* is known as the *median string* problem [33], and finds numerous applications in computational biology [25, 50], DNA storage system [23, 51], speech recognition [33], and classification [39]. This problem is known to be NP-hard [18, 46] (even W[1]-hard [46]), and can be solved by standard dynamic programming [53, 34] in time $O(2^m n^m)$ when the input has $m = |S|$ strings of length n each. From the perspective of approximation algorithms, a multiplicative 2-approximation to the median is straightforward (this works in every metric space by simply reporting the best among the input strings, i.e., $y^* \in S$ that minimizes the objective). However, no polynomial-time

algorithm is known to break below 2-approximation (i.e., achieve factor $2-\delta$ for fixed $\delta > 0$) for the median string problem, despite several heuristic algorithms and results for special cases [7, 35, 20, 48, 1, 26, 42, 8].

Although the median string (or equivalently multiple sequence alignment) is a common heuristic for trace reconstruction [55, 47], to the best of our knowledge there is no definite connection known between these two problems. We show that both the problems are roughly the same in the average-case model. It is not difficult to show that any string close to the unknown string is an approximate median. To see this, we can show that for a set S of m traces of an (unknown) random string s, their optimal median objective value is at least $(1-O(\epsilon))pnm$, for any $\epsilon \in [110p\log(1/p), 1/6]$, with high probability (see the full version). On the other hand, the median objective value with respect to s itself is at most $(1+\epsilon)pnm$, for any $\epsilon > 0$, with high probability. Hence, the unknown string s is an $(1+O(\epsilon))$-approximate median of S, for any $\epsilon \in [110p\log(1/p), 1/6]$, and by the triangle inequality, every string close (in edit distance) to s is also an approximate median of S. One of the major contributions of this paper is the converse direction, showing that given a set of traces of an unknown string, any approximate median of the traces is close (in edit distance) to the unknown string. This is true even for three traces.

▶ **Theorem 3.** *For a large enough $n \in \mathbb{N}$ and a noise parameter $p \in (0, 0.001)$, let the string $s \in \{0,1\}^n$ be chosen uniformly at random, and let s_1, s_2, s_3 be three traces generated by the insertion-deletion channel $R_p(s)$. If x_{med} is a $(1+\epsilon)$-approximate median of $\{s_1, s_2, s_3\}$ for $\epsilon \in [110p\log(1/p), 1/6]$, then $\Pr[\mathrm{ED}(s, x_{med}) \leq O(\epsilon) \cdot \mathtt{OPT}] \geq 1 - n^{-3}$, where $\mathtt{OPT}$ denotes the optimal median objective value of $\{s_1, s_2, s_3\}$.*

An immediate consequence (see Corollary 14) is that for every $3 \leq m < n^{O(1)}$ traces, every $(1+\epsilon)$-approximate median x_{med} satisfies $\mathrm{ED}(s, x_{\mathrm{med}}) \leq O(\epsilon)\frac{\mathtt{OPT}}{m}$.

Thus if we could solve any of the two problems (even approximately), we get an approximate solution to the other problem. E.g., the current best (exact) trace reconstruction algorithm for the average-case [28] immediately gives us an $n^{1+o(1)}$ time algorithm to find an $(1+O(\epsilon))$-approximate median of a set of (at least $\exp(O(\log^{1/3} n))$) traces. (Note, to apply the result of [28], we need at least $\exp(O(\log^{1/3} n))$ traces.) We leverage this interplay between the two problems to design an efficient algorithm for approximate trace reconstruction. Since one can compute the (exact) median of three strings s_1, s_2, s_3 in time $O(|s_1| \cdot |s_2| \cdot |s_3|)$ [53, 34], the above theorem immediately provides us the unknown string up to some small edit error in time $O(n^3)$. We further reduce the running time to near-linear by cleverly partitioning each of the traces into $\mathrm{polylog}\, n$-size blocks and then applying the median algorithm on these blocks. Finally, we concatenate all the block-medians to get an "approximate" unknown string, leading to Theorem 1. One may further note that Theorem 1 also provides a $(1+O(\epsilon))$-approximate median for any set of traces in the average-case (again due to Theorem 3).

Taking the smallest possible ϵ in Theorem 3, we get that for three traces generated from s, with high probability $\mathrm{ED}(s, x_{\mathrm{med}}) \leq \tilde{O}(p^2 n)$. In comparison, it is not hard to see that with high probability $\mathtt{OPT}$ is bounded by roughly $3pn$. We conjecture that as the number of traces increases, the median string converges to the unknown string s. In particular, $\mathrm{ED}(s, x_{\mathrm{med}}) \leq \epsilon n$ when using $\mathrm{poly}(1/\epsilon)$ traces (instead of just three), with high probability. We hope that our technique can be extended to prove the above conjecture, but we leave it open for future work.

The main implication of this conjecture is an $\tilde{O}(n)$ time approximate trace reconstruction algorithm, for any fixed $\epsilon > 0$, as follows. It is straightforward to extend our approximate median finding algorithm (in Section 5) to more input strings. (For brevity, we present only

for three input strings.) For m strings, the running time would be $n(\log n)^{O(m)}$, and thus even for $m = \text{poly}(1/\epsilon)$ strings this running time is $n \,\text{polylog}\, n$. As a consequence, we will be able to reconstruct in $\tilde{O}(n)$ time a string z such that $\text{ED}(s, z) \leq \epsilon n$, which in particular implies Conjecture 2.

1.1 Related Work

A systematic study on the trace reconstruction problem has been started since [37, 38, 5]. However, some of its variants appeared even in the early '70s [30]. One of the main objectives here is to reduce the number of traces required, aka the sample complexity. Both the deletion only and the insertion-deletion channels have been considered so far. In the general worst-case version, the problem considers the unknown string s to be any arbitrary string from $\{0,1\}^n$. The very first result by Batu et al. [5] asserts that for small deletion probability (noise parameter) $p \leq \frac{1}{n^{1/2+\epsilon}}$, to reconstruct s considering $O(n \log n)$ samples suffice. A very recent work [11] improved the sample complexity to $\text{poly}(n)$ while allowing a deletion probability $p \leq \frac{1}{n^{1/3+\epsilon}}$. For any constant deletion probability bounded away from 1, the first subexponential (more specifically, $2^{\tilde{O}(\sqrt{n})}$) sample complexity was shown by [29], which was later improved to $2^{O(n^{1/3})}$ [45, 17], and then finally to $2^{O(n^{1/5})}$ [10].

Another natural variant that has also been widely studied is the average-case, where the unknown string s is randomly chosen from $\{0,1\}^n$. It turns out that this version is significantly simpler than the worst-case in terms of the sample complexity. For sufficiently small noise parameter ($p = o(1)$ as a function of n), efficient trace reconstruction algorithms are known [5, 31, 54]. For any constant noise parameter bounded away from 1 in case of insertion-deletion channel, the current best sample complexity is $\exp(O(\log^{1/3} n))$ [28] improving up on $\exp(O(\log^{1/2} n))$ [49]. Both of these results are built on the worst-case trace reconstruction by [45, 17]. Furthermore, the trace reconstruction algorithm of [28] runs in $n^{1+o(1)}$ time.

In the case of the lower bound, information-theoretically, it is easy to see that $\Omega(\log n)$ samples must be needed when the deletion probability is at least some constant. In the worst-case model, the best known lower bound on the sample complexity is $\tilde{\Omega}(n^{3/2})$ [9]. For the average-case, McGregor, Price, and Vorotnikova [40] showed that $\Omega(\log^2 n)$ samples are necessary to reconstruct the unknown (random) string s. This bound was further improved to $\tilde{\Omega}(\log^{9/4} n)$ by Holden and Lyons [27], and very recently to $\tilde{\Omega}(\log^{5/2} n)$ by Chase [9].

The results described above show an exponential gap between the upper bound and lower bound of the sample complexity. The natural question is, instead of reconstructing the unknown string exactly, if we allow some error in the reconstructed string, then can we reduce the sample complexity? Recently, Davies et al. [16] presented an algorithm that for a specific class of strings (considering various run-lengths or density assumptions), can compute an approximate trace with ϵn additive error under the edit distance while using only $\text{polylog}(n)$ samples. The authors also established that to approximate within the edit distance $n^{1/3-\delta}$, the number of required samples is $n^{1+3\delta/2}/\text{polylog}(n)$, for $0 < \delta < 1/3$, in the worst case. Independently, Grigorescu et al. [24] showed assuming deletion probability $p = 1/2$, there exist two strings within edit distance 4 such that any *mean-based algorithm* requires $\exp(\Omega(\log^2 n))$ samples to distinguish them.

1.2 Technical Overview

The key contribution of this paper is a linear-time approximate trace reconstruction algorithm that uses only three traces to reconstruct an unknown (random) string up to some small edit error (Theorem 1). To get our result, we establish a relation between the (approximate)

trace reconstruction problem and the (approximate) median string problem. Consider a uniformly random (unknown) string $s \in \Sigma^n$. We show that for any three traces of s generated by the probabilistic insertion-deletion channel R_p, an arbitrary $(1+\epsilon)$-approximate median of the three trace must be, with high probability, $O(\epsilon)\texttt{OPT}$-close in edit distance to s (Theorem 3). Once we establish this connection, it suffices to solve the median problem (even approximately). The median of three traces can be solved optimally in $O(n^3)$ time using a standard dynamic programming algorithm [53, 34]. It is not difficult to show that the optimal median objective value $\texttt{OPT}$ is at least $3(1-O(\epsilon))pn$ (see the full version), and thus the computed median is at edit distance at most $O(\epsilon pn)$ from the unknown string s. This result already beats the known lower bound for *exact* trace reconstruction in terms of sample complexity. However, the running time is cubic in n, whereas the current best average-case trace reconstruction algorithm runs in time $n^{1+o(1)}$ [28].

Next we briefly describe the algorithm that improves the running time to $\tilde{O}(n)$. Instead of finding a median of the entire traces, we compute the median block-by-block and then concatenate the resulting blocks. A natural idea is that each such block is just the median of three substrings taken from the three traces, but the challenge is to identify which substring to take from each trace, particularly because s is not known. To mitigate this issue, we take the first trace s_1 and partition it into disjoint blocks of length $\Theta(\log^2 n)$ each. For each such block, we consider its middle $\log^2 n$-size sub-block as an *anchor*. We then locate for each anchor its corresponding substrings in the other two traces s_2 and s_3, using any approximate pattern matching algorithm under the edit metric (e.g. [36, 22]) to find the *best match* of the anchor inside s_2, s_3. Each anchor has a *true match* in s_2 and in s_3, i.e., the portion that the anchor generated under the noise channel. Since the anchors in s_1 are "well-separated" (by at least $\omega(\log n)$), their true matches are also far apart both in s_2, s_3. Further, exploiting the fact that s is a random string, we can argue that each anchor's best match and true match overlap almost completely, i.e., except for a small portion (see Section 5). We thus treat these best match blocks as anchors in s_2 and s_3 and partition them into blocks. From this point, the algorithm is straightforward. Just consider the first block of each of s_1, s_2, s_3 and compute their median. Then consider the second block from each trace and compute their median, and so on. Finally, concatenate all these block medians, and output the resulting string.

