Efficient Average-Case Population Recovery in the Presence of Insertions and Deletions

Frank Ban
UC Berkeley, Berkeley, CA, USA
fban@berkeley.edu

Xi Chen
Columbia University, New York, NY, USA
http://www.cs.columbia.edu/~xichen
xichen@cs.columbia.edu

Rocco A. Servedio
Columbia University, New York, NY, USA
http://www.cs.columbia.edu/~rocco
rocco@cs.columbia.edu

Sandip Sinha
Columbia University, New York, NY, USA
https://sites.google.com/view/sandips
sandip@cs.columbia.edu

─── **Abstract** ───

A number of recent works have considered the *trace reconstruction problem*, in which an unknown source string $x \in \{0,1\}^n$ is transmitted through a probabilistic channel which may randomly delete coordinates or insert random bits, resulting in a *trace* of x. The goal is to reconstruct the original string x from independent traces of x. While the asymptotically best algorithms known for worst-case strings use $\exp(O(n^{1/3}))$ traces [8, 21], several highly efficient algorithms are known [23, 13] for the *average-case* version of the problem, in which the source string x is chosen uniformly at random from $\{0,1\}^n$. In this paper we consider a generalization of the above-described average-case trace reconstruction problem, which we call *average-case population recovery in the presence of insertions and deletions*. In this problem, rather than a single unknown source string there is an unknown distribution over s unknown source strings $x^1, \dots, x^s \in \{0,1\}^n$, and each sample given to the algorithm is independently generated by drawing some x^i from this distribution and returning an independent trace of x^i. Building on the results of [23] and [13], we give an efficient algorithm for the average-case population recovery problem in the presence of insertions and deletions. For any support size $1 \le s \le \exp(\Theta(n^{1/3}))$, for a $1 - o(1)$ fraction of all s-element support sets $\{x^1, \dots, x^s\} \subset \{0,1\}^n$, for every distribution $\mathcal{D}$ supported on $\{x^1, \dots, x^s\}$, our algorithm can efficiently recover $\mathcal{D}$ up to total variation distance at most ε with high probability, given access to independent traces of independent draws from $\mathcal{D}$. The running time of our algorithm is $\mathrm{poly}(n, s, 1/\varepsilon)$ and its sample complexity is $\mathrm{poly}(s, 1/\varepsilon, \exp(\log^{1/3} n))$. This polynomial dependence on the support size s is in sharp contrast with the *worst-case* version of the problem (when $x^1, \dots, x^s$ may be any strings in $\{0,1\}^n$), in which the sample complexity of the most efficient known algorithm [3] is doubly exponential in s.

1   Introduction

Background: Worst-case and average-case trace reconstruction.   In the problem of *trace reconstruction in the presence of insertions and deletions*, there is an unknown and arbitrary n-bit source string $x \in \{0,1\}^n$ and the goal is to reconstruct x given access to independent *traces* of x. A *trace* of x is a copy that has been passed through a noise channel which independently removes each bit of x with some probability q (the *deletion rate*) and also independently inserts random bits according to some *insertion rate* q'.[1] Intuitively, the insertion-deletion channel (or even just the deletion channel with no insertions) is challenging to deal with because it is difficult to determine which coordinate of the source string (if any, if insertions are possible) is responsible for a given coordinate of a received trace.

The insertion / deletion trace reconstruction problem is motivated by connections to recovery problems arising in biology (see e.g. [2, 6, 1]) and has been the subject of considerable research, especially in recent years. The worst-case version of this problem, in which the source string x can be an arbitrary element of $\{0,1\}^n$, appears to be quite difficult even for small constant noise rates. In early work [4] gave an efficient algorithm that succeeds in the deletion-only model if the deletion rate q is quite low, at most $O(1/n^{1/2+\varepsilon})$. Also in the deletion-only model, [14] showed that $\exp(\tilde{O}(\sqrt{n}))$ many traces suffice for any constant deletion rate q bounded away from 1. More recently, this result was improved in simultaneous and independent works of [8] and [21], each of which showed that for any constant insertion and deletion rates q, q', $\exp(O(n^{1/3}))$ traces suffice to reconstruct any $x \in \{0,1\}^n$. These algorithms, which run in $\exp(O(n^{1/3}))$ time, give the best results to date for the worst-case problem. (On the lower bound side, recent work of [12] obtained an $\tilde{\Omega}(n^{5/4})$ lower bound on the number of traces required from the deletion channel, improving an earlier $\Omega(n)$ lower bound due to [19]. Later work of [5] improved this lower bound to $\tilde{\Omega}(n^{3/2})$.)

Since the worst-case trace reconstruction problem seems to be quite difficult, and since the assumption that the source string x is completely arbitrary may be overly pessimistic in various contexts, it is natural to consider an *average-case* version of the problem in which the source string x is assumed to be drawn uniformly at random from $\{0,1\}^n$. This average-case problem has been intensively studied, and interestingly it turns out to be significantly easier than the worst-case problem. [4] showed that for most source strings x, in the deletion-only setting only $O(\log n)$ traces suffice for deletion rates q as large as $O(1/\log n)$. [16] considered the insertion / deletion noise channel and obtained an $O(\log n)$-trace average-case algorithm for noise rates up to $O(1/\log^2 n)$, which was later improved to $O(1/\log n)$ in [25]. [14] were the first to give an efficient (using poly(n) traces) average-case algorithm, for the deletion-only model, that succeeds for some constant deletion rate (their algorithm could handle deletion rates up to about $q = 0.07$). Building on the worst-case results of [8] and [21], a number of significantly stronger average-case results have recently been established. [23] gave an average-case algorithm for the deletion-only problem which uses $\exp(O(\log^{1/2} n))$ many traces for any deletion rate $q < 1/2$. Improving on this, [13] gave an average-case algorithm which can handle both insertions and deletions at any constant rate and uses only $\exp(O(\log^{1/3} n))$ many traces. (A simple reduction shows that any improvement on this sample complexity for the average-case problem would imply an improvement of the $\exp(O(n^{1/3}))$-trace worst-case algorithms of [8] and [21].)

---

[1] A detailed description of the channel is given in Section 2. Augmented variants of this insertion / deletion noise model can also be considered, for example allowing for bit-flips as well as insertions and deletions, but unlike deletions and insertions bit-flips can typically be handled in a straightforward fashion. In this paper we confine our attention to the insertion / deletion channel.

Beyond trace reconstruction: Population recovery from the deletion channel.   Inspired by a related problem known as *population recovery*, recent work of [3] has considered a challenging extension of the trace reconstruction problem. Population recovery is the problem of learning an unknown *distribution* over an unknown set of n-bit strings, given access to independent draws from the distribution that have been independently corrupted according to some noise channel. Most research in population recovery has focused on two noise models, namely the *bit-flip* noise channel (in which each coordinate is independently flipped with some fixed probability) and the *erasure* noise channel (in which each coordinate is independently replaced by '?' with some fixed probability), both of which have been intensively studied, see e.g. [10, 26, 24, 9, 20, 18, 7, 9]. [3] considered the problem of *population recovery from the deletion channel*. This is a generalization of the deletion-channel trace reconstruction problem: now there is an unknown distribution over s unknown source strings $x^1, \dots, x^s \in \{0,1\}^n$, and each sample provided to the learner is obtained by first drawing a string x^i from this distribution and then passing it through the deletion noise channel. It is clear that this problem is at least as difficult as the trace reconstruction problem (which is the $s = 1$ case), and indeed having multiple source strings turns out to pose significant new challenges. [3] considered the worst-case version of this problem, and showed that any distribution $\mathcal{D}$ over any set of s unknown source strings can be recovered to total variation distance ε given $2^{\sqrt{n} \cdot (\log n)^{O(s)}} / \varepsilon^2$ many traces from the deletion channel. [3] also gave a lower bound, showing that for any $s \le n^{0.49}$ at least $n^{\Omega(s)}$ many traces are required. Population recovery-type problems have also been studied in the computational biology literature, specifically for DNA storage (see e.g. [22, 27]). In these settings, the population of strings corresponds to a collection of DNA sequences.

Summarizing, while population recovery from the deletion channel is a natural problem, the above results (and the fact that it is at least as difficult as trace reconstruction) indicate that it is also a hard one. Thus it is natural to investigate *average-case* versions of this problem; this is the subject of the current work.

1.1   Our result: Average-case population recovery in the insertion / deletion model

In the average-case model we consider, there is a given *population size* $s \ge 1$, i.e. there is a set $x^1, \dots, x^s$ of s strings which are assumed to be drawn independently and uniformly from $\{0,1\}^n$. Associated with this population is an *arbitrary* vector of non-negative probability values $p_1, \dots, p_s$, where p_i is the probability that the distribution $\mathcal{D}$ puts on string x^i. Thus in our model the support of the distribution is "average-case" but the actual distribution over that support is "worst-case."

Building on the work of [13], our main result is a highly efficient algorithm for average-case population recovery in the presence of insertions and deletions. We show that even for extremely large population sizes s (up to $\exp(\Theta(n^{1/3}))$), the average-case population recovery problem can be solved by a highly efficient algorithm which has running time polynomial in n (the length of unknown strings), s (the population size), and $1/\varepsilon$ (where ε is the total variation distance between $\mathcal{D}$ and the distribution returned by the algorithm). The sample complexity of our algorithm is polynomial in s, $1/\varepsilon$, and $\exp(\log^{1/3} n)$. Thus our algorithm extends the average-case trace reconstruction results of [13] to the more challenging setting of s-string population recovery with essentially the best possible dependence on the new parameters s and $1/\varepsilon$ (which are not present in the original trace reconstruction problem but are inherent in the population recovery problem).