The claim that the best match and true match of an anchor in s_1 are the same except for a small portion, is crucial from two aspects. First, it ensures that any r-th block of s_2, s_3 contains the true match of the r-th anchor of s_1. Consequently, computing a median of these blocks reconstructs the corresponding portion of the unknown string s up to edit distance $O(\epsilon)p\log^2 n$ with high probability. Thus for "most of the blocks", we can reconstruct up to such edit distance bound. We can make the length of the non-anchor portions negligible compared to the anchors (simply because a relatively small "buffer" around each anchor suffices), and thus, we may ignore them and still ensure that the output string is $O(\epsilon pn)$-close (in edit distance) to the unknown string s. (See the proof of Lemma 24 for the details.) The second use of that crucial claim is that it helps in searching for the best match of each anchor "locally" (within a $O(\log^2 n)$-size window) in each s_j, $j \in \{2, 3\}$. As a result, we bound the running time of the pattern matching step by $\tilde{O}(n)$. The median computations are also on $\Theta(\log^2 n)$-size blocks, and thus takes total $\tilde{O}(n)$ time.

It remains to explain the key contribution, which is the connection between the (approximate) trace reconstruction and the (approximate) median string problem. Its first ingredient is that there is an "almost unique" alignment between the unknown string s and a trace of it generated by the insertion-deletion channel R_p. Next, we use this to argue about the

similarity between an approximate median and the unknown string s. Let us now briefly describe how the uniqueness (or robustness) of the alignment between a random string s and $R_p(s)$ helps us in showing the similarity between s and any approximate median of the traces. Let s_1, s_2, s_3 be three independent traces of s, generated by R_p. We can view s_1 as a uniformly random string, and s_2, s_3 are generated from s_1 by a insertion-deletion channel R_q with a higher noise rate $q \approx 2p$. (See Section 2 for the details.) Hence, any near-optimal alignment between s_1, s_2 and s_1, s_3 "agree" with the planted alignment A^q induced by R_q (denoted by $A^q_{1,2}$ and $A^q_{1,3}$ respectively). Next, we consider the alignment $A_{1,2}$ from s_1 to s_2 via s, that we get by composing the planted alignment from s_1 to s induced by R_p (actually the inverse of the alignment from s to s_1) with the planted alignment from s to s_2 induced by R_p. Similarly, consider the alignment $A_{1,3}$ from s_1 to s_3 via s. Then we take any $(1+\epsilon)$-approximate median x_{med} of $\{s_1, s_2, s_3\}$. Consider an optimal alignment between s_1, x_{med}, and x_{med}, s_2, and x_{med}, s_3. Use these three alignments to define an alignment $M_{1,2}$ from s_1 to s_2 via x_{med}, and an alignment $M_{1,3}$ from s_1 to s_3 via x_{med}. It is not hard to argue that both $A_{1,2}$ and $M_{1,2}$ are near-optimal alignments between s_1, s_2. Thus, both of them agree with the planted alignment $A^q_{1,2}$. Similarly, both $A_{1,3}$ and $M_{1,3}$ agree with the planted alignment $A^q_{1,3}$. Observe, s_2, s_3 are not independently generated from s_1 by R_q. The overlap between $A^q_{1,2}$ and $A^q_{1,3}$ essentially provides an alignment from s_1 to s. Again, using the robustness property of the planted alignment, this overlap between $A^q_{1,2}$ and $A^q_{1,3}$ agrees with the planted alignment from s_1 to s by R_p (actually the inverse of the alignment from s to s_1). On the other hand, since $M_{1,2}$ agrees with $A^q_{1,2}$ and $M_{1,3}$ agrees with $A^q_{1,3}$, there is also a huge agreement between the overlap of $M_{1,2}, M_{1,3}$ and the overlap of $A^q_{1,2}, A^q_{1,3}$. The overlap between $M_{1,2}, M_{1,3}$ is essentially the optimal alignment from s_1 to x_{med} (that we have considered before). This in turn implies that there is a huge agreement between the optimal alignment from s_1 to x_{med} and the planted alignment from s_1 to s by R_p. Hence, we can deduce that x_{med} and s are the same in most of the portions, and thus have small edit distance. We provide the detailed analysis in Section 4.

We have just seen that it suffices to show that a near-optimal alignment between a random string s and $R_p(s)$ is almost unique (or robust). We provide below an overview of this analysis (see Section 3 for details). We start by considering the random string s and a string y generated by passing s through the noise channel R_p. For sake of analysis, we can replace R_p with an *equivalent* probabilistic model G_p, that first computes a random alignment A^p between s and y, and only then fills in random characters in s and in the insertion-positions in y. This model is more convenient because it separates the two sources of randomness, for example we can condition on one (A^p) when analyzing typical behavior of the other (characters of s).

In expectation, the channel G_p generates a trace y by performing about pn random edit operations in s (planting insertions/deletions), hence the planted alignment A^p has expected cost about pn. But can these edit operations cancel each other? Can they otherwise interact, leading to the optimal edit distance being smaller? For example, suppose $s[i] = 0$. If G_p first inserts a 0 before $s[i]$ and then deletes $s[i]$, then clearly these two operations cancel each other. We show that such events are unlikely. Following this intuition, we establish our first claim, that with high probability the edit distance between s, y is large, specifically $\text{ED}(s, y) \geq (1 - 6\epsilon)pn$ for $\epsilon \geq 15p \log(1/p)$, see Lemma 7; thus, the planted alignment A^p is near-optimal. Towards proving this, we first show that a vast majority of the planted edit operations are well-separated, i.e., have $\Theta(1/p)$ positions between them. In this case, for one operation to cancel another one, the characters appearing between them in s must all be equal, which happens with a small probability because s is random.

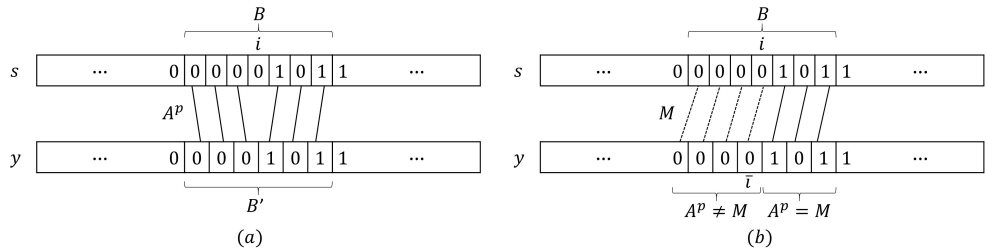

Figure 1 (a) An example of well separated edit operation: A^p deletes $s[i]$ and aligns rest of the characters in block B with block B'. (b) M aligns $s[i]$ and $y[\bar{i}]$. For each index j appearing left of i in B, $A^p[j] \neq M(j)$.

Formally, for almost all indices i where G_p performs some edit operation, the block around it $B = \{i - \frac{c}{r}, i - \frac{c}{r} + 1, \ldots, i + \frac{c}{r}\}$ in s (for a small constant $c > 0$), satisfies that i is the only index in B that G_p edits (see Lemma 6). Next we show that in every *optimal* alignment between s and y, almost all these blocks contribute a cost of 1. As otherwise, there is locally an alignment M that aligns each index in B to some character in y, while G_p makes exactly one edit operation, say deletes $s[i]$. See for example Figure 1, where M aligns $s[i]$ and $y[\bar{i}]$ whereas A^p deletes $s[i]$. In this case, M and A^p must disagree on at least c/r indices (all indices either to the right or to the left of i in B). In Figure 1, all $j \in [i - \frac{c}{r}, i]$ satisfy $M[j] \neq A^p[j]$. The crux is that any pair of symbols in s, y are chosen independently at random unless A^p aligns their positions. Thus probability that in each of the $\frac{c}{r}$ pairs aligned by M, the two matched symbols will be equal is $(1/|\Sigma|)^{\frac{c}{r}}$. In the formal proof, we address several technical issues, like having not just one but many blocks, and possible correlations due to overlaps between different pairs, which are overcome by a carefully crafted union bound.

We further need to prove that the planted alignment is robust, in the sense that, with high probability, every near-optimal alignment between s and y must "agree" with the planted alignment A^p on all but a small fraction of the edit operations. Formally, we again consider a partition of s into blocks containing exactly one planted edit operation, and show that for almost all such blocks B, if A^p maps B to a substring B' in y, that near-optimal alignment also maps B to B' (see Lemma 10). To see this, suppose there is a near-optimal alignment that maps B to $\bar{B} \neq B'$. Then following an argument similar to the above, we can show there are many indices in the block B such that A^p and M disagree on them. Thus, in each such block, M tries to match many pairs of symbols that are chosen independently at random, and therefore the probability that M matches B and $\bar{B}$ with a cost at most 1 is small. Compared to Lemma 6, an extra complication here is that now we allow M to match B and $\bar{B}$ with cost at most 1 (and not only 0), and in particular $\bar{B}$ can have three different lengths: $|B|, |B| - 1, |B| + 1$. Hence the analysis must argue separately for all these cases, requiring a few additional ideas/observations.

1.3 Preliminaries

Alignments. For two strings x, y of length n, an *alignment* is a function $A : [n] \to [n] \cup \{\perp\}$ that is monotonically increasing on the *support* of A, defined as $\mathrm{supp}(A) := A^{-1}([n])$, and also satisfies $x[i] = y[A(i)]$ for all $i \in \mathrm{supp}(A)$. An alignment is essentially a common subsequence of x, y, but provides the relevant location information. Define the *length* (or support size) of the alignment as $\mathrm{len}(A) := |\mathrm{supp}(A)|$, i.e., the number of positions in x (equivalently in y)

that are matched by A. Define the *cost of* A to be the number of positions in x and in y that are not matched by A, i.e., $\text{cost}(A) \coloneqq 2(n - \text{len}(A))$. Let $ED(x, y)$ denotes the minimum cost of an alignment between x, y.

Given a substring $x' = x[i_1, i_2]$ of x, let $\ell_1 \coloneqq \min\{k \in [i_1, i_2] \mid A(k) \neq \perp\}$ and $\ell_2 \coloneqq \max\{k \in [i_1, i_2] \mid A(k) \neq \perp\}$, be the first and last positions in the substring x' that are matched by alignment A. If the above is not well-defined, i.e., $A(k) = \perp$ for all $k \in [i_1, i_2]$, then by convention $\ell_1 = \ell_2 = 0$. Let $A(x') \coloneqq y[A(\ell_1), A(\ell_2)]$ be the *mapping* of x' under A. If $\ell_1 = \ell_2 = 0$, then by convention y' is an empty string. Let $\mathcal{U}_{x'} \coloneqq \{i_1 \leq k \leq i_2; k \notin \text{supp}(A)\}$ be the positions in x' that are not aligned by A, and similarly let $\mathcal{U}_{y'}$ be the positions in y' not aligned by A. These quantities are related because the number of matched positions in x' is the same as in y', giving us $|x'| - |\mathcal{U}_{x'}| = |y'| - |\mathcal{U}_{y'}|$. Define the *cost of alignment* A *on substring* x' to be

$$\text{cost}_A(x') \coloneqq |\mathcal{U}_{x'}| + |\mathcal{U}_{y'}|.$$

By abusing the notation, sometimes we will also use $\text{cost}_A([i_1, i_2])$ in place of $\text{cost}_A(x')$. These definitions easily extend to strings of non-equal length, and even of infinite length.

▶ **Lemma 4.** *Given two strings x, y and an alignment A, let $x_1, \ldots, x_p$ be disjoint substrings of x. Then*

1. *$A(x_1), \ldots, A(x_p)$ are disjoint substrings of y; and*
2. $\text{cost}_A(x) \geq \sum_{i \in [p]} \text{cost}_A(x_i)$

Proof. The first claim directly follows from the fact that $x_1, \ldots, x_p$ are disjoint and A is monotonically increasing. Since $x_1, \ldots, x_p$ are disjoint, also $\mathcal{U}_{A(x_1)}, \ldots, \mathcal{U}_{A(x_p)}$ are disjoint. Similarly, since $A(x_1), \ldots, A(x_p)$ are disjoint, also $\mathcal{U}_{x_1}, \ldots, \mathcal{U}_{x_p}$ are disjoint. Therefore, $\text{cost}_A(x) \geq \sum_{i \in [p]}(|\mathcal{U}_{x_1}| + |\mathcal{U}_{A(x_1)}|)$. Hence we can claim $\text{cost}_A(x) \geq \sum_{i \in [p]} \text{cost}_A(x_i)$. ◀

For an alignment $A : [n] \to [n] \cup \{\perp\}$, we define the inverse alignment $A^{-1} : [n] \to [n] \cup \{\perp\}$ as follows: For each $j \in [n]$, if $A(i) = j$ for some $i \in [n]$, set $A^{-1}(j) = i$; otherwise, set $A^{-1}(j) = \perp$.

We use the notation $\circ$ for composition of two functions. Composition of two alignments (or inverse of alignments) is defined in a natural way.

Approximate Median. Given a set $S \subseteq \Sigma^*$ and a string $y \in \Sigma^*$, we refer the quantity $\sum_{x \in S} \text{ED}(y, x)$ by the *median objective value* of S with respect to y, denoted by $\texttt{Obj}(S, y)$.

Given a set $S \subseteq \Sigma^*$, a *median* of S is a string $y^* \in \Sigma^*$ (not necessarily from S) such that $\texttt{Obj}(S, y^*)$ is minimized, i.e., $y^* = \arg\min_{y \in \Sigma^*} \texttt{Obj}(S, y)$. We refer $\texttt{Obj}(S, y^*)$ by $\texttt{OPT}(S)$. Whenever it will be clear from the context, for brevity we will drop S from both $\texttt{Obj}(S, y)$ and $\texttt{OPT}(S)$. We call a string $\tilde{y}$ a *c-approximate median*, for some $c > 0$, of S iff $\texttt{Obj}(S, \tilde{y}) \leq c \cdot \texttt{OPT}(S)$.

2 Probabilistic Generative Model

Let us first introduce a probabilistic generative model. For simplicity, our model is defined using infinite-length strings, but our algorithmic analysis will consider only a finite prefix of each string. Fixing a finite alphabet Σ, we denote by $\Sigma^{\mathbb{N}}$ the set of all infinite-length strings over Σ. We write $x \odot y$ to denote the concatenation of two finite-length strings x and y.

We actually describe two probabilistic models that are equivalent. The first model R_p is just the insertion-deletion channel mentioned in Section 1. These models are given an arbitrary string x (base string) to generate a random string y (a trace), but in our intended

application x is usually a random string. The second model G_p consists of two stages, first "planting" an alignment between two strings, and only then placing random symbols (accordingly). This is more convenient in the analysis, because we often want to condition on the planted alignment and rely on the randomness in choosing symbols. We provide the formal description of the model R_p and G_p along with a few key properties of them in Appendix A.

3 Robustness of the Insertion-Deletion Channel

In this section we analyze the finite-length version of our probabilistic model G_p (from Section 2), which gets a random string $x \in \Sigma^n$ and generates from it a trace y. We provide a high-probability estimates for the cost of the planted alignment A^p between x and y (Lemma 5), and for the optimal alignment (i.e., edit distance) between the two strings (Lemmas 7 and 8). It follows that with high probability the planted alignment A^p is near-optimal. We then further prove that the planted alignment is robust, in the sense that, with high probability, every near-optimal alignment between the two strings must "agree" with the planted alignment A^p on all but a small fraction of the edit operations (Lemmas 10 and 11).

Assume henceforth that $x \in \Sigma^n$ is a random string x, and given a parameter $p > 0$, generate from it a string y by the random process G_p described in Section 2, denoting by $A^p_{x,y}$ the random mapping used in this process. A small difference here is that now x has finite length, but it can also be viewed as an n-length prefix of an infinite string. Similarly, now y has finite length and is obtained by applying G_p on $x[1,n]$, and it can be viewed also as a finite prefix of an infinite string.

Let $\mathcal{I}^{A^p_{x,y}}$ be the set of indices $i \in [n]$, for which process G_p performs at least one insert/delete operation after (not including) $x[i-1]$ and up to (including) $x[i]$ (i.e., inserting at least one character between $x[i-1], x[i]$, or deleting $x[i]$, or both). This information can clearly be described using $A^p_{x,y}$ alone (independently of x and of the symbols chosen for insertions to y in the second stage of process G_p); we omit the formal definition. When clear from the context, we shorten $\mathcal{I}^{A^p_{x,y}}$ to $\mathcal{I}$.

▶ **Lemma 5.** *For every $i \in [n]$, the probability that $i \in \mathcal{I}$ is*

$$r = r(p) := \sum_{k=1}^{\infty} (2-p)(p/2)^k = p. \tag{2}$$

For all $\epsilon \in [r, 1]$, we have $\Pr[|\mathcal{I}| \notin (1 \pm \epsilon)rn] \leq 2e^{-\epsilon^2 rn/3}$.

Proof. For every $i \in [n]$, the probability that G_p performs $k \geq 1$ edit operations after $x[i-1]$ and up to $x[i]$ is $(p/2)^k + (p/2)^k(1-p) = (p/2)^k(2-p)$, where the first summand represents $k-1$ insertions and one deletion, and the second summand represents k insertions and no deletion. Thus $r = \Pr[i \in \mathcal{I}] = \sum_{k=1}^{\infty}(2-p)(p/2)^k = p$.

For every $i \in [n]$, the probability it appears in $\mathcal{I}$ is r. Then $\mathbb{E}[|\mathcal{I}|] = r \cdot n$. These events are independent, hence by Chernoff's bound, the probability that $|\mathcal{I}| \notin (1 \pm \epsilon)rn$ is at most $2e^{-\epsilon^2 rn/3}$. ◀

As shown in (2), $r := \sum_{k=1}^{\infty}(2-p)(p/2)^k = p$. Hence from now on we replace r by p. Given $\epsilon \in [p, 1]$ and $i \in [n]$, we consider the event $\mathcal{S}_\epsilon(i)$, which informally means that process G_p makes a single "well-spaced" edit operation at position i, i.e., there is an edit operation at position i and no other edit operations within ($\frac{2\epsilon}{p}$ positions away from i. To define it formally, we separate it into two cases, an insertion and a deletion. Observe that these events depend on A^p alone. Let $\mathcal{S}^{del}_\epsilon(i)$ be the event that

1. $A^p(i+1) = A^p(i-1)+1$ (thus $A^p(i) = \perp$); and
2. for all $j \in [2, \frac{2\epsilon}{p}]$, we have $A^p(i+j) = A^p(i+j-1)+1$ and $A^p(i-j) = A^p(i-j+1)-1$ (in particular, they are not $\perp$).

Similarly, let $\mathcal{S}^{ins}_\epsilon(i)$ be the event that

1. $A^p(i) = A^p(i-1)+2$ (thus no index is mapped to $A^p(i-1)+1$);
2. for all $j \in [1, \frac{2\epsilon}{p}]$, we have $A^p(i+j) = A^p(i+j-1)+1$; and
3. for all $j \in [2, \frac{2\epsilon}{p}]$, and $A^p(i-j) = A^p(i-j+1)-1$.

Now define the set of indices for which any of these two events happens

$$\tilde{\mathcal{I}}_\epsilon^{A^p_{x,y}} \coloneqq \{i \in [n] \mid \text{event } \mathcal{S}_\epsilon(i) \coloneqq \mathcal{S}^{del}_\epsilon(i) \cup \mathcal{S}^{ins}_\epsilon(i) \text{ occurs}\}.$$

When clear from the context, we shorten $\tilde{\mathcal{I}}_\epsilon^{A^p_{x,y}}$ to $\tilde{\mathcal{I}}$.

▶ **Lemma 6.** *For every $\epsilon \geq p$, we have* $\Pr[|\tilde{\mathcal{I}}| \leq (1-5\epsilon)pn] \leq e^{-\epsilon^2 p^2 n/2}$.

Proof. For an index $i \in [n]$, define the random variable $X_i \in \{0,1\}$ to be an indicator for the union event $\mathcal{S}^{del}_\epsilon(i) \cup \mathcal{S}^{ins}_\epsilon(i)$. Observe that each of the two events, $\mathcal{S}^{del}_\epsilon(i)$ and $\mathcal{S}^{ins}_\epsilon(i)$, occurs with probability $\frac{p}{2}(1-p)^{4\epsilon/p}/(1-p/2)$, and these events are disjoint. Hence, $\Pr[X_i = 1] = p(1-p)^{4\epsilon/p}/(1-p/2) \geq p(1-p)^{4\epsilon/p} \geq p(1-4\epsilon)$

Next we prove a deviation bound for the random variable $X \coloneqq \sum_{i\in[n]} X_i = |\tilde{\mathcal{I}}|$, which has expectation is $\mathbb{E}[X] \geq (1-4\epsilon)pn$. Observe that $\{X|X_1,\ldots,X_i\}_{i=1}^n$ is a Doob Martingale, and let us apply the method of bounded differences. Revealing X_i (after $X_1,\ldots,X_{i-1}$ are already known) might affect the value of X_j's for $j < i + \frac{4\epsilon}{p}$, but by definition their sum is bounded $\sum_{i\leq j<i+\frac{4\epsilon}{p}} X_j \leq 2$, while the other X_j's are independent of X_i hence the expectation of $\sum_{j\geq i+\frac{2\epsilon}{p}} X_j$ by revealing it. Together, we see that $|\mathbb{E}[X|X_1,\ldots,X_i] - \mathbb{E}[X|X_1,\ldots,X_{i-1}]| \leq 2$, and therefore by Azuma's inequality, $\Pr[X \leq (1-5\epsilon)pn] \leq \Pr[X \leq \mathbb{E}[X] - \epsilon pn] < e^{-2\epsilon^2 p^2 n^2/(4n)} = e^{-\epsilon^2 p^2 n/2}$. ◀

Edit Distance (Optimal Alignment) between x, y. For each $i \in \tilde{\mathcal{I}}$, define a window $W^i_\epsilon = [i - \frac{\epsilon}{p}, i + \frac{\epsilon}{p}]$.