In more detail, we prove the following theorem (the exact definition of a random trace drawn from the insertion / deletion noise channel $\mathcal{C}_{q,q'}(\mathcal{D})$ will be given in Section 2):

▶ **Theorem 1.** *Fix any two constants $q, q' \in [0, 1)$ as* deletion *and* insertion rates, *respectively. There is an algorithm A with the following property: Let $\delta_{hard} \geq \exp(-\Theta(n^{1/3}))$ be a fraction of hard support sets, let $\delta_{fail} \geq \exp(-\Theta(n^{1/3}))$ be a failure probability, let $\varepsilon \geq \exp(-\Theta(n^{1/3}))$ be an accuracy parameter, let $1 \leq s \leq \exp(\Theta(n^{1/3}))$ be a support size, and let $x^1, \ldots, x^s$ be a support set (viewed as an ordered list of strings in $\{0, 1\}^n$). For at least a $(1 - \delta_{hard})$-fraction of all 2^{ns} many possible s-element support sets, it is the case that for any probability distribution $\mathcal{D}$ supported on $\{x^1, \ldots, x^s\}$, given $n, s, \varepsilon, \delta_{hard}, \delta_{fail}$, and access to $\mathcal{C}_{q,q'}(\mathcal{D})$, algorithm A uses $\mathrm{poly}(s, 1/\varepsilon, \exp(\log^{1/3} n), \exp(\log^{1/3}(1/\delta_{hard})), \log(1/\delta_{fail}))$ random traces from $\mathcal{C}_{q,q'}(\mathcal{D})$, runs in time $\mathrm{poly}(n, s, 1/\varepsilon, 1/\delta_{hard}, \log(1/\delta_{fail}))$ and has the following property: with probability at least $1 - \delta_{fail}$ it outputs a hypothesis distribution $\mathcal{D}'$ over $\{0, 1\}^n$ such that $d_{\mathrm{TV}}(\mathcal{D}, \mathcal{D}') \leq \varepsilon$.*

Discussion. Taken together with the recent results of [3], Theorem 1 shows that the average-case and worse-case versions of population recovery in the presence of insertions and deletions have dramatically different complexities. The best known algorithm for the population size-s worst-case population recovery problem [3] has a doubly exponential dependence on s; even for s constant this sample complexity is significantly worse than the best known sample complexity for the $s = 1$ worst-case trace reconstruction problem, which is $\exp(\Theta(n^{1/3}))$ by [8, 21]. The $n^{\Omega(s)}$ sample complexity lower bound given in [3] shows that an exponential dependence on s is inherent for worst-case population recovery. In contrast, Theorem 1 shows that a *polynomial* sample complexity (and running time) dependence on s is achievable for the average-case problem, and that passing from $s = 1$ to larger values of s does not incur much increase in complexity for the average-case problem.

In independent work, [17] studied a different generalization of trace reconstruction which they called *matrix reconstruction*. Instead of reconstructing a string by sampling traces where each character of the string has some probability of being deleted, the goal in *matrix reconstruction* is to reconstruct a matrix by sampling traces where each row and column of the matrix has some probability of being deleted. They used similar techniques to those used in this paper.

1.2 Our techniques

A natural way to approach our problem is to attempt to reduce it to the $s = 1$ case, which as described above is just the average-case trace reconstruction problem which was solved by [13]. However, two challenges arise in carrying out such a reduction. The first challenge is that the analysis of [13] only gives an algorithm which succeeds on a $1 - \Theta(1/n)$ fraction of all source strings $x \in \{0, 1\}^n$. So if the population size s is much larger than n, then a random population of s source strings will with high probability contain $\Theta(s/n)$ many "hard-to-reconstruct" strings. It is not clear how to proceed if these hard-to-reconstruct strings have significant weight under the distribution $\mathcal{D}$ (which they may, since the distribution $\mathcal{D}$ over the s source strings is assumed to be completely arbitrary).

We get around this challenge by showing that for any arbitrarily small δ, the algorithm of [13] can be extended in a black-box way to succeed on any $1 - \delta$ fraction of all source strings $x \in \{0, 1\}^n$ (at the cost of a modest increase in running time and sample complexity depending on the value of δ). By taking $\delta \ll 1/s$, with high probability the random population will consist entirely of source strings x^i each of which could be reconstructed in isolation if we were given only traces coming from x^i.

The second (main) challenge, of course, is that we are not given traces from each individual string x^i in isolation, but rather we are given a mixture of traces from all s strings $x^1, \ldots, x^s$. The main contribution of our work is a clustering procedure which lets us (with high probability over a population of s random source strings) correctly group together traces that came from the same source string. Given the ability to do such clustering, we can indeed use the [13] algorithm on each obtained cluster to identify each of the source strings which has non-negligible weight under the distribution $\mathcal{D}$, and given the identity of these source strings that each trace came from, it is straightforward to output a high-accuracy hypothesis for the unknown distribution $\mathcal{D}$ over these strings.

The core clustering procedure which we develop is a simple algorithm which we call A_{cluster} (see Algorithm 1 in Section 4). This algorithm takes two traces a and b as input, and outputs either "same" or "different." Its performance guarantee is the following: If a and b were generated as two independent traces from *the same* randomly chosen source string $\boldsymbol{x}$, then with high probability A_{cluster} outputs "same," whereas if a, b were generated as two traces coming from *two independent* uniform random source strings $\boldsymbol{x}^1$ and $\boldsymbol{x}^2$ respectively, then with high probability A_{cluster} outputs "different." (See Theorem 5 for a detailed statement.)

The idea underlying A_{cluster} is as follows. Given a trace a (which we view as a string over $\{-1, 1\}$), imagine breaking it up into contiguous segments (which we call "*blocks*") and summing the ± 1 bits within each block, and let $\text{sum}(a, i)$ denote the sum of the bits in the i-th block. We do the same for the trace b and obtaining a value $\text{sum}(b, i)$ from the i-th block of b. The high-level idea is that, for a suitable choice of the block size, in general there will be significant overlap between the positions in $\{1, \ldots, n\}$ (of the source string) that gave rise to the elements of the i-th block of a and the i-th block of b. As a result,

- On the one hand, if a and b came from the same source $\boldsymbol{x}$, then there will be significant cancellation in the difference $\text{sum}(a, i) - \text{sum}(b, i)$ and this difference will tend to be "small" in magnitude.
- On the other hand, if a and b came from independent source strings $\boldsymbol{x}^1$ and $\boldsymbol{x}^2$, then there will be no such cancellation and the difference $\text{sum}(a, i) - \text{sum}(b, i)$ will not be so "small" in magnitude.

Therefore by checking the magnitude of $\text{sum}(a, i) - \text{sum}(b, i)$ across many different blocks i, it is possible to determine with high confidence whether or not a and b came from the same source string or not.

2 Preliminaries

We write $[n] = \{1, \ldots, n\}$ for a positive integer n. We index strings $x \in \{0, 1\}^n$ as $x = (x_1, \ldots, x_n)$. We use **bold font** to denote random variables (which may be real-valued, integer-valued, $\{0, 1\}^*$-valued, etc.).

We consider an insertion-deletion noise channel $\mathcal{C}_{q,q'}$ defined as by [13]. Given a *deletion rate* q and an *insertion rate* q', both in $[0, 1)$, the insertion-deletion channel $\mathcal{C}_{q,q'}$ acts on an $x \in \{0, 1\}^n$ as follows: First, for each $j \in [n]$, $\boldsymbol{G}_j(q') - 1$ many independent and uniform bits from $\{0, 1\}$ are inserted before the j-th bit of x, where $\boldsymbol{G}_1(q'), \ldots, \boldsymbol{G}_n(q')$ are i.i.d. geometric random variables satisfying

$$\mathbf{Pr}\left[\boldsymbol{G}_j(q') = \ell\right] = (q')^{\ell-1}(1 - q')$$

(i.e. each $\boldsymbol{G}_j(q')$ is distributed as Geometric$(1 - q')$). Then each bit of the resulting string is independently deleted with probability q. The resulting string is the output from $\mathcal{C}_{q,q'}(x)$, and we write "$\boldsymbol{y} \sim \mathcal{C}_{q,q'}(x)$" to indicate that $\boldsymbol{y}$ is a random trace generated from x in this way. If $\mathcal{D}$ is a distribution over n-bit strings, we write "$\boldsymbol{y} \sim \mathcal{C}_{q,q'}(\mathcal{D})$" to indicate that $\boldsymbol{y}$ is obtained by first drawing $\boldsymbol{x} \sim \mathcal{D}$ and then drawing $\boldsymbol{y} \sim \mathcal{C}_{q,q'}(\boldsymbol{x})$.

3   **Achieving an arbitrarily small fraction of "hard" strings in average-case trace reconstruction**

Fix any constants $q, q' \in [0, 1)$ as deletion and insertion rates, receptively. We will use asymptotic notation such as $O(\cdot)$ and $\Theta(\cdot)$ to hide constants that depend on q and q'.