▶ **Lemma 7.** *For every $\epsilon \in [15p\log\frac{1}{p}, \frac{1}{6}]$, we have* $\Pr[\mathrm{ED}(x,y) < (1-6\epsilon)pn] \leq 2e^{-\epsilon^2 p^2 n/2}$.

At a high level, our proof avoids a direct union bound over all low-cost potential alignments, because there are too many of them. Instead, we introduce a smaller set of basic events that "covers" all these potential alignments, which is equivalent to carefully grouping the potential alignments to get a more "efficient" union bound.

Proof of Lemma 7. We assume henceforth that A^p (the alignment from process G_p) is known and satisfies $|\tilde{\mathcal{I}}| > (1-5\epsilon)pn$, which occurs with high probability by Lemma 6. In other words, we condition on A^p and proceed with a probabilistic analysis based only on the randomness of x and of the characters inserted into y.

Our plan is to define basic events $\mathcal{E}_{S,\bar{S}}$ for every two subsets $S, \bar{S} \subset [n]$ of the same size $\ell = |S| = |\bar{S}|$, representing positions in x and in y, respectively. We will then show that our event of interest is bounded by these events

$$\Big\{\mathrm{ED}(x,y) < (1-6\epsilon)pn\Big\} \subseteq \bigcup_{S,\bar{S}|\ell=\epsilon pn} \mathcal{E}_{S,\bar{S}}, \tag{3}$$

and bound the probability of each basic event by

$$\Pr[\mathcal{E}_{S,\bar{S}}] \leq |\Sigma|^{-\epsilon\ell/(3p)}. \tag{4}$$

The proof will then follow easily using a union bound and a simple calculation.

To define the basic event $\mathcal{E}_{S,\bar{S}}$, we need some notation. Write $S = \{i_1, i_2, \ldots, i_\ell\}$ in increasing order, and similarly $\bar{S} = \{\bar{i}_1, \bar{i}_2, \ldots, \bar{i}_\ell\}$, and use these to define ℓ blocks in x and in y, namely, $B_{i_j} = x[i_j - \frac{\epsilon}{p}, i_j + \frac{\epsilon}{p}]$ and $\bar{B}_{i_j} = y[\bar{i}_j - \frac{\epsilon}{p}, \bar{i}_j + \frac{\epsilon}{p}]$. Notice that all the blocks are of the same length $1 + 2\frac{\epsilon}{p}$. Now define $\mathcal{E}_{S,\bar{S}}$ to be the event that (i) $S \subseteq \tilde{\mathcal{I}}$;[2] (ii) the blocks $\bar{B}_{\bar{i}_1}, \ldots, \bar{B}_{\bar{i}_\ell}$ in y are disjoint; and (iii) each block B_{i_j} in x is equal to its corresponding block $\bar{B}_{\bar{i}_j}$ in y. Notice that conditions (i) and (ii) actually depend only on A^p, and thus can be viewed as restrictions on the choice of $S, \bar{S}$ in (3); with this viewpoint in mind, we can simply write

$$\mathcal{E}_{S,\bar{S}} \coloneqq \{B_{i_1} = \bar{B}_{\bar{i}_1}, \ldots, B_{i_\ell} = \bar{B}_{\bar{i}_\ell}\}.$$

We proceed to prove (3). Suppose there is an alignment M from x to y with $\mathrm{cost}(M) < (1 - 6\epsilon)pn$, and consider its cost around each position $i \in \tilde{\mathcal{I}}$, namely, $\mathrm{cost}_M[i - \frac{\epsilon}{p}, i + \frac{\epsilon}{p}]$. These intervals in x are disjoint (by definition of $\tilde{\mathcal{I}}$), and thus by Lemma 4,

$$\sum_{i \in \tilde{\mathcal{I}}} \mathrm{cost}_M(x[i - \tfrac{\epsilon}{p}, i + \tfrac{\epsilon}{p}]) \leq \mathrm{cost}(M) < (1 - 6\epsilon)pn.$$

Let $S \subset \tilde{\mathcal{I}}$ include (the indices of) the summands equal to 0. Each other summand contributes at least 1, thus $|\tilde{\mathcal{I}}| - |S| = |\tilde{\mathcal{I}} \setminus S| \cdot 1 < (1-6\epsilon)pn$ and by rearranging $|S| > |\tilde{\mathcal{I}}| - (1-6\epsilon)pn > \epsilon pn$. To get the exact size $|S| = \epsilon pn$, we can replace S with an arbitrary subset of it of the exact size. Now define $\bar{S} = \{M(i) \mid i \in S\}$. It is easy to verify that the event $\mathcal{E}_{S,\bar{S}}$ holds. Indeed, each $i \in S$ satisfies $\mathrm{cost}_M[i - \frac{\epsilon}{p}, i + \frac{\epsilon}{p}] = 0$, which implies $M(i) \neq \perp$, and thus $|\bar{S}| = |S|$. Moreover, the block $x[i - \frac{\epsilon}{p}, i + \frac{\epsilon}{p}]$ in x is equal to the corresponds block in y, and these blocks in y are disjoint. This completes the proof of (3).

Next, we prove (4). Fix $S, \bar{S} \subset [n]$ of the same size ℓ, and assume requirements (i) and (ii) hold (otherwise, the probability is 0). Let B_{i_j} and $\bar{B}_{\bar{i}_j}$ be the corresponding blocks in x and in y. Consider for now a given $j \in [\ell]$. The requirement $B_{i_j} = \bar{B}_{\bar{i}_j}$ means that for all $t \in \{-\frac{\epsilon}{p}, \ldots, 0, \ldots, +\frac{\epsilon}{p}\}$ we require $x[i_j + t] = y[\bar{i}_j + t]$. The issue is that x and y are random but correlated through A^p; in particular, the symbols $x[i_j + t]$ and $y[\bar{i}_j + t]$ are chosen independently at random unless A^p aligns their positions, i.e., $A^p(i_j + t) = \bar{i}_j + t$. The key observation is that this last event cannot happen for both $t = -1$ and $t = 1$, because in that case, $A^p(i_j + 1) - A^p(i_j - 1) = \bar{i}_j + 1 - (\bar{i}_j - 1) = 2$; however, $i_j \in \tilde{\mathcal{I}}$ implies that A^p has exactly one edit operation (insertion or deletion) in the interval $[i_j - 1, i_j + 1]$ (and not at its endpoints), thus $A^p(i_j + 1) - A^p(i_j - 1) \in \{1, 3\}$. Assume first that $A^p(i_j + t) \neq \bar{i}_j + t$ for $t = 1$. Then the same must hold also for all $t = 2, \ldots, \frac{\epsilon}{p}$; indeed, we again use that $i_j \in \tilde{\mathcal{I}}$, which implies that A^p has no edit operations near position i_j, thus $A^p(i_j + t) = A^p(i_j + 1) + (t - 1) \neq \bar{i}_j + 1 + (t - 1)$. The argument for $t = -1$ is similar, and we conclude that the requirement $B_{i_j} = \bar{B}_{\bar{i}_j}$ encompasses at least $\frac{\epsilon}{p}$ requirements of the form $x[i_j + t] = y[\bar{i}_j + t]$ where these two positions are not aligned by A^p, and thus these two symbols are chosen independently at random.

The above argument applies to every $j \in [\ell]$, yielding overall at least $\ell \cdot \frac{\epsilon}{p}$ requirements of the form $x[i_j + t] = y[\bar{i}_j + t]$, where these two symbols are chosen independently at random. Observe that each $y[\bar{i}_j + t]$ is either a character $x[t']$ (for t' arising from A^p) or completely independent. Since each character of x appears in at most 2 requirements (once on each

[2] This implies that the blocks $B_{i_1}, \ldots, B_{i_\ell}$ in x are disjoint.

side), we can extract a subset of at one-third of the requirements such that the positions in x appearing there are all distinct, and thus the events are independent.[3] We overall obtain at least $\frac{1}{3}\ell \cdot \frac{\epsilon}{p}$ requirements, each occurring independently with probability $1/|\Sigma|$, and thus

$$\Pr[\mathcal{E}_{S,\bar{S}}] \le |\Sigma|^{-\epsilon\ell/(3p)}.$$

Finally, we are in position to prove the lemma. Combining (3) and (4) and a union bound

$$\begin{aligned}\Pr[\mathrm{ED}(x,y) < (1-6\epsilon)pn] &\le \binom{n}{\ell}^2 \cdot |\Sigma|^{-\frac{\epsilon\ell}{3p}} \le \left(\frac{ne}{\ell}\right)^{2\ell} \cdot 2^{-\frac{\epsilon\ell}{3p}} = \left(\frac{e}{\epsilon p}\right)^{2\epsilon pn} \cdot 2^{-\epsilon^2 n/3} \\ &\le (p^2)^{-2\epsilon pn} \cdot 2^{-\epsilon(15p\log(1/p))n/3} \le p^{-4\epsilon pn+5\epsilon pn} \le p^{\epsilon pn}.\end{aligned}$$

Recall that this was all conditioned on A^p, which had error probability at most $e^{-\epsilon^2 p^2 n/2}$ (by Lemma 6), and now Lemma 7 follows by a union bound. ◀

A similar bound holds for even smaller values of ϵ, provided that the alphabet size is large. The proof is the same, except for the final calculation.

▶ **Lemma 8.** *Suppose* $|\Sigma| \ge (\frac{1}{p})^{15}$. *Then for every* $\epsilon \in [p.\frac{1}{6}]$, *we have* $\Pr[\mathrm{ED}(x,y) < (1-6\epsilon)pn] \le 2e^{-\epsilon^2 p^2 n/2}$.

Following an argument similar to the proof of Lemma 7 we can make the following claim.

▶ **Lemma 9.** *Let* $\epsilon \in [15p\log\frac{1}{p}, \frac{1}{6}]$. *Then with probability at least* $1 - 2e^{-\epsilon^2 p^2 n/2}$, *every alignment* M *between* x, y *satisfies* $|\{i \in \tilde{\mathcal{I}} \mid \mathrm{cost}_M(x[i-\frac{\epsilon}{r}, i+\frac{\epsilon}{r}]) = 0\}| \le 6\epsilon pn$.

Near-Optimal Alignments between x, y. Given $\epsilon > 0$, a potential alignment M between x, y, and an index $i \in [n]$, define the event

$$\mathcal{E}_\epsilon^M(i) := \begin{cases} A^p(i-\frac{\epsilon}{p}) = \min\{M(k) \ne \perp \mid k \in [i-\frac{\epsilon}{p}, i+\frac{\epsilon}{p}]\}; \text{ and} \\ A^p(i+\frac{\epsilon}{p}) = \max\{M(k) \ne \perp \mid k \in [i-\frac{\epsilon}{p}, i+\frac{\epsilon}{p}]\}. \end{cases} \tag{5}$$

By convention, $\mathcal{E}_\epsilon^M(i)$ is *not* satisfied if the minimization/maximization is over the empty set (because $M(k) = \perp$ for all relevant k). We will only use it for $i \in \tilde{\mathcal{I}}$, in which case both $A^p(i-\frac{\epsilon}{p}), A^p(i+\frac{\epsilon}{p}) \ne \perp$. Intuitively, this event means that A^p and M agree on the block boundaries; for example, in the simpler case where all relevant $M(k) \ne \perp$, this event simply means that $A^p(i-\frac{\epsilon}{p}) = M(i-\frac{\epsilon}{p})$ and $A^p(i+\frac{\epsilon}{p}) = M(i+\frac{\epsilon}{p})$.

Denote the set of indices where the event $\mathcal{E}_\epsilon^M(i)$ occurs and the cost of M over substring $x[i-\frac{\epsilon}{p}, i+\frac{\epsilon}{p}]$ is 1, by

$$\tilde{\mathcal{I}}_{\epsilon,M}^{A^p_{x,y}} := \{i \in \tilde{\mathcal{I}} \mid \text{ event } \mathcal{E}_\epsilon^M(i) \text{ occurs and } \mathrm{cost}_M([i-\tfrac{\epsilon}{p}, i+\tfrac{\epsilon}{p}]) = 1\ \}.$$

When clear from the context, we shorten $\tilde{\mathcal{I}}_{\epsilon,M}^{A^p_{x,y}}$ to $\tilde{\mathcal{I}}_M$.

▶ **Lemma 10.** *Let* $\epsilon \in [42p\log\frac{1}{p}, \frac{1}{6}]$ *and* $p \le \delta \le \epsilon$. *Then with probability at least* $1 - 4e^{-\frac{\epsilon^2 p^2 n}{2}}$, *every alignment* M *between* x, y *with* $\mathrm{cost}(M) \le (1+\delta)pn$ *satisfies* $|\tilde{\mathcal{I}}_M| \ge (1 - 23\epsilon - \delta)pn$.

[3] To see this, consider an auxiliary graph whose a vertex for each character $x[t]$, and connect two by an edge if they appear in the same constraint. Since every vertex has degree at most 2, a greedy matching contains at least one third of the edges.

At a high level, the proof follows the outline of Lemma 7, and avoids a direct union bound over all (relevant) potential alignments, because there are too many of them. Instead, we introduce a smaller set of basic events that "covers" all these potential alignments. However the analysis is more elaborate with additional cases that require new technical ideas. The proof appears in the full version.

A similar bound holds for even smaller values of ϵ, provided that the alphabet size is large.

▶ **Lemma 11.** *Suppose* $|\Sigma| \geq (\frac{1}{p})^{42}$. *Then for every* $\epsilon \in [p, \frac{1}{6}]$ *and every* $p \leq \delta \leq \epsilon$, *with probability at least* $1 - 4e^{-\frac{\epsilon^2 p^2 n}{2}}$, *every alignment* M *between* x, y *with* $\text{cost}(M) \leq (1+\delta)pn$ *satisfies* $|\tilde{\mathcal{I}}_M| \geq (1 - 23\epsilon - \delta)pn$.

4 Robustness of Approximate Median

In this section, we consider the (approximate) median string problem on a set of strings generated by our probabilistic model G_p (from Section 2). For a random (unknown) string $s \in \Sigma^n$, G_p generates a set $S = \{s_1, s_2, \cdots, s_m\}$ of independent traces of s. We show that with high probability, any $(1+\epsilon)$-approximate median of S must be close (in edit distance) to the unknown string s. In other words, any $(1+\epsilon)$-approximate median must "agree" with the unknown string s in most of the portions. It is true even when $m = 3$. In this section, we state the results and the proofs by considering $m = 3$. In particular, we prove Theorem 3. At the end of the section, we remark on why such result with three traces also directly provides a similar result for any $m > 3$ traces. Another way to interpret this result is the following. Suppose we take a set of three traces and find its $(1+\epsilon)$-approximate median. Then if we add more traces in the set, its $(1+\epsilon)$-approximate median does not change by much. So in some sense, $(1+\epsilon)$-approximate median is robust in the case of average-case traces.

For the purpose of the analysis, we start by considering infinite length strings (as in Section 2), and then later we will move to the finite-length versions. Recall, U denotes the uniform distribution over strings $x \in \Sigma^{\mathbb{N}}$, i.e., each character $x[i]$, for $i \in \mathbb{N}$, is chosen uniformly at random and independently from Σ. Consider a parameter $p \in (0, 0.001)$ and define $q := \frac{p(4-3p)}{2-p^2}$. (Note, $q = 2p - \Theta(p^2)$.) Then consider the following two processes:

- **Process 1**: Draw a string s from U. Then draw three strings s_1, s_2, s_3 independently from $G_p(s)$. Output the tuple (s, s_1, s_2, s_3).
- **Process 2**: Draw a string x_1 from U. Then draw $\bar{x}$ from $G_p(x_1)$ (and denote the corresponding alignment function by $A^p_{1,\bar{x}}$). Finally, draw x_2, x_3 independently from $G_p(\bar{x})$ (and denote the corresponding alignment functions by $A^p_{\bar{x},2}, A^p_{\bar{x},3}$ respectively). Output the tuple $(\bar{x}, x_1, x_2, x_3)$.

As an immediate corollary of Proposition 18 (see Appendix A), we know that the distributions on (s, s_1) and $(\bar{x}, x_1)$ are the same. So we conclude the following about the above two processes.

▷ Claim 12. The probability distributions on (s, s_1, s_2, s_3) and $(\bar{x}, x_1, x_2, x_3)$, the tuples generated by Process 1 and Process 2 respectively, are identical.

Note, we want to investigate the property of an approximate median of the strings generated through Process 1. Due to the above claim, instead of considering the strings s_1, s_2, s_3 from now on we focus on x_1, x_2, x_3 generated through Process 2. By Proposition 16 (see Appendix A), both x_2 and x_3 can be viewed as strings drawn from $G_q(x_1)$. Let us

use the notations $A^q_{1,2}$ and $A^q_{1,3}$ to denote the alignment functions produced by the random process G_q while generating x_2 and x_3 respectively, from x_1. We want to emphasize that the process G_q is considered solely for the purpose of the analysis.

Next, we use the alignments $A^p_{1,\bar{x}}$, $A^p_{\bar{x},2}$ (and $A^p_{\bar{x},3}$) to define an alignment between x_1, x_2 (and x_1, x_3) via $\bar{x}$. Let $A^p_{1,\bar{x},2}$ and $A^p_{1,\bar{x},3}$ denote $A^p_{\bar{x},2} \circ A^p_{1,\bar{x}}$ and $A^p_{\bar{x},3} \circ A^p_{1,\bar{x}}$ respectively. (See Section 1.3 for the definition of the notation $\circ$.)

Median of n-length prefixes of x_1, x_2, x_3. So far in this section we have talked about infinite length strings. From now on we restrict ourselves to the the n-length prefixes of x_1, x_2 and x_3 denoted by $x_1[1,n], x_2[1,n]$ and $x_3[1,n]$ respectively. By abusing the notations, we simply use x_1, x_2 and x_3 to also denote $x_1[1,n], x_2[1,n]$ and $x_3[1,n]$ respectively. Also, we consider the (n-length) restriction of all the alignment functions (defined so far) accordingly. Again, for simplicity, we use the same notations to refer to these restricted alignment functions.

Now, we consider the (approximate) median string problem on the set $S = \{x_1, x_2, x_3\}$. Recall, for any string y, $\texttt{Obj}(S, y) := \sum_{k=1}^{3} \text{ED}(x_k, y)$, and $\texttt{OPT}(S) = \min_{y \in \Sigma^*} \texttt{Obj}(S, y)$. Since throughout this section, $S = \{x_1, x_2, x_3\}$, to simplify the notations, we drop S from both $\texttt{Obj}$ and $\texttt{OPT}$. The main result of this section is the following.

▶ **Theorem 13.** *For a large enough $n \in \mathbb{N}$ and a noise parameter $p \in (0, 0.001)$, let $\bar{x}, x_1, x_2$ and x_3 be the n-length prefixes of the strings generated by Process 2. If x_{med} is a $(1+\epsilon)$-approximate median of $S = \{x_1, x_2, x_3\}$ for $\epsilon \in [110p \log(1/p), 1/6]$, then* $\Pr[\text{ED}(\bar{x}, x_{med}) \leq 195\epsilon \cdot \texttt{OPT}(S)] \geq 1 - e^{-\log^2 n}$.

We would like to emphasize that (for the simplicity in the analysis) we have made no attempt to optimize the constants. By a more careful analysis, both the range of p and the constant involved in the bound of $\text{ED}(\bar{x}, x_{\text{med}})$ could be improved significantly. The above theorem together with Claim 12 immediately gives us Theorem 3. Note, in Theorem 3, we do not have any length restrictions on the traces. On the other hand, the above theorem considers $\bar{x}, x_1, x_2$ and x_3 to be of length n. However, by a standard application of Chernoff-Hoeffding bound, it suffices to restrict ourselves to the $(n - \sqrt{n} \log n)$-length prefixes of all the traces (of Theorem 3). Then we can apply the above theorem over them, to get Theorem 3. The proof of Theorem 13 appears in the full version.

For more than three traces. So far, we have shown that for any set $\{s_1, s_2, s_3\}$ of three traces of s, its any $(1+\epsilon)$-approximate median is close to s. Below we argue that a similar result for any arbitrary number (less than some $\text{poly}(n)$) of traces directly follows.

▶ **Corollary 14.** *For a large enough $n \in \mathbb{N}$ and a noise parameter $p \in (0, 0.001)$, let the string $s \in \{0,1\}^n$ be chosen uniformly at random, and let $s_1, \cdots, s_m$ be $m = n^{O(1)}$ traces generated by $G_p(s)$. If x_{med} is a $(1+\epsilon)$-approximate median of $S = \{s_1, \cdots, s_m\}$ for any $\epsilon \in [110p \log(1/p), 1/6]$, then* $\Pr[\text{ED}(s, x_{med}) \leq O(\epsilon) \cdot \frac{\texttt{OPT}(S)}{m}] \geq 1 - n^{-1}$.

We defer the proof to the full version.