The main result of [13] is an algorithm which successfully performs trace reconstruction on at least $(1 - O(1)/n)$-fraction of all n-bit strings (which is $1 - M/n$ for some constant $M = M(q, q')$ that only depends on q and q'). In more detail, their main result is the following:

▶ **Theorem 2.** *Fix any constants $q, q' \in [0, 1)$. There is a deterministic algorithm $A_{average\text{-}case}$ with the following property: It is given (1) a confidence parameter $\delta > 0$, (2) the length n of an unknown string $x \in \{0, 1\}^n$ and (3) access to $\mathcal{C}_{q,q'}(x)$, uses*

$$\exp\left(O\left(\log^{1/3} n\right)\right) \cdot \log(1/\delta) \tag{1}$$

traces drawn from $\mathcal{C}_{q,q'}(x)$, and runs in time $\mathrm{poly}(n, \log(1/\delta))$. For at least $(1 - O(1/n))$-fraction of all strings $x \in \{0, 1\}^n$,[2] it is the case that, algorithm $A_{average\text{-}case}(\delta, n, \mathcal{C}_{q,q'}(x))$ outputs the string x with probability at least $1 - \delta$ (over the randomness of traces drawn from $\mathcal{C}_{q,q'}(x)$).

Note that in the above theorem the fraction of "hard" strings $x \in \{0, 1\}^n$ on which the [13] algorithm does not succeed is $\Theta(1/n)$. In our setting, to achieve results for general population sizes s, we may require the fraction of "hard" strings on which the reconstruction algorithm does not succeed to be smaller than this; to see this, suppose for example that we are considering a population of size $s = n^2$. If a $\Theta(1/n)$ fraction of strings are "hard" and n^2 strings are chosen uniformly at random to form the support of our distribution $\mathcal{D}$, then we would expect $\Theta(n)$ many hard strings to be present in the support set (i.e. the population) of n^2 strings. If the unknown distribution over the n^2 strings (which, recall, may be any distribution over that support) puts a significant amount of its probability mass on these hard strings, then it may not be possible to successfully recover the population.

In this section we show that the fraction of strings in $\{0, 1\}^n$ that are "hard" can be driven down from $\Theta(1/n)$ to an arbitrarily small fraction in the [13] result, at the cost of a corresponding modest increase in the sample complexity and running time of the algorithm. (As suggested by the discussion given above, such an extension is crucial for us to be able to handle populations of size $s = \omega(n)$.) It may be possible to verify this directly via a careful reworking of the [13] proof, but that proof is involved and such a verification would be quite tedious. Instead we give a simple and direct argument which uses Theorem 2 in a black-box way to prove the following generalization of it, in which only an arbitrarily small fraction of strings are hard to reconstruct:

▶ **Theorem 3.** *Fix any constants $q, q' \in [0, 1)$. There is a deterministic algorithm $A'_{average\text{-}case}$ with the following property: It is given (1) $\tau > 0$ as the desired fraction of hard strings, (2) a confidence parameter δ, (3) the length n of the unknown string $x \in \{0, 1\}^n$ and (4) access to $\mathcal{C}_{q,q'}(x)$. It uses*

$$\exp\left(O\left(\left(\log \max\{n, 1/\tau\}\right)^{1/3}\right)\right) \cdot \log(1/\delta)$$

---

[2] Theorem 1 of [13] only claims a $1 - o_n(1)$ fraction of strings x, but the proof shows that the fraction is $1 - O_{q,q'}(1)/n$; see e.g. the discussion at the beginning of Section 1.3 of [13].

many traces drawn from $\mathcal{C}_{q,q'}(x)$, and runs in time $\mathrm{poly}(\max\{n, 1/\tau\}, \log(1/\delta))$. *For at least* $1 - \tau$ *fraction of all strings* $x \in \{0,1\}^n$, *it is the case that algorithm* $A'_{average\text{-}case}(\tau, \delta, n, \mathcal{C}_{q,q'}(x))$ *outputs the string* x *with probability at least* $1 - \delta$.

We note that the sample complexity of Theorem 3 interpolates smoothly between the average-case result of [13], in which a $\tau = \Theta(1/n)$ fraction of strings are hard, and the worst-case results of [8, 21], in which no strings in $\{0,1\}^n$ (equivalently, at most a $\tau = 1/2^{n+1}$ fraction of strings) are hard.

The high-level idea underlying Theorem 3 is very simple: By padding the input string x (which should be thought of as uniformly random over $\{0,1\}^n$) with random bits, it is possible to obtain a uniformly random N-bit string, and by running algorithm $A_{\text{average-case}}$ over this string of length N, with probability $1 - \Theta(1/N)$ it is possible to reconstruct this N-bit string, from which the original input string x can be reconstructed. Taking N to be suitably large this yields the desired result. We give a detailed proof below.

Let M be the constant hidden in the $O(1/n)$ in Theorem 2. We note that if $\tau \geq M/n$ then we may simply use $A_{\text{average-case}}$, so we henceforth assume that $\tau < M/n$.

The algorithm. Algorithm $A'_{\text{average-case}}(\tau, \delta, n, \mathcal{C}_{q,q'}(x))$ works by running an auxiliary algorithm $A^*(\tau, n, \mathcal{C}_{q,q'}(x))$ (which always outputs an n-bit string) $O(\log(1/\delta))$ many times. If at least $9/16$ of the $O(\log(1/\delta))$ runs of A^* yield the same n-bit string then this is the output of $A'_{\text{average-case}}$, and otherwise $A'_{\text{average-case}}$ outputs "failure." Below we will show that for at least $1 - \tau$ fraction of all strings $x \in \{0,1\}^n$, $A^*(\tau, n, \mathcal{C}_{q,q'}(x))$ outputs the correct string x with probability at least $5/8$. It follows from the Chernoff bound that $A'_{\text{average-case}}$ achieves the desired $1 - \delta$ success probability.

We turn to describing and analyzing $A^*(\tau, n, \mathcal{C}_{q,q'}(x))$, which works as follows:

1. It draws a string $\boldsymbol{z}$ uniformly from $\{0,1\}^{N-n}$ (the value of $N > n$ will be specified later).
2. Let $m = m(N)$ be the following parameter:

$$m(N) = \exp\left(O\left(\log^{1/3} N\right)\right) \cdot \log\left(1/\delta'\right),$$

where $\delta' = 1/8$. This is the number of traces needed by $A_{\text{average-case}}$ to achieve confidence parameter δ' on strings of length N (as in (1)). For m times, algorithm A^* independently repeats the following: at the i-th repetition it draws a string $\boldsymbol{y}^{(i)} \sim \mathcal{C}_{q,q'}(x)$, constructs a string $\boldsymbol{y}'^{(i)}$ that is distributed according to $\mathcal{C}_{q,q'}(\boldsymbol{z})$, and constructs $\boldsymbol{a}^{(i)} := \boldsymbol{y}^{(i)} \circ \boldsymbol{y}'^{(i)}$ which is the concatenation of $\boldsymbol{y}^{(i)}$ and $\boldsymbol{y}'^{(i)}$.
3. Finally, it uses the m strings $\boldsymbol{a}^{(1)}, \ldots, \boldsymbol{a}^{(m)}$ to run algorithm $A_{\text{average-case}}$ with length N and confidence parameter δ'. Let $w \in \{0,1\}^N$ be the string that $A_{\text{average-case}}$ returns. The output of A^* is $w_1 w_2 \cdots w_n$, the first n characters of w.

Proof of correctness. We first observe that (as an immediate consequence of the definition of the noise channel $\mathcal{C}_{q,q'}$) each string $\boldsymbol{a}^{(i)} = \boldsymbol{y}^{(i)} \circ \boldsymbol{y}'^{(i)}$ generated as in Step 2 of $A^*(\tau, n, \mathcal{C}_{q,q'}(x))$ is distributed precisely as a draw from $\mathcal{C}_{q,q'}(x \circ \boldsymbol{z})$. By the choice of $m = m(N)$ in Step 2, the strings $\boldsymbol{a}^{(1)}, \ldots, \boldsymbol{a}^{(m)}$ constitute precisely the required traces for a run of $A_{\text{average-case}}$ on the N-bit string $x \circ \boldsymbol{z}$.

Let us say that the strings in $\{0,1\}^N$ which $A_{\text{average-case}}$ (with parameters δ' and N) correctly reconstructs with probability at least $1 - \delta'$ are *good* strings, and that the other strings in $\{0,1\}^N$ are *bad* strings. By Theorem 2, at most an (M/N)-fraction of all strings in $\{0,1\}^N$ are bad. The value of N is set to $N := 4M/\tau$,[3] so $M/N = \tau/4$, and it is the case

---

[3] Note that $N \geq 4n$ using $\tau < M/N$, so $N - n > 0$ and indeed Step 1 makes sense.

that at most a $\tau/4$ fraction of strings in $\{0,1\}^N$ are bad. For each $x \in \{0,1\}^n$, let γ_x denote the fraction of strings $z \in \{0,1\}^{N-n}$ such that $x \circ z$ is bad. The average over all $x \in \{0,1\}^n$ of γ_x is at most $\tau/4$, and consequently at most a τ fraction of strings x have $\gamma_x \geq 1/4$.

$\triangleright$ Claim 4.   If $x \in \{0,1\}^n$ has $\gamma_x < 1/4$, then $A^*(\tau, n, \mathcal{C}_{q,q'}(x))$ outputs x with probability at least $5/8$.