5 Near-Linear time Approximate Trace Reconstruction

In this section, we describe a linear-time algorithm that reconstructs the unknown string using only three traces, up to some small edit error. In particular, we prove Theorem 1. Before describing our linear-time algorithm, first note, we can compute an (exact) median

of three traces using a standard dynamic programming algorithm [53, 34] in cubic time. Then by Theorem 13, that median string will be close (in edit distance) to the unknown string. More specifically, the edit distance between the computed median string and the unknown string will be at most $O(\epsilon pn)$ with high probability. In this section, we design a more sophisticated method to compute an approximation of the unknown string. For that purpose, we first divide each trace into "well-separated" blocks of size $\log^2 n$ each. Then we run the dynamic programming-based median algorithm [53, 34] on these small blocks. Thus we spend only $\operatorname{poly}\log n$ time per block, and hence in total $\tilde{O}(n)$ time. Since we consider "well-separated" blocks, they are independent. Thus we apply Theorem 13 for each of these blocks (instead of the whole string). Using standard Chernoff-Hoeffding bound, we get that most of these block medians are close to their corresponding block of the unknown string. Hence, by concatenating these block medians, we get back the whole unknown string up to some small edit error. We defer the detailed description of our algorithm to Appendix B.

6 Conclusion

Trace reconstruction in the average case is a well-studied problem. The problem is to reconstruct an unknown (random) string by reading a few traces of it generated via some noise (insertion-deletion) channel. The main objective here is to minimize the sample complexity and also the efficiency of the reconstruction algorithm. There is an exponential gap between the current best upper and lower bound in the sample complexity despite several attempts. The best lower bound is $\tilde{\Omega}(\log^{5/2} n)$ [9]. A natural question is whether it is possible to beat this lower bound by allowing some error in the reconstructed string. This version is also referred to as the approximate trace reconstruction problem; however, nothing is known except for a few special cases.

Our result not only beats the lower bound of the exact trace reconstruction but uses only three traces. The reconstructed string is $O(\epsilon pn)$ close (in edit distance) to the unknown string with high probability. We establish a connection between the approximate trace reconstruction and the approximate median string problem, another utterly significant problem. We show that both the problems are essentially the same. We leverage this connection to design a near-linear time approximate reconstruction algorithm using three traces.

An exciting future direction is to get a similar result for the worst-case, where the unknown string is arbitrary. It will also be fascinating if we could show some non-trivial sample complexity lower bound for that version.

References

1 J. Abreu and Juan Ramón Rico-Juan. A new iterative algorithm for computing a quality approximate median of strings based on edit operations. *Pattern Recognition Letters*, 36:74–80, 2014.

2 Nir Ailon, Moses Charikar, and Alantha Newman. Aggregating inconsistent information: Ranking and clustering. *J. ACM*, 55(5):23:1–23:27, 2008. doi:10.1145/1411509.1411513.

3 Frank Ban, Xi Chen, Adam Freilich, Rocco A Servedio, and Sandip Sinha. Beyond trace reconstruction: Population recovery from the deletion channel. In *60th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 745–768. IEEE, 2019.

4 Tugkan Batu, Funda Ergün, Joe Kilian, Avner Magen, Sofya Raskhodnikova, Ronitt Rubinfeld, and Rahul Sami. A sublinear algorithm for weakly approximating edit distance. In *Proceedings of the 35th Annual ACM Symposium on Theory of Computing*, STOC '03, pages 316–324. ACM, 2003.

5 Tugkan Batu, Sampath Kannan, Sanjeev Khanna, and Andrew McGregor. Reconstructing strings from random traces. In *Proceedings of the Fifteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2004*, pages 910–918. SIAM, 2004.

6 Hervé Cardot, Peggy Cénac, and Antoine Godichon-Baggioni. Online estimation of the geometric median in Hilbert spaces: Nonasymptotic confidence balls. *Annals of Statistics*, 45(2):591–614, 2017. doi:10.1214/16-AOS1460.

7 Francisco Casacuberta and M. D. Antonio. A greedy algorithm for computing approximate median strings. In *Proc. of National Symposium on Pattern Recognition and Image Analysis*, pages 193–198, 1997.

8 Diptarka Chakraborty, Debarati Das, and Robert Krauthgamer. Approximating the median under the ulam metric. In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 761–775. SIAM, 2021. doi:10.1137/1.9781611976465.48.

9 Zachary Chase. New lower bounds for trace reconstruction. *Annales de l'Institut Henri Poincaré, Probabilités et Statistiques*, 57(2):627–643, 2021.

10 Zachary Chase. Separating words and trace reconstruction. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, pages 21–31, 2021.

11 Xi Chen, Anindya De, Chin Ho Lee, Rocco A. Servedio, and Sandip Sinha. Polynomial-time trace reconstruction in the low deletion rate regime. In James R. Lee, editor, *12th Innovations in Theoretical Computer Science Conference, ITCS 2021*, volume 185 of *LIPIcs*, pages 20:1–20:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021.

12 Xi Chen, Anindya De, Chin Ho Lee, Rocco A. Servedio, and Sandip Sinha. Polynomial-time trace reconstruction in the smoothed complexity model. In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms*, pages 54–73. SIAM, 2021.

13 Mahdi Cheraghchi, Ryan Gabrys, Olgica Milenkovic, and Joao Ribeiro. Coded trace reconstruction. *IEEE Transactions on Information Theory*, 66(10):6084–6103, 2020.

14 Flavio Chierichetti, Ravi Kumar, Sandeep Pandey, and Sergei Vassilvitskii. Finding the Jaccard median. In *Proceedings of the twenty-first annual ACM-SIAM symposium on Discrete Algorithms*, pages 293–311. SIAM, 2010. doi:10.1137/1.9781611973075.25.

15 Michael B. Cohen, Yin Tat Lee, Gary Miller, Jakub Pachocki, and Aaron Sidford. Geometric median in nearly linear time. In *Proceedings of the forty-eighth annual ACM Symposium on Theory of Computing*, pages 9–21, 2016.

16 Sami Davies, Miklós Z. Rácz, Cyrus Rashtchian, and Benjamin G. Schiffer. Approximate trace reconstruction. *CoRR*, abs/2012.06713, 2020. arXiv:2012.06713.

17 Anindya De, Ryan O'Donnell, and Rocco A Servedio. Optimal mean-based algorithms for trace reconstruction. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, pages 1047–1056, 2017.

18 Colin de la Higuera and Francisco Casacuberta. Topology of strings: Median string is NP-complete. *Theor. Comput. Sci.*, 230(1-2):39–48, 2000. doi:10.1016/S0304-3975(97)00240-5.

19 Cynthia Dwork, Ravi Kumar, Moni Naor, and D. Sivakumar. Rank aggregation methods for the web. In *Proceedings of the Tenth International World Wide Web Conference, WWW 10*, pages 613–622, 2001. doi:10.1145/371920.372165.

20 Igor Fischer and Andreas Zell. String averages and self-organizing maps for strings. *Proceedings of the neural computation*, pages 208–215, 2000.

21 P. Thomas Fletcher, Suresh Venkatasubramanian, and Sarang Joshi. Robust statistics on riemannian manifolds via the geometric median. In *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2008.

22 Zvi Galil and Kunsoo Park. An improved algorithm for approximate string matching. *SIAM Journal on Computing*, 19(6):989–999, 1990.

23 Nick Goldman, Paul Bertone, Siyuan Chen, Christophe Dessimoz, Emily M. LeProust, Botond Sipos, and Ewan Birney. Towards practical, high-capacity, low-maintenance information storage in synthesized DNA. *Nature*, 494(7435):77–80, 2013.

24 Elena Grigorescu, Madhu Sudan, and Minshen Zhu. Limitations of mean-based algorithms for trace reconstruction at small distance. *CoRR*, abs/2011.13737, 2020. arXiv:2011.13737.

25 Dan Gusfield. *Algorithms on Strings, Trees, and Sequences - Computer Science and Computational Biology*. Cambridge University Press, 1997.

26 Morihiro Hayashida and Hitoshi Koyano. Integer linear programming approach to median and center strings for a probability distribution on a set of strings. In *BIOINFORMATICS*, pages 35–41, 2016.

27 Nina Holden and Russell Lyons. Lower bounds for trace reconstruction. *The Annals of Applied Probability*, 30(2):503–525, 2020.

28 Nina Holden, Robin Pemantle, Yuval Peres, and Alex Zhai. Subpolynomial trace reconstruction for random strings and arbitrary deletion probability. *Mathematical Statistics and Learning*, 2(3):275–309, 2020.

29 Thomas Holenstein, Michael Mitzenmacher, Rina Panigrahy, and Udi Wieder. Trace reconstruction with constant deletion probability and related results. In *Proceedings of the Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2008*, pages 389–398. SIAM, 2008.

30 V. V. Kalashnik. Reconstruction of a word from its fragments. *Computational Mathematics and Computer Science (Vychislitel'naya matematika i vychislitel'naya tekhnika), Kharkov*, 4:56–57, 1973.

31 Sampath Kannan and Andrew McGregor. More on reconstructing strings from random traces: insertions and deletions. In *Proceedings. International Symposium on Information Theory, 2005. ISIT 2005.*, pages 297–301. IEEE, 2005.

32 Claire Kenyon-Mathieu and Warren Schudy. How to rank with few errors. In *Proceedings of the Thirty-Ninth Annual ACM Symposium on Theory of Computing*, STOC '07, pages 95–103. ACM, 2007. doi:10.1145/1250790.1250806.

33 Teuvo Kohonen. Median strings. *Pattern Recognition Letters*, 3(5):309–313, 1985. doi:10.1016/0167-8655(85)90061-3.

34 Joseph B Kruskal. An overview of sequence comparison: Time warps, string edits, and macromolecules. *SIAM review*, 25(2):201–237, 1983. doi:10.1137/1025045.

35 Ferenc Kruzslicz. Improved greedy algorithm for computing approximate median strings. *Acta Cybernetica*, 14(2):331–339, 1999.

36 Gad M. Landau and Uzi Vishkin. Fast parallel and serial approximate string matching. *Journal of Algorithms*, 10(2):157–169, 1989.

37 Vladimir I. Levenshtein. Efficient reconstruction of sequences. *IEEE Transactions on Information Theory*, 47(1):2–22, 2001. doi:10.1109/18.904499.

38 Vladimir I. Levenshtein. Efficient reconstruction of sequences from their subsequences or supersequences. *Journal of Combinatorial Theory, Series A*, 93(2):310–332, 2001. doi:10.1006/jcta.2000.3081.

39 Carlos D. Martínez-Hinarejos, Alfons Juan, and Francisco Casacuberta. Use of median string for classification. In *Proceedings 15th International Conference on Pattern Recognition. ICPR-2000*, volume 2, pages 903–906. IEEE, 2000. doi:10.1109/ICPR.2000.906220.

40 Andrew McGregor, Eric Price, and Sofya Vorotnikova. Trace reconstruction revisited. In *Algorithms - ESA 2014 - 22th Annual European Symposium, Wroclaw, Poland, September 8-10, 2014. Proceedings*, volume 8737 of *Lecture Notes in Computer Science*, pages 689–700. Springer, 2014.

41 Stanislav Minsker. Geometric median and robust estimation in Banach spaces. *Bernoulli*, 21(4):2308–2335, 2015.

42 P. Mirabal, J. Abreu, and D. Seco. Assessing the best edit in perturbation-based iterative refinement algorithms to compute the median string. *Pattern Recognition Letters*, 120:104–111, April 2019.