Proof. The probability that $z \sim \{0,1\}^{N-n}$ such that $x \circ z$ is good is at least $3/4$. If $x \circ z$ is good then with probability at least $1 - \delta' = 7/8$ the output of $A_{\text{average-case}}$ as run in Step 3 is the string $x \circ z$ and hence the output of A^* is x. The claim follows since $(3/4) \cdot (7/8) > 5/8$.
$\triangleleft$

Hence for at least a $(1-\tau)$-fraction of $x \in \{0,1\}^n$, a run of $A^*(\tau, n, \mathcal{C}_{q,q'}(x))$ outputs x with probability at least $5/8$. For any such x, a simple Chernoff bound shows that with probability at least $1 - \delta$, at least $9/16$ of the $O(\log(1/\delta))$ many independent runs of A^* will output x. This concludes the proof of Theorem 3.
◀

4   The core clustering result

In this section we state and prove the key clustering result that is used in the main algorithm. Intuitively, it gives an efficient procedure with the following performance guarantee: Given two traces, the procedure can determine with high probability whether the two traces were both obtained as traces from the same uniform random string $\boldsymbol{x} \sim \{0,1\}^n$, or the two traces were obtained from two independent uniform random strings $\boldsymbol{x}^1, \boldsymbol{x}^2 \sim \{0,1\}^n$.

In more detail, the main result of this section is the following theorem:

▶ **Theorem 5.** *Fix any constants $q, q' \in [0,1)$. There is a deterministic algorithm $A_{cluster}$ with the following performance guarantee: It is given a positive integer n and a pair of binary strings z and z'. Let $\delta_{cluster} := \exp(-\Theta(n^{1/3}))$. Then $A_{cluster}(n, z, z')$ runs in time $O(n)$ and satisfies the following two properties:*

1. *Suppose that $\boldsymbol{x}$ is uniform random over $\{0,1\}^n$ and $\boldsymbol{z}, \boldsymbol{z}'$ are independent draws from $\mathcal{C}_{q,q'}(\boldsymbol{x})$. Then with probability at least $1 - \delta_{cluster}$, algorithm $A_{cluster}(n, \boldsymbol{z}, \boldsymbol{z}')$ outputs "same."*

2. *Suppose that $\boldsymbol{x}^1, \boldsymbol{x}^2$ are independent uniform random strings over $\{0,1\}^n$, $\boldsymbol{z} \sim \mathcal{C}_{q,q'}(\boldsymbol{x}^1)$ and $\boldsymbol{z}' \sim \mathcal{C}_{q,q'}(\boldsymbol{x}^2)$. Then with probability at least $1 - \delta_{cluster}$, $A_{cluster}(n, \boldsymbol{z}, \boldsymbol{z}')$ outputs "different."*

4.1   Proof of Theorem 5

For convenience, we consider strings over $\{-1, 1\}$ instead of $\{0, 1\}$ in the rest of this section. We need the following technical lemma:

▶ **Lemma 6.** *Let $\tau \in (0, 1]$ be a constant. Then there exist three positive constants c_1, c_2 and c_3 (that only depend on τ) such that the following property holds. For all positive integers m and m' such that $m' \leq (1 - \tau)m$ and m is sufficiently large, letting $\mathbf{X}_1, \ldots, \mathbf{X}_m$ be independent and uniform random variables over $\{-1, 1\}$, we have*

$$\boldsymbol{Pr}\left[\left|\mathbf{X}_1 + \cdots + \mathbf{X}_m\right| \geq c_1\sqrt{m}\right] \geq c_2 + c_3 \quad and \quad \boldsymbol{Pr}\left[\left|\mathbf{X}_1 + \cdots + \mathbf{X}_{m'}\right| \geq c_1\sqrt{m}\right] \leq c_2 - c_3.$$

Proof. The Berry-Esseen theorem (see e.g. [11]) establishes closeness between the cdf of a sum of "well-behaved" independent random variables (such as $\mathbf{X}_1, \ldots, \mathbf{X}_m$) and the cdf of a Normal distribution with the same mean and variance. By the Berry-Esseen theorem, the probability of $|\mathbf{X}_1 + \cdots + \mathbf{X}_m| \geq c_1\sqrt{m}$ is within an additive $\pm o_m(1)$ of the corresponding probability of $|\mathbf{G}_1| \geq c_1\sqrt{m}$, where $\mathbf{G}_1 \sim \mathcal{N}(0, m)$.

We first consider the case that m' is not too small compared to m, say $m' > m^{1/3}$. In this case the Berry-Esseen theorem implies that the probability of $|\mathbf{X}_1 + \cdots + \mathbf{X}_{m'}| \geq c_1\sqrt{m}$ is also within an additive $\pm o_m(1)$ of the corresponding probability for Gaussian random variables, which is now $\mathbf{Pr}[|\mathbf{G}_2| \geq c_1\sqrt{m}]$ with $\mathbf{G}_2 \sim \mathcal{N}(0, m')$. So in this case Lemma 6 is an immediate consequence of an analogous statement for Gaussian random variables,

$$\mathbf{Pr}\left[\left|\mathbf{G}_1\right| \geq c_1\sqrt{m}\right] \geq c_2 + c_3 \quad \text{and} \quad \mathbf{Pr}\left[\left|\mathbf{G}_2\right| \geq c_1\sqrt{m}\right] \leq c_2 - c_3, \tag{2}$$

where $\mathbf{G}_1 \sim \mathcal{N}(0, m)$ and $\mathbf{G}_2 \sim \mathcal{N}(0, m')$. The first probability in (2) is the probability that a Gaussian's magnitude exceeds its mean by at least c_1 standard deviations, while the second probability in (2) is the probability that a Gaussian's magnitude exceeds its mean by at least $c_1/\sqrt{1-\tau}$ standard deviations. Given this, for suitable c_1, c_2, c_3 depending only on τ, the inequalities (2) are a straightforward consequence of the following standard bounds on the cdf of a Gaussian $\mathbf{G} \sim \mathcal{N}(0, \sigma^2)$ [[11], Section 7.1]:

$$\left(\frac{1}{x} - \frac{1}{x^3}\right) \cdot \frac{e^{-x^2/2}}{\sqrt{2\pi}} \leq \mathbf{Pr}\left[\mathbf{G} \geq x\sigma\right] \leq \frac{1}{x} \cdot \frac{e^{-x^2/2}}{\sqrt{2\pi}}, \quad \text{for all } x > 0.$$

Finally we consider the case that m' is very small compared to m, say $m' \leq m^{1/3}$. In this case, by the Berry-Esseen theorem we have that $\mathbf{Pr}[|\mathbf{X}_1 + \cdots + \mathbf{X}_m| \geq c_1\sqrt{m}]$ is $\pm o_m(1)$-close to the probability that a Gaussian's magnitude exceeds its mean by c_1 standard deviations, which is at least some absolute constant, while $\mathbf{Pr}[|\mathbf{X}_1 + \cdots + \mathbf{X}_{m'}| \geq c_1\sqrt{m}]$ is zero for sufficiently large m, because $m' \leq m^{1/3} < c_1\sqrt{m}$ for m sufficiently large. This finishes the proof of Lemma 6. ◀

Recall that constants $q, q' \in [0, 1)$ denote the deletion probability and insertion probability respectively. Let $p, p' \in (0, 1]$ be $p = 1 - q$ and $p' = 1 - q'$. Then the expected length of a string drawn from $\mathcal{C}_{q,q'}(x)$ with $x \in \{-1, 1\}^n$ is $np/p' = \alpha n$, where $\alpha := p/p'$ is a positive constant.

Given a string $x \in \{-1, 1\}^n$, we start by describing an equivalent way of drawing $\mathbf{z} \sim \mathcal{C}_{q,q'}(x)$. We say a string r over $[n] \cup \{*\}$ is an n-*pattern* if every $i \in [n]$ appears in r at most once and integers appear in r in ascending order. We write $\mathcal{R}_{n,q,q'}$ to denote the following distribution over n-patterns. To draw $\mathbf{r} \sim \mathcal{R}_{n,q,q'}$ we start with $r^{(0)} = (1, 2, \ldots, n)$. Then for each $j \in [n]$, $\mathbf{G}_j(q') - 1$ many $*$'s are inserted before the j-th entry (with value j) of $r^{(0)}$ to obtain $\mathbf{r}^{(1)}$. Finally each entry of $\mathbf{r}^{(1)}$ is independently deleted with probability q to obtain the final string $\mathbf{r}$. Using $\mathcal{R}_{n,q,q'}$, drawing $\mathbf{z} \sim \mathcal{C}_{q,q'}(x)$ can be done equivalently as follows:

1. Draw an n-pattern $\mathbf{r} \sim \mathcal{R}_{n,q,q'}$.
2. For each index $i \in [n]$ that appears in $\mathbf{r}$, replace it by x_i in $\mathbf{r}$.
3. Replace each $*$ in $\mathbf{r}$ with an independent and uniform draw from $\{-1, 1\}$.

Next we introduce a number of parameters and constants that will be used in the clustering algorithm A_{cluster}. Two parameters $\tilde{s}$ and t used in the algorithm are

$$t := n^{2/3} \quad \text{and} \quad \tilde{s} = \left\lfloor \frac{\alpha n}{4t} \right\rfloor = \Theta(n^{1/3})$$

so that $2\tilde{s}t \leq \alpha n/2$. Three constants β, γ and δ are defined as c_1, c_2 and c_3 in Lemma 6 with τ set to be the following constant in $(0, 1)$: $\tau = 0.7pp'$. For each $\ell \in [\tilde{s}]$, we let I_ℓ denote the following set of integers:

$$I_\ell = \big[(2\ell - 2)t + 1, (2\ell - 1)t\big] \cap \mathbb{Z}. \tag{3}$$

Given a string z over $\{-1, 1\}$ (or an n-pattern r), we will refer to entries z_i of z (or r_i of r) over $i \in I_\ell$ as the ℓ-th *block* of z (or r). So each block consists of t entries and two consecutive blocks are separated by a gap of t entries. Given an n-pattern r and an integer ℓ, we write $B_\ell(r) \subset [n]$ to denote the set of $i \in [n]$ that appears in the ℓ-th block of r.

The algorithm A_{cluster} is described in Algorithm 1. Before stating the key technical lemma (Lemma 7) and using it to prove Theorem 5, we give some intuition for the algorithm A_{cluster}.

■ **Algorithm 1** Description of the clustering algorithm A_{cluster}.

Algorithm $A_{\text{cluster}}(n, z, z')$

Input: A positive integer n and two strings z and z' over $\{-1, 1\}$.

Output: "Same" or "different."
1. For each $\ell \in [\tilde{s}]$, set Z_ℓ to be the sum of z_i over $i \in I_\ell$, with $z_i = 0$ when $i > |z|$.
2. For each $\ell \in [\tilde{s}]$, set Z'_ℓ to be the sum of z'_i over $i \in I_\ell$, with $z'_i = 0$ when $i > |z'|$.
3. Count the number of $\ell \in [\tilde{s}]$ such that $\big|Z_\ell - Z'_\ell\big| \geq \beta\sqrt{2t}$.
4. If the number of such ℓ is at least $\gamma\tilde{s}$, return "different;" otherwise, return "same."

Recall the two cases in Theorem 5. We start with the easier second case, where $\boldsymbol{x}^1$ and $\boldsymbol{x}^2$ are drawn from $\{-1, 1\}^n$ uniformly and independently, $\boldsymbol{z} \sim \mathcal{C}_{q,q'}(\boldsymbol{x}^1)$ and $\boldsymbol{z}' \sim \mathcal{C}_{q,q'}(\boldsymbol{x}^2)$. First it is easy to show (see property (0) of Lemma 7) that $|\boldsymbol{z}|, |\boldsymbol{z}'| \geq \alpha n/2$ with very high probability. When this happens, $\boldsymbol{Z}_\ell$ is the sum of t independent and uniform random variables over $\{-1, 1\}$ and the same holds for $\boldsymbol{Z}'_\ell$. Moreover, $\boldsymbol{Z}_\ell$ and $\boldsymbol{Z}'_\ell$ are independent of each other since $\boldsymbol{x}^1$ and $\boldsymbol{x}^2$ are drawn independently and thus, $\boldsymbol{Z}_\ell - \boldsymbol{Z}'_\ell$ can be equivalently written as the sum of $2t$ independent and uniform variables over $\{-1, 1\}$. Furthermore, the $\tilde{s}$ random variables $\boldsymbol{Z}_\ell - \boldsymbol{Z}'_\ell$ over $\ell \in [\tilde{s}]$ are independent. Thus, it follows from Lemma 6 and our choices of β, γ and δ that the probability of each $|\boldsymbol{Z}_\ell - \boldsymbol{Z}'_\ell| \geq \beta\sqrt{2t}$ is at least $\gamma + \delta$ and with very high probability, the number of such $\ell \in [\tilde{s}]$ is at least $\gamma\tilde{s}$, in which case the algorithm returns "different" as desired.

In the first case of Theorem 5, we draw $\boldsymbol{x}$ from $\{-1, 1\}^n$ uniformly and then draw $\boldsymbol{z}, \boldsymbol{z}'$ from $\mathcal{C}_{q,q'}(\boldsymbol{x})$ independently. Equivalently one can view the process as first drawing two n-patterns $\mathbf{r}$ and $\mathbf{r}'$ independently from $\mathcal{R}_{n,q,q'}$ and $\boldsymbol{x}$ from $\{-1, 1\}^n$. The string $\boldsymbol{z}$ (or $\boldsymbol{z}'$) is then obtained by replacing each $i \in [n]$ in $\mathbf{r}$ (or $\mathbf{r}'$) by $\boldsymbol{x}_i$ and each $*$ by an independent draw from $\{-1, 1\}$. Again we assume that $|\mathbf{r}|, |\mathbf{r}'| \geq \alpha n/2$, which happens with high probability. When this is the case, each of $\boldsymbol{Z}_\ell$ and $\boldsymbol{Z}'_\ell$ for $\ell \in [\tilde{s}]$ remains the sum of t independent uniform random variables over $\{-1, 1\}$. However, when an index $i \in [n]$ appears in the ℓ-th block of both $\mathbf{r}, \mathbf{r}'$, then $\boldsymbol{x}_i$ appears in both sums $\boldsymbol{Z}_\ell, \boldsymbol{Z}'_\ell$ and gets cancelled out in their difference $\boldsymbol{Z}_\ell - \boldsymbol{Z}'_\ell$.

Our main technical lemma shows that with very high probability over draws $\mathbf{r}$ and $\mathbf{r}'$ from $\mathcal{R}_{n,q,q'}$, the following two properties hold: (1) $|B_\ell(\mathbf{r}) \cap B_\ell(\mathbf{r}')| \geq \tau t$ for every $\ell \in [\tilde{s}]$, i.e., there are at least τt many integers that appear in the ℓ-th block of both $\mathbf{r}$ and $\mathbf{r}'$; and (2) No index $i \in [n]$ appears in two different blocks of $\mathbf{r}$ and $\mathbf{r}'$ (i.e., it cannot be the case that both $i \in B_\ell(\mathbf{r})$ and $i \in B_{\ell'}(\mathbf{r}')$ with $\ell \neq \ell'$; intuitively the reason why we leave a gap of t entries between two consecutive blocks is to achieve this property). Fixing such a pair of n-patterns r and r', property (1) implies that each $\boldsymbol{Z}_\ell - \boldsymbol{Z}'_\ell$ can be written as the sum of

at most $(1-\tau)2t$ many independent $\{-1,1\}$-variables; given this, it follows directly from Lemma 6 that the probability of each $|\boldsymbol{Z}_\ell - \boldsymbol{Z}'_\ell| \geq \beta\sqrt{2t}$ is at most $\gamma - \delta$. Furthermore (2) implies that the $\tilde{s}$ variables $\boldsymbol{Z}_\ell - \boldsymbol{Z}'_\ell$ over $\ell \in [\tilde{s}]$ are independent. This lets us easily infer that the number of ℓ such that $|\boldsymbol{Z}_\ell - \boldsymbol{Z}'_\ell| \geq \beta\sqrt{2t}$ is less than $\gamma\tilde{s}$ with very high probability, in which case the algorithm returns "same" as desired.

As discussed above, the main technical lemma we require is as follows:

▶ **Lemma 7.** *Let* $\mathbf{r}, \mathbf{r}'$ *be two n-patterns drawn independently from* $\mathcal{R}_{n,q,q'}$. *Then with probability at least* $1 - \exp(-\Omega(n^{1/3}))$, *the following three properties all hold:*
(0): $|\mathbf{r}|, |\mathbf{r}'| \geq \alpha n/2$.
(1): $\left| B_\ell(\mathbf{r}) \cap B_\ell(\mathbf{r}') \right| \geq \tau t$ *for all* $\ell \in [\tilde{s}]$.
(2): *If an* $i \in [n]$ *appears in both* $B_\ell(\mathbf{r})$ *and* $B_{\ell'}(\mathbf{r}')$ *for some* $\ell, \ell' \in [\tilde{s}]$, *then we have* $\ell = \ell'$.

The detailed proof of Lemma 7 is given in Appendix A; here we give some intuition.

To prove Lemma 7, we show that $\mathbf{r}, \mathbf{r}' \sim \mathcal{R}_{n,q,q'}$ satisfy each of the three properties with probability at least $1 - \exp(-\Omega(n^{1/3}))$; the lemma follows from a union bound. Property (0) follows from tail bounds on sums of independent Geometric random variables and from standard Chernoff bounds (see Claim 12) for insertions and deletions, respectively. Indeed property (0) holds with probability $1 - \exp(-\Omega(n))$.

Properties (1) and (2) follow from Lemma 13 in Appendix A. To state the lemma, recall that we write $\mathbf{r}^{(1)}$ to denote the string over $[n] \cup \{*\}$ obtained after insertions during the generation of $\mathbf{r} \sim \mathcal{R}_{n,q,q'}$. For each $i \in [n]$, we use $\boldsymbol{Y}_i$ to denote the number of characters before i in $\mathbf{r}^{(1)}$ that survive deletions; note that $\boldsymbol{Y}_i$ is the number of characters that appear before i in $\mathbf{r}$ if i survives in $\mathbf{r}$, but $\boldsymbol{Y}_i$ is well defined even if i was deleted. By definition, we have $\mathbf{E}[\boldsymbol{Y}_i] = ((i/p') - 1)p$. Lemma 13 shows that for constant $c \in (0,1)$, with probability $1 - \exp(-\Omega(n^{1/3}))$, $|\boldsymbol{Y}_i - \mathbf{E}[\boldsymbol{Y}_i]| \leq ct$. Lemma 13 again follows from tail bounds on sums of independent Geometric random variables and from standard Chernoff bounds. We define $\boldsymbol{Y}'_i$ similarly for $\mathbf{r}'$ and the same statement also holds for $\boldsymbol{Y}'_i$.

Property (2) follows directly from Lemma 13, since for an $i \in [n]$ to appear in two different blocks, it must be the case that $|\boldsymbol{Y}_i - \boldsymbol{Y}'_i| \geq t$ and thus, either $|\boldsymbol{Y}_i - \mathbf{E}[\boldsymbol{Y}_i]| \geq t/2$ or $|\boldsymbol{Y}'_i - \mathbf{E}[\boldsymbol{Y}'_i]| \geq t/2$ (as we have $\mathbf{E}[\boldsymbol{Y}'_i] = \mathbf{E}[\boldsymbol{Y}_i]$), which happens with probability at most $\exp(-\Omega(n^{1/3}))$ by Lemma 13.

To prove property (1) for $\ell \in [\tilde{s}]$, we focus on the following interval of indices in $[n]$:

$$I_\ell^{(0)} := \left[\frac{p'}{p}(2\ell - 1.9)t + 1, \frac{p'}{p}(2\ell - 1.1)t \right] \cap \mathbb{Z},$$

and show that with probability at least $1 - \exp(-\Omega(n^{1/3}))$, we have both
(a) At least $\tau t = 0.7tpp'$ indices in $I_\ell^{(0)}$ survive in both $\mathbf{r}$ and $\mathbf{r}'$; and
(b) Every $i \in I_\ell^{(0)}$ that survives in both $\mathbf{r}$ and $\mathbf{r}'$ lies in both $B_\ell(\mathbf{r})$ and $B_\ell(\mathbf{r}')$.
Item (a) follows from a Chernoff bound: the length of $I_\ell^{(0)}$ is $0.8tp'/p$ and every element survives independently in both strings with probability p^2. Letting i_0, i_1 be the left and right ends of $I_\ell^{(0)}$, item (b) holds when $\boldsymbol{Y}_{i_0}, \boldsymbol{Y}'_{i_0}, \boldsymbol{Y}_{i_1}, \boldsymbol{Y}'_{i_1}$ do not shift too far ($0.1t$) away from their expectations which happens with probability at least $1 - \exp(-\Omega(n^{1/3}))$ by Lemma 13.

Finally we use Lemma 7 to prove Theorem 5:

Proof of Theorem 5. We start with the second case in which $\boldsymbol{x}^1, \boldsymbol{x}^2$ are independent uniform random strings over $\{0,1\}^n$, $\boldsymbol{z} \sim \mathcal{C}_{q,q'}(\boldsymbol{x}^1)$ and $\boldsymbol{z}' \sim \mathcal{C}_{q,q'}(\boldsymbol{x}^2)$. By our discussion earlier, $\boldsymbol{z}$ and $\boldsymbol{z}'$ can be generated equivalently by first drawing $\boldsymbol{r}, \boldsymbol{r}' \sim \mathcal{R}_{n,q,q'}$, then drawing $\boldsymbol{x}^1, \boldsymbol{x}^2$,

and finally deriving z (or z') from r (or r') using x^1 (or x^2) as well as independent random bits for the $*$'s. By Lemma 7, r and r' satisfy all three properties with probability at least $1 - \exp(-\Omega(n^{1/3}))$. Fixing r and r' that satisfy all three properties (for the first case we only need property (0)), we show that $A_{\mathrm{cluster}}(n, z, z')$ returns "different" with probability at least $1 - \exp(-\Omega(n^{1/3}))$ conditioning on $r = r$ and $r' = r'$; the lemma for this case then follows.

To this end, it follows from property (0) that each $Z_\ell - Z'_\ell$ is a sum of $2t$ independent uniform random variables over $\{-1, 1\}$ and thus, each $\ell \in [\tilde{s}]$ satisfies $|Z_\ell - Z'_\ell| \geq \beta\sqrt{2t}$ with probability at least $\gamma + \delta$. Moreover, the $\tilde{s}$ variables $Z_\ell - Z'_\ell$ are independent. It follows from a Chernoff bound (and that δ is a positive constant) that A_{cluster} returns "different" with probability $1 - \exp(-\Omega(\tilde{s})) = 1 - \exp(-\Omega(n^{1/3}))$.

For the second case we can similarly generate z, z' by first drawing $r, r' \sim \mathcal{R}_{n,q,q'}$, then drawing x, and finally deriving z, z' from r, r' using the same x and independent random bits for the $*$'s. Again it follows from Lemma 7 that r, r' satisfy all three properties with probability $1 - \exp(-\Omega(n^{1/3}))$. Fixing r, r' that satisfy all three properties, we show that $A_{\mathrm{cluster}}(n, z, z')$ returns "same" with probability $1 - \exp(-\Omega(n^{1/3}))$, conditioning on $r = r$ and $r' = r'$; the lemma for this case then follows.

For this purpose, properties (0) and (1) imply that each $Z_\ell - Z'_\ell$ is the sum of at most $(1 - \tau)2t$ many independent uniform random variables over $\{-1, 1\}$. Lemma 6 implies that the probability of $|Z_\ell - Z'_\ell| \geq \beta\sqrt{2t}$ is at most $\gamma - \delta$. Moreover, property (2) implies that these $\tilde{s}$ variables $Z_\ell - Z'_\ell$ are independent. It similarly follows from a Chernoff bound that A_{cluster} returns "same" with probability $1 - \exp(-\Omega(\tilde{s})) = 1 - \exp(-\Omega(n^{1/3}))$.   ◄

5   Putting the pieces together: Proof of Theorem 1

In this section we combine the main results from earlier sections, Theorem 3 from Section 3 and Theorem 5 from Section 4, together with standard results on learning discrete distributions, to prove Theorem 1.

5.1   Learning discrete distributions

We recall the following folklore result on learning a discrete distribution from independent samples:

▶ **Theorem 8.** *Fix* $\gamma, \kappa > 0, N \in \mathbb{N}$. *Let* $\mathcal{P}$ *be an unknown probability distribution over the discrete set* $\{1, \ldots, N\}$, *and let* $S = \{i_1, \ldots, i_m\}$ *be independent draws from* $\mathcal{P}$, *where* $m = O((N/\kappa^2) \cdot \log(1/\gamma))$. *Let* $\hat{\mathcal{P}}_S$ *denote the empirical probability distribution over* $[N]$ *corresponding to* S. *Then with probability at least* $1 - \gamma$ *over the draw of* S, *the variation distance* $d_{\mathrm{TV}}(\hat{\mathcal{P}}_S, \mathcal{P})$ *is at most* κ.

We will need a corollary which says that removing low-frequency elements has only a negligible effect:

▶ **Corollary 9.** *Let* $\mathcal{P}, m$ *and* S *be as above. Let* S' *be the subset of* S *obtained by removing each element* j *whose frequency in* S *is at most* $\kappa/(2N)$, *and let* $\hat{\mathcal{P}}_{S'}$ *denote the empirical distribution over* $[N]$ *corresponding to* S'. *Then with probability at least* $1 - \gamma$ *over the draw of* S', $d_{\mathrm{TV}}(\hat{\mathcal{P}}_{S'}, \mathcal{P})$ *is at most* κ.

Proof. By Theorem 8, with probability at least $1 - \delta$ the hypothesis $\hat{\mathcal{P}}_S$ from Theorem 8 is $\kappa/2$-close to $\mathcal{P}$. The corollary follows since the variation distance between $\hat{\mathcal{P}}_S$ and $\hat{\mathcal{P}}_{S'}$ is at most $N \cdot \kappa/(2N) = \kappa/2$.   ◄

Algorithm 2 Description of the main algorithm A.

Algorithm $A(n, s, \varepsilon, \delta_{\mathrm{hard}}, \delta_{\mathrm{fail}}, \mathcal{C}_{q,q'}(\mathcal{D}))$, with constants $q, q' \in [0, 1)$ as deletion and insertion rates.

Input: String length n, support size $s \leq \exp(\Theta(n^{1/3}))$, accuracy parameter $\varepsilon \geq \exp(-\Theta(n^{1/3}))$, fraction of hard support sets $\delta_{\mathrm{hard}} \geq \exp(-\Theta(n^{1/3}))$, failure probability $\delta_{\mathrm{fail}} \geq \exp(-\Theta(n^{1/3}))$, and access to $\mathcal{C}_{q,q'}(\mathcal{D})$ where $\mathcal{D}$ is a probability distribution over s strings in $\{0, 1\}^n$.

Output: Either a probability distribution $\mathcal{D}'$ or "fail."