43 Michael Mitzenmacher. A survey of results for deletion channels and related synchronization channels. *Probability Surveys*, 6:1–33, 2009.

44 Shyam Narayanan. Population recovery from the deletion channel: Nearly matching trace reconstruction bounds. *arXiv preprint*, 2020. arXiv:2004.06828.

45 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 2017*, pages 1042–1046. ACM, 2017.

46 François Nicolas and Eric Rivals. Complexities of the centre and median string problems. In *14th Annual Symposium on Combinatorial Pattern Matching, CPM 2003*, pages 315–327, 2003.

47 Lee Organick, Siena Dumas Ang, Yuan-Jyue Chen, Randolph Lopez, Sergey Yekhanin, Konstantin Makarychev, Miklos Z. Racz, Govinda Kamath, Parikshit Gopalan, Bichlien Nguyen, et al. Random access in large-scale dna data storage. *Nature biotechnology*, 36(3):242, 2018.

48 Oscar Pedreira and Nieves R. Brisaboa. Spatial selection of sparse pivots for similarity search in metric spaces. In *International Conference on Current Trends in Theory and Practice of Computer Science*, pages 434–445. Springer, 2007.

49 Yuval Peres and Alex Zhai. Average-case reconstruction for the deletion channel: Subpolynomially many traces suffice. In *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017*, pages 228–239. IEEE Computer Society, 2017.

50 Pavel Pevzner. *Computational molecular biology: an algorithmic approach*. MIT press, 2000.

51 Cyrus Rashtchian, Konstantin Makarychev, Miklós Z. Rácz, Siena Ang, Djordje Jevdjic, Sergey Yekhanin, Luis Ceze, and Karin Strauss. Clustering billions of reads for DNA data storage. In *Advances in Neural Information Processing Systems 30*, pages 3360–3371. Curran Associates, Inc., 2017.

52 Richard J Roberts, Mauricio O Carneiro, and Michael C Schatz. The advantages of smrt sequencing. *Genome Biology*, 14(7):405, 2013.

53 David Sankoff. Minimal mutation trees of sequences. *SIAM Journal on Applied Mathematics*, 28(1):35–42, 1975. doi:10.1137/0128004.

54 Krishnamurthy Viswanathan and Ram Swaminathan. Improved string reconstruction over insertion-deletion channels. In *Proceedings of the nineteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 399–408, 2008.

55 SM Hossein Tabatabaei Yazdi, Ryan Gabrys, and Olgica Milenkovic. Portable and error-free dna-based data storage. *Scientific reports*, 7(1):1–6, 2017.

A Description of Probabilistic Channel

Model $R_p(x)$. Given an infinite-length string $x \in \Sigma^{\mathbb{N}}$ and a parameter $p \in [0,1]$. Consider the following random procedure:

1. Initialize $i = 1$. (We use i to point to the current index positions of the input string.) Also, initialize an empty string Out.
2. Do the following independently at random:
   a. With probability $1-p$, set $Out \leftarrow Out \odot x[i]$ and increment i. (Match $x[i]$.)
   b. With probability $p/2$, increment i. (Delete $x[i]$.)
   c. With probability $p/2$, choose independently uniformly at random a character $a \in \Sigma$ and set $Out \leftarrow Out \odot a$. (Insert a random character.)

We call this procedure R_p, and denote the randomized output string Out by $R_p(x)$.

Model $G_p(x)$. This model first provides a randomized mapping (alignment) $A^p : \mathbb{N} \to \mathbb{N} \cup \{\perp\}$, and then uses this alignment (and x) to generate the output string. First, given a parameter $p \in [0,1]$, consider the following random procedure to get a mapping A^p:

1. Initialize $i = 1$ and $j = 1$. (Indices of current positions in the input and output strings, respectively.)
2. Do the following independently at random:
   a. With probability $1-p$, set $A^p(i) \leftarrow j$ and increment both i and j. (Match $x[i]$.)
   b. With probability $p/2$, set $A^p(i) \leftarrow \perp$ and increment i. (Delete $x[i]$.)
   c. With probability $p/2$, increment j. (Insertion.)

Next, use this A^p and a given string $x \in \Sigma^{\mathbb{N}}$ to generate a string $y \in \Sigma^{\mathbb{N}}$ as follows. For each $j \in \mathbb{N}$,

- If there is $i \in \mathbb{N}$ with $A^p(i) = j$ then set $y[j] \leftarrow x[i]$. (Match $x[i]$.)
- Otherwise, choose independently uniformly at random a character $a \in \Sigma$ and set $y[j] \leftarrow a$. (Insert a random character.)

We denote by $G_p(x)$ the randomized string y generated as above. By construction, A^p is an alignment between x and $G_p(x)$.

Basic Properties. We claim next that R_p and G_p are equivalent, which is useful because we find it more convenient to analyze G_p. We use $X_1 \overset{\text{dist}}{=} X_2$ to denote that two random variable $X_i \sim D_i$, $i \in \{1, 2\}$ have equal distribution, i.e., $D_1 = D_2$. The next two propositions are immediate.

▶ **Proposition 15.** *For every string $x \in \Sigma^{\mathbb{N}}$ and $p \in [0, 1]$, we have $R_p(x) \overset{\text{dist}}{=} G_p(x)$.*

▶ **Proposition 16** (Transitivity)**.** *For every $x \in \Sigma^{\mathbb{N}}$ and $p \in [0, 1]$, let*

$$q(p) := \frac{p(4 - 3p)}{2 - p^2}. \tag{6}$$

Then $G_p(G_p(x)) \overset{\text{dist}}{=} G_{q(p)}(x)$.

Additional Properties (Random Base String). Let U be the uniform distribution over strings $x \in \Sigma^{\mathbb{N}}$, i.e., each character $x[i]$ is chosen uniformly at random and independently from Σ. We now state two important observations regarding the process G_p. The first one is a direct corollary of Proposition 16. The second observation follows because the probability of insertion is the same as that of deletion at any index in the random process G_p.

▶ **Corollary 17** (Transitivity)**.** *Let $p \in [0, 1]$ and let $q(p)$ be as in (6). Draw a random string $X \sim U$, and use it to draw $Y \sim G_p(G_p(X))$ and $Z \sim G_{q(p)}(X)$. Then $(X, Y) \overset{\text{dist}}{=} (X, Z)$.*

▶ **Proposition 18** (Symmetry)**.** *Let $p \in [0, 1]$. Draw a random string $X \sim U$ and use it to draw another string $Y \sim G_p(X)$. Then $(X, Y) \overset{\text{dist}}{=} (Y, X)$.*

B Near-Linear-Time Median Algorithm

Formally, our result is the following.

▶ **Theorem 19.** *There is a small non-negative constant $c_0 < 1$ and a deterministic algorithm that, for every sufficiently large $n \in \mathbb{N}$ and noise parameter $p \in (0, c_0]$, given as input three traces $s_1, s_2, s_3 \sim G_p(s)$, for a uniformly random (but unknown) string $s \in \{0, 1\}^n$, and an accuracy parameter $\epsilon \in [110p \log(1/p), 1/6]$, outputs in time $\tilde{O}(n)$ a string z that satisfies $\Pr[\mathrm{ED}(s, z) \le 5270\epsilon pn] \ge 1 - n^{-1}$.*

Before describing the algorithm we would like to introduce a few notations, which we use in this section. For a string $x \in \Sigma^n$, let $y = x[i, i + 1, \cdots, j]$ be a substring of it. Then we use the notation $\texttt{start}(y)$ to denote the index i and $\texttt{end}(y)$ to denote the index j.

Description of the algorithm. Let us now describe the algorithm. First, partition s_1 into $r = \frac{|s_1|}{\ell}$ disjoint blocks $s_1^1, s_1^2, \cdots, s_1^r$ each of length $\ell = \log^2 n + \frac{240}{p}\log^{3/2} n$. For each s_1^i, let us call the middle $\log^2 n$-size sub-block, denoted by y_1^i, an *anchor*. Next, for each $i \in [r]$ and $j \in \{2,3\}$, find the *best match* of the anchor y_1^i in the string s_j i.e., for each y_1^i find a substring (breaking ties arbitrarily) in s_j that has the minimum edit distance with y_1^i. Let us denote the matched substrings in s_2 and s_3 by y_2^i and y_3^i respectively. Then for each $j \in \{2,3\}$, we divide s_j into blocks $s_j^1, \cdots, s_j^r$ (some of the s_j^i's could be empty) by treating $y_j^1, \cdots, y_j^r$ as anchors. More specifically,

- Set the start index of s_j^1 to be 1. For any other non-empty block s_j^i, set its start index to be $\lfloor(\texttt{end}(y_j^{i-1}) + \texttt{start}(y_j^i))/2\rfloor$.
- For the last non-empty block in s_j, set its end index to be $|s_j|$. For any other non-empty block s_j^i, set its end index to be $\lfloor(\texttt{end}(y_j^i) + \texttt{start}(y_j^{i+1}))/2\rfloor - 1$.

Next, for each $i \in [r]$, compute a median of $\{s_1^i, s_2^i, s_3^i\}$, and let it be denoted by z^i. Finally, output $z = z^1 \odot \cdots \odot z^r$ (i.e., the concatenation of all the z^i's).

Correctness proof. Before proceeding with the correctness proof, let us state a known fact about the edit distance between two random strings from [4].

▶ **Proposition 20** ([4]). *For any two strings $x \in \Sigma^m$ and $y \in \Sigma^n$ drawn uniformly at random,* $\Pr[\text{ED}(x,y) \geq \frac{\max\{m,n\}}{10}] \geq 1 - 2^{-\max\{m,n\}/10}$.

▷ Claim 21. For every two substrings x, y of length at least $60\log n$, of s_i, s_j respectively, where $i, j \in [3]$, such that $(A^p_{s,s_i})^{-1}(x)$ and $(A^p_{s,s_j})^{-1}(y)$ are two disjoint substrings of s, $\Pr[\text{ED}(x,y) \geq \frac{\max\{|x|,|y|\}}{10}] \geq 1 - n^{-4}$.

Proof. By Proposition 18, x and y are two strings chosen uniformly at random by picking each of its symbols independently uniformly at random from Σ. Then the claim directly follows from Proposition 20 together with a standard application of union bound. ◁

Let us now define *true match* for each block y_1^i in strings s_2 and s_3. For each $j \in \{2,3\}$, we call the block $A^p_{s,s_j}((A^p_{s,s_1})^{-1}(y_1^i))$ in s_j the true match of y_1^i, denoted by t_j^i. Next, we want to claim that for each block y_1^i, its best match y_j^i in a string s_j, for $j \in \{2,3\}$, is close to its true match t_j^i. The following lemma is crucial to show the correctness of the algorithm and also to establish a linear-time bound for the algorithm.