1. Draw T traces $\boldsymbol{y}^1, \ldots, \boldsymbol{y}^T$ from $\mathcal{C}_{q,q'}(\mathcal{D})$, where

$$T = \frac{s}{\varepsilon^2} \cdot \exp\left(\Theta\left(\left(\log \max\left\{n, \frac{2s}{\delta_{\mathrm{hard}}}\right\}\right)^{1/3}\right)\right) \cdot \log\left(\frac{3s}{\delta_{\mathrm{fail}}}\right).$$

2. For each pair of traces $\boldsymbol{y}^i, \boldsymbol{y}^j$ with $1 \leq i < j \leq T$, run $A_{\mathrm{cluster}}(n, \boldsymbol{y}^i, \boldsymbol{y}^j)$ from Section 4. If the $\binom{T}{2}$-many outcomes of A_{cluster} (corresponding to $\binom{T}{2}$ many answers of "same" or "different") do not correspond to a disjoint union of cliques then halt and output "fail," otherwise continue.

3. Let the resulting clusters / cliques be denoted $C_1, \ldots, C_r$, so $C_1 \sqcup \cdots \sqcup C_r$ is a partition of the set $\{\boldsymbol{y}^1, \ldots, \boldsymbol{y}^T\}$ of traces.[a] Call C_i *large* if it contains at least $T \cdot (\varepsilon/(2s))$ many elements. Let $C'_1, \ldots, C'_{r'}$ denote the large clusters for some $r' \leq r$, and let $C'_{\mathrm{total}} = \sum_i |C'_i|$.

4. For each large multiset C'_i, run $A'_{\mathrm{average\text{-}case}}$ from Section 3 using n and strings from C'_i, in which τ is set to $\delta_{\mathrm{hard}}/(2s)$ and δ is set to $\delta_{\mathrm{fail}}/(3s)$. Let z^i be the output of $A'_{\mathrm{average\text{-}case}}$ on this input.

5. Distribution $\mathcal{D}'$ that A outputs is supported on $z^1, \ldots, z^{r'}$ and puts weight $|C'_i|/C'_{\mathrm{total}}$ on z^i.

---

[a] Strictly speaking, each C_i is a multiset.

5.2 Proof of Theorem 1

Algorithm A is given in Algorithm 2. Its proof of correctness is given below.

Proof. Suppose that the true underlying support of $\mathcal{D}$ is $\mathcal{X} = (x^1, \ldots, x^s)$ (as an ordered list). We consider s instances of algorithm $A'_{\mathrm{average\text{-}case}}$ from Section 3, where each instance has parameters n, $\tau = \delta_{\mathrm{hard}}/(2s)$ and $\delta = \delta_{\mathrm{fail}}/(3s)$, and the i-th one runs on T^* many traces drawn from $\mathcal{C}_{q,q'}(x^i)$, where

$$T^* = \exp\left(\Theta\left(\left(\log \max\left\{n, \frac{2s}{\delta_{\mathrm{hard}}}\right\}\right)^{1/3}\right)\right) \cdot \log\left(\frac{3s}{\delta_{\mathrm{fail}}}\right)$$

as specified in Section 3 (so we have $T = (s/\varepsilon^2) \cdot T^*$). We say that $\mathcal{X}$ is a *hard support* if either

(a) At least one string x^i, $i \in [s]$, is hard for algorithm $A'_{\mathrm{average\text{-}case}}$; or

(b) After drawing T traces from $\mathcal{C}_{q,q'}(x^{(i)})$ for each $i \in [s]$, A_{cluster} fails on one of these $\binom{sT}{2}$ many pairs of traces with probability at least $\delta_{\mathrm{fail}}/3$.

We consider a random support $\boldsymbol{\mathcal{X}} = (\boldsymbol{x}^1, \ldots, \boldsymbol{x}^s)$ drawn from $\{0, 1\}^n$ independently and uniformly. Theorem 3 says the probability of a uniform random string being hard for $A'_{\mathrm{average\text{-}case}}$ is at most $\delta_{\mathrm{hard}}/(2s)$. A union bound says the probability our support

satisfies (a) is at most $\delta_{\text{hard}}/2$. On the other hand, for each support $\mathcal{X}$, we let $\lambda(\mathcal{X})$ denote the probability that A_{cluster} fails on at least one of the $\binom{sT}{2}$ pairs. Theorem 5 implies $\mathbf{E}_{\mathcal{X}}[\lambda(\mathcal{X})] \leq \binom{sT}{2} \cdot \delta_{\text{cluster}} \leq (\delta_{\text{fail}}/3) \cdot \delta_{\text{hard}}/2$, where the last inequality follows by setting the constant hidden in the $\Theta(n^{1/3})$ of upper and lower bounds for $s, \varepsilon, \delta_{\text{hard}}$ and δ_{fail} to be sufficiently small (compared to the constant hidden in δ_{cluster}). By Markov, a random support satisfies (b) with probability at most $\delta_{\text{hard}}/2$. A union bound on (a) and (b) says the probability of a random support being hard is at most δ_{hard}.

If $\mathcal{D}'$ is a probability distribution where $d_{\text{TV}}(\mathcal{D}, \mathcal{D}') \leq \varepsilon$, then we say that $\mathcal{D}'$ is ε-accurate. It suffices to show that for a support $\mathcal{X}$ that is not hard and an arbitrary distribution $\mathcal{D}$ on that support set, the probability that our algorithm A fails to output an ε-accurate distribution $\mathcal{D}'$ is at most δ_{fail}.

Our algorithm has three points of failure. In Step 2, it could fail to cluster the T traces correctly. Given the correct clustering in Step 2, it could fail to learn the underlying string for some cluster in Step 4. Finally, given the correct support, it could fail to output an ε-accurate distribution $\mathcal{D}'$ in Step 5.

By the definition of hard supports we have that Step 2 returns an incorrect clustering with probability at most $\delta_{\text{fail}}/3$. Given a correct clustering in Step 2, each large C_i' will have at least $T \cdot (\varepsilon/2s) = T^*/\varepsilon \geq T^*$ elements. Since no x^i is hard for $A'_{\text{average-case}}$, by Theorem 3 the probability any instance of $A'_{\text{average-case}}$ fails is at most $\delta_{\text{fail}}/(3s)$. By a union bound, the probability of a Step 4 error is at most $\delta_{\text{fail}}/3$.

Since $T \geq \Omega((s/\varepsilon^2) \cdot \log(3/\delta_{\text{fail}}))$ and the large clusters are defined to have size at least a $\varepsilon/2s$ fraction of the number of traces, then by Corollary 9 with $N = s$, $\kappa = \varepsilon$, $\gamma = \delta_{\text{fail}}/3$, and $m = T$, given the correct support the probability that Step 5 fails to output an ε-accurate probability distribution is at most $\delta_{\text{fail}}/3$. By a union bound, the probability of failure on a support that is not hard is at most δ_{fail}.

By Theorem 5 Step 2 takes time $O(nT^2)$. By Theorem 3 Step 4 takes time $\text{poly}(n, s/\delta_{\text{hard}}, \log(1/\delta_{\text{fail}}))$. Step 5 takes time $O(s)$ to compute the weights used in $\mathcal{D}'$. Therefore, the overall running time of the algorithm is $\text{poly}(n, s, 1/\varepsilon, 1/\delta_{\text{hard}}, \log(1/\delta_{\text{fail}}))$. The theorem follows since the sample complexity T is at most $\text{poly}(s, 1/\varepsilon, \exp(\log^{1/3} n)$, $\exp(\log^{1/3}(1/\delta_{\text{hard}})), \log(1/\delta_{\text{fail}}))$. ◀

References

1   Alexandr Andoni, Mark Braverman, and Avinatan Hassidim. Phylogenetic Reconstruction with Insertions and Deletions. Manuscript, 2014.

2   Alexandr Andoni, Constantinos Daskalakis, Avinatan Hassidim, and Sébastien Roch. Global Alignment of Molecular Sequences via Ancestral State Reconstruction. In *ICS*, pages 358–369, 2010.

3   Frank Ban, Xi Chen, Adam Freilich, Rocco A. Servedio, and Sandip Sinha. Beyond trace reconstruction: Population recovery from the deletion channel. *CoRR*, abs/1904.05532, 2019. `arXiv:1904.05532`.

4   T. Batu, S. Kannan, S. Khanna, and A. McGregor. Reconstructing strings from random traces. In *Proceedings of the Fifteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2004*, pages 910–918, 2004.

5   Zachary Chase. New lower bounds for trace reconstruction. *arXiv preprint*, 2019. `arXiv:1905.03031`.

6   Constantinos Daskalakis and Sébastien Roch. Alignment-Free Phylogenetic Reconstruction. In *RECOMB*, pages 123–137, 2010.

7   A. De, M. Saks, and S. Tang. Noisy population recovery in polynomial time. Technical Report TR-16-026, Electronic Colloquium on Computational Complexity, 2016. To appear in FOCS 2016.

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

9   Anindya De, Ryan O'Donnell, and Rocco A. Servedio. Sharp bounds for population recovery. *CoRR*, abs/1703.01474, 2017. `arXiv:1703.01474`.

10 Z. Dvir, A. Rao, A. Wigderson, and A. Yehudayoff. Restriction access. In *Innovations in Theoretical Computer Science*, pages 19–33, 2012.

11 W. Feller. *An introduction to probability theory and its applications*. John Wiley & Sons, 1968.

12 Nina Holden and Russell Lyons. Lower bounds for trace reconstruction. Available at `arXiv:1808.02336`, 2018.

13 Nina Holden, Robin Pemantle, and Yuval Peres. Subpolynomial trace reconstruction for random strings and arbitrary deletion probability. *CoRR*, abs/1801.04783, 2018.

14 T. Holenstein, M. Mitzenmacher, R. Panigrahy, and U. 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, 2008.

15 Svante Janson. Tail bounds for sums of geometric and exponential variables. *Statistics & Probability Letters*, 135:1–6, 2018. `doi:10.1016/j.spl.2017.11.017`.

16 Sampath Kannan and Andrew McGregor. More on Reconstructing Strings from Random Traces: Insertions and Deletions. In *IEEE International Symposium on Information Theory*, pages 297–301, 2005.

17 Akshay Krishnamurthy, Arya Mazumdar, Andrew McGregor, and Soumyabrata Pal. Trace Reconstruction: Generalized and Parameterized. *arXiv preprint*, 2019. `arXiv:1904.09618`.

18 S. Lovett and J. Zhang. Improved Noisy Population Recovery, and Reverse Bonami-Beckner Inequality for Sparse Functions. In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015, Portland, OR, USA, June 14-17, 2015*, pages 137–142, 2015.

19 Andrew McGregor, Eric Price, and Sofya Vorotnikova. Trace Reconstruction Revisited. In *Proceedings of the 22nd Annual European Symposium on Algorithms*, pages 689–700, 2014.

20 Ankur Moitra and Michael E. Saks. A Polynomial Time Algorithm for Lossy Population Recovery. In *54th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2013, 26-29 October, 2013, Berkeley, CA, USA*, pages 110–116, 2013.

21 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, 2017.

22 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.

23 Yuval Peres and Alex Zhai. Average-Case Reconstruction for the Deletion Channel: Subpolynomially Many Traces Suffice. In *FOCS*, pages 228–239, 2017.

24 Yury Polyanskiy, Ananda Theertha Suresh, and Yihong Wu. Sample complexity of population recovery. In *Proceedings of the 30th Conference on Learning Theory, COLT 2017, Amsterdam, The Netherlands, 7-10 July 2017*, pages 1589–1618, 2017.

25 Krishnamurthy Viswanathan and Ram Swaminathan. Improved string reconstruction over insertion-deletion channels. In *Proceedings of the 19th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 399–408, 2008.

26 A. Wigderson and A. Yehudayoff. Population recovery and partial identification. *Machine Learning*, 102(1):29–56, 2016. Preliminary version in FOCS 2012.

27 S.M. Hossein Tabatabaei Yazdi, Ryan Gabrys, and Olgica Milenkovic. Portable and Error-Free DNA-Based Data Storage. *Scientific Reports*, 7(1):5011, 2017.

A   Deferred proof of Lemma 7

We recall Lemma 7:

Lemma 7 (restated).   *Let* $\mathbf{r}, \mathbf{r}'$ *be two n-patterns drawn independently from* $\mathcal{R}_{n,q,q'}$. *Then with probability at least* $1 - \exp(-\Omega(n^{1/3}))$, *the following three properties all hold:*
(0): $|\mathbf{r}|, |\mathbf{r}'| \geq \alpha n/2$.
(1): $|B_\ell(\mathbf{r}) \cap B_\ell(\mathbf{r}')| \geq \tau t$ *for all* $\ell \in [\tilde{s}]$.
(2): *If an* $i \in [n]$ *appears in both* $B_\ell(\mathbf{r})$ *and* $B_{\ell'}(\mathbf{r}')$ *for some* $\ell, \ell' \in [\tilde{s}]$, *then we have* $\ell = \ell'$.

We will use the following tail bounds for sums of independent geometric random variables, which are special cases of results proved by [15].

▶ **Theorem 10** (Theorems 2.1 and 3.1 in [15]). *Let* $p' \in (0, 1]$, *and* $\mathbf{X}_1, \cdots, \mathbf{X}_n$ *be independent* Geometric (p') *random variables. Let* $\mathbf{X} = \sum_{i \in [n]} \mathbf{X}_i$ *and* $\mu = \mathbf{E}[\mathbf{X}] = n/p'$. *Then the following holds:*

1. *For any* $\lambda \geq 1$, *we have*

$$\mathbf{Pr}\left[X \geq \lambda\mu\right] \leq \exp\left(-p'\mu(\lambda - 1 - \ln\lambda)\right).$$

2. *For any* $0 < \lambda \leq 1$, *we have*

$$\mathbf{Pr}\left[X \leq \lambda\mu\right] \leq \exp\left(-p'\mu(\lambda - 1 - \ln\lambda)\right).$$

Note that $\lambda - 1 - \ln\lambda \geq 0$ for all $\lambda > 0$, with equality only at $\lambda = 1$. We first derive a simpler expression for the tail bounds, using the following claim:

▷ **Claim 11.**   Let $f : (-1, \infty) \to \mathbb{R}$ be defined as $f(x) = x - \ln(1 + x)$. The following properties hold: (i) $f(0) = 0$; (ii) $f(x) > x^2/4$ for all $x \in (-1, 1] \setminus \{0\}$; and (iii) $f(x) \geq x/4$ for all $x \geq 1$.

Proof. The claim follows from elementary calculus. For item (ii) it can be shown that $g(x) = f(x) - x^2/4$ attains its minimum value 0 at $x = 0$ and is strictly convex in $(-1, 1]$. For item (iii) it is easy to verify that $h(x) = f(x) - x/4$ satisfies $h'(x) > 0$ for all $x \geq 1$, and hence its minimum value is $h(1) \geq 0.05$.   ◁

Letting $x = \lambda - 1$ in Theorem 10, this claim allows us to replace the $\lambda - 1 - \ln\lambda$ term in the exponent of the tail bounds by either $(\lambda - 1)^2/4$ or $(\lambda - 1)/4$, depending on whether $\lambda < 2$ or $\lambda \geq 2$.

Now, we state and prove a few claims that will be useful for proving Lemma 7. The first claim states that property (0) in Lemma 7 holds with probability at least $1 - \exp(-\Omega(n))$.

▷ **Claim 12.**   With probability at least $1 - \exp(-\Omega(n))$, $\mathbf{r} \sim \mathcal{R}_{n,q,q'}$ satisfies that $|\mathbf{r}^{(1)}| \geq \alpha n/2$.

Proof. Let $\mathbf{r}^{(1)}$ be the random string defined earlier in the generation of $\mathbf{r} \sim \mathcal{R}_{n,q,q'}$. As $|\mathbf{r}^{(1)}|$ is a sum of n independent Geometric (p') random variables, we have $\mu = \mathbf{E}[|\mathbf{r}^{(1)}|] = n/p'$. Invoking Theorem 10 with $\lambda = 3/4$ and Part (1) of Claim 11 with $x = \lambda - 1$, the probability of $|\mathbf{r}^{(1)}| < 3n/(4p')$ is $\exp(-\Omega(n))$.

Fixing any realization $r^{(1)}$ of $\mathbf{r}^{(1)}$ with $|r^{(1)}| \geq 3n/(4p')$, it follows from the standard Chernoff bound that the probability of $|\mathbf{r}| < \alpha n/2 \leq (2p)/3 \cdot |r^{(1)}|$ is at most $\exp(-\Omega(n))$. This finishes the proof.   ◁

Fix an $i \in [n]$. Let $\boldsymbol{Y}_i^*$ be the random variable denoting the number of characters before i in $\mathbf{r}^{(1)}$ (after insertions). Recall that $\boldsymbol{Y}_i$ denotes the number of characters before i in $\mathbf{r}^{(1)}$ that survive deletions; note that $\boldsymbol{Y}_i$ is well-defined even if i is deleted. Then $\mathbf{E}[\boldsymbol{Y}_i^*] = (i/p') - 1$, and $\mathbf{E}[\boldsymbol{Y}_i] = p \cdot \mathbf{E}[\boldsymbol{Y}_i^*] = ((i/p') - 1)p$.

▶ **Lemma 13.** *For any* $i \in [n]$, *the probability that* $\big| \boldsymbol{Y}_i - \mathbf{E}[\boldsymbol{Y}_i] \big| \geq 0.05t$ *is at most* $\exp(-\Omega(n^{1/3}))$.

Proof. Let $\varepsilon = 0.05$ in the proof. We have

$$\big| \boldsymbol{Y}_i - \mathbf{E}[\boldsymbol{Y}_i] \big| \leq \big| \boldsymbol{Y}_i - p\boldsymbol{Y}_i^* \big| + \big| p\boldsymbol{Y}_i^* - p\mathbf{E}[\boldsymbol{Y}_i^*] \big| = \big| \boldsymbol{Y}_i - p\boldsymbol{Y}_i^* \big| + p \cdot \big| \boldsymbol{Y}_i^* - \mathbf{E}[\boldsymbol{Y}_i^*] \big|.$$

We first show that $|\boldsymbol{Y}_i^* - \mathbf{E}[\boldsymbol{Y}_i^*]| \leq \varepsilon t/(2p)$ with probability at least $1 - \exp(-\Omega(n^{1/3}))$. Next conditioning on any fixed realization $r^{(1)}$ of $\mathbf{r}^{(1)}$ with $|Y_i^* - \mathbf{E}[\boldsymbol{Y}_i^*]| \leq \varepsilon t/(2p)$ (in particular this implies that $Y_i^* = O(n)$) we show that $|\boldsymbol{Y}_i - pY_i^*| \leq \varepsilon t/2$ with probability at least $1 - \exp(-\Omega