▷ Claim 22. For each $i \in [r]$ and $j \in \{2,3\}$, with probability $1 - 5n^{-2}$,

1. $|\texttt{start}(t_j^i) - \texttt{start}(y_j^i)| \leq \frac{200}{p}\log n$, and
2. $|\texttt{end}(t_j^i) - \texttt{end}(y_j^i)| \leq \frac{200}{p}\log n$.

Proof. Let us partition y_1^i into $\frac{p\log n}{10}$ sub-blocks $y_1^{i,1}, \cdots, y_1^{i,(p\log n)/10}$, each of size $\frac{10\log n}{p}$. Next, for each of these sub-blocks $y_1^{i,k}$ consider its true match in the string s_j (for any $j \in \{2,3\}$) defined as $t_j^{i,k} := A^p_{s,s_j}((A^p_{s,s_1})^{-1}(y_1^{i,k}))$.

Now, consider an (arbitrary) optimal alignment B between y_1^i and y_j^i. Observe, if for all $1 \leq k \leq (p\log n)/10$, $B(y_1^{i,k})$ has a non-empty overlap with the corresponding true match $t_j^{i,k}$, then the claim is true. So from now on, let us assume that at least for some block $y_1^{i,k}$, $B(y_1^{i,k})$ does not overlap with $t_j^{i,k}$. Let $y_1^{i,k'}$ be the right-most sub-block such that there is a non-empty overlapping between $B(y_1^{i,k'})$ and $t_j^{i,k'}$. Then for each $k'+1 \leq k \leq (p\log n)/10$, by Claim 21, $\text{cost}_B(y_1^{i,k}) \geq \frac{\log n}{p}$ with probability at least $1 - n^{-4}$.

We want to claim that $k' \geq \frac{p \log n}{10} - \frac{10}{1-24p}$. If not, then we deduce that y_j^i is not the best match of y_1^i in the string s_j. To argue this, suppose $k' < \frac{p \log n}{10} - \frac{10}{1-24p}$. Then we modify the mapping B to derive another mapping B' as follows: B' respects B till the block $y_1^{i,k'-1}$. Next, B' deletes the block $y_1^{i,k'}$, and then use the mapping $A^p_{s,s_j} \circ (A^p_{s,s_1})^{-1}$ to map the remaining blocks $y_1^{i,k'+1}, \cdots, y_1^{i,(p \log n)/10}$. By Lemma 5 (applied on strings of size at least $\frac{10 \log n}{p}$) together with an union bound, we get that for all the blocks of s_1 of size at least $\frac{10 \log n}{p}$, the cost of the alignment $A^p_{s,s_j} \circ (A^p_{s,s_1})^{-1}$ is at most $24 \log n$ with probability at least $1 - 2n^{-2}$.

Clearly, the cost of this new alignment B' is at least $(\frac{10}{1-24p}+1)\frac{\log n}{p} - (\frac{10 \log n}{p} + \frac{240 \log n}{1-24p}) > 0$ less than that of B. Hence, y_j^i cannot be the best match of y_1^i in s_j. So we deduce that $k' \geq \frac{p \log n}{10} - \frac{10}{1-24p}$. Note, if an alignment function just deletes all the blocks $y_1^{i,k'+1}, \cdots, y_1^{i,(p \log n)/10}$, it would cost at most $\frac{100 \log n}{p(1-24p)}$. Thus, since t_j^i is the best match of y_1^i, the cost of B for these blocks $y_1^{i,k'+1}, \cdots, y_1^{i,(p \log n)/10}$ must be at most $\frac{100 \log n}{p(1-24p)}$. From this we conclude that $|\texttt{end}(t_j^i) - \texttt{end}(y_j^i)| \leq \frac{100}{p(1-24p)} \log n \leq \frac{200}{p} \log n$ (for the choice of p we have).

Similarly, we can argue that $|\texttt{start}(t_j^i) - \texttt{start}(y_j^i)| \leq \frac{200}{p} \log n$. This concludes the proof. ◁

The following is an immediate corollary of the above claim.

▶ **Corollary 23.** *With probability at least $1 - 10n^{-2}$, for each $i \in [r]$ and $j \in \{2, 3\}$, y_j^i and y_j^{i+1} do not overlap.*

Proof. Consider the substring between the blocks y_1^i and y_1^{i+1}, which is of length $\frac{480}{p} \log^{3/2} n$. By Lemma 5, $A^p_{s,s_j} \circ (A^p_{s,s_1})^{-1}$ maps that substring into a substring of length at least $240 \log^{3/2} n > \frac{400}{p} \log n$ in s_j with probability at least $1 - n^{-3}$. Now, it directly follows from Claim 22 that y_j^i and y_j^{i+1} do not overlap. ◀

Next, we use the above to establish an upper bound on the edit distance between the unknown string s and the recovered string z.

▶ **Lemma 24.** *With probability at least $1 - n^{-1}$, $\mathrm{ED}(s, z) \leq 1550\epsilon pn$.*

Proof. For any $i \in [r]$, consider the set $S^i := \{s_1^i, s_2^i, s_3^i\}$. Consider the substring y^i of the string s such that $A^p_{s,s_1}(y^i) = y_1^i$ (i.e., y^i maps to y_1^i by the alignment A^p_{s,s_1}). Next, for the analysis purpose, consider the set $T^i = \{y_1^i, t_2^i, t_3^i\}$. Recall, by the definition of $t_j^i = A^p_{s,s_j}(y^i)$, for $j \in \{2, 3\}$ are the traces generated by G_p from the block y^i.

Thus by Lemma 5, with probability at least $1 - 3n^{-4}$, for each $t \in T^i$, $\mathrm{ED}(y^i, t) \leq (1 + \epsilon)p|y^i|$. Since y_1^i is a substring of s_1^i (where $|s_1^i| = |y_1^i| + \frac{240}{p} \log^{3/2} n$), by triangular inequality, $\mathrm{ED}(y^i, s_1^i) \leq (1 + \epsilon)p|y^i| + \frac{240}{p} \log^{3/2} n$. Next observe, for each $j \in \{2, 3\}$, by Corollary 23, t_j^i is a substring of s_j^i. Furthermore, by definition, $|s_j^i| \leq |y_j^i| + \frac{240}{p} \log^{3/2} n$. Thus, again by triangular inequality, $\mathrm{ED}(y^i, s_j^i) \leq (1 + \epsilon)p|y^i| + \frac{240}{p} \log^{3/2} n$. So we get

$$\texttt{Obj}(S^i, y^i) \leq 3(1 + \epsilon)p|y^i| + \frac{750}{p} \log^{3/2} n. \tag{7}$$

Since z^i is an (exact) median of S^i, $\texttt{Obj}(S^i, z^i) \leq \texttt{Obj}(S^i, y^i)$. Next, it follows from Claim 22 and the construction of the blocks s_j^i, for $j \in \{2, 3\}$, that t_j^i is a substring of s_j^i where $|t_j^i| \geq |s_j^i| - \frac{500}{p} \log^{3/2} n$. Hence, we can deduce that

$$\begin{aligned}\texttt{Obj}(T^i, z^i) &\le \texttt{Obj}(S^i, z^i) + \frac{1500}{p}\log^{3/2} n \\ &\le \texttt{Obj}(S^i, y^i) + \frac{1500}{p}\log^{3/2} n \\ &\le 3(1+\epsilon)p|y^i| + \frac{2500}{p}\log^{3/2} n \qquad \text{by (7).} \end{aligned} \tag{8}$$

Further observe, it follows from Proposition 16 and Lemma 7, for each $j \in \{2, 3\}$,

$$\mathrm{ED}(y_1^i, t_j^i) \ge (1-6\epsilon)q|y_1^i| = (1-6\epsilon)q\log^2 n.$$

Recall, $q = 2p - \Theta(p^2)$. Then

$$\texttt{OPT}(T^i) \ge 3(1-7\epsilon)p\log^2 n. \tag{9}$$

From (8) and (9), we conclude that z^i is an $(1+9\epsilon)$-approximate median of T^i. So by Theorem 13, $\mathrm{ED}(y^i, z^i) \le 1755\epsilon \cdot \texttt{OPT}(T^i)$ with probability at least $1 - e^{-2\log^2(\log n)}$.

Now, since all the y^i's are generated by picking each symbol uniformly at random and by our construction for each $j \in \{2, 3\}$ t_j^i's are disjoint, the sets T^i's are independent. Hence, by applying standard Chernoff-Hoeffding bound, we get that with probability at least $1 - n^{-1}$, all but at most $e^{-2\log^2(\log n)}r + \sqrt{r\log r}$ many blocks satisfy, $\mathrm{ED}(y^i, z^i) \le 1755\epsilon \cdot \texttt{OPT}(T^i)$.

Let s' denote the string $y^1 \odot \cdots \odot y^r$. Note as s' is a subsequence of s,

$$\mathrm{ED}(s, s') = |s| - |s'| \le \frac{500n}{p}\log^{1/2} n$$

Then,

$$\begin{aligned}\mathrm{ED}(s, z) &\le \mathrm{ED}(s', z) + \frac{500n}{p\log^{1/2} n} \\ &\le \sum_{i=1}^{r} \mathrm{ED}(y^i, z^i) + \frac{500n}{p\log^{1/2} n} \\ &\le 1755\epsilon \sum_{i=1}^{r} \texttt{OPT}(T^i) + (e^{-2\log^2(\log n)}r + \sqrt{r\log r})2\log^2 n + \frac{500n}{p\log^{1/2} n} \\ &\le 5270\epsilon pn.\end{aligned}$$

The first inequality follows by triangular inequality and the last inequality follows by (8) and $r = \Theta(n/\log^2 n)$. ◀

Running time analysis. Partitioning the string s_1 into r blocks clearly takes linear time. The main challenge here is to find the best match y_j^i (for $j \in \{2, 3\}$) for each block y_1^i. To do this, for each $j \in \{2, 3\}$, we start with the first $10\log^2 n$-sized substring of s_j and run the approximate pattern matching algorithm under the edit metric by [36, 22] to find the best match y_j^1 for y_1^1 (which takes $O(\log^4 n)$ time). Next, we consider the $10\log^2 n$-sized substring of s_j starting from the end index of y_j^1, and in a similar way find the best match y_j^2 for y_1^2. We continue until we find the best matches for all the blocks $y_1^1, \cdots, y_1^r$. Lemma 5 ensures that y_j^1 indeed lies on the first $10\log^2 n$-sized substring of s_j with probability at least $1 - n^{-4}$. Then Corollary 23 together with Lemma 5 guarantees that to find the best match for a block y_1^i, it suffices to look into the $10\log^2 n$-sized substring of s_j after the best match of the previous block y_1^{i-1}. Hence, we can identify the best matches for all the blocks $y_1^1, \cdots, y_1^r$ in time $\tilde{O}(n)$ (since $r = \Theta(\frac{n}{\log^2 n})$). Once we get s_1^i, s_2^i, s_3^i for each $i \in [r]$, we can compute their median using the dynamic programming algorithm [53, 34] in time $O(\log^6 n)$ time. So, the total running time is $\tilde{O}(n)$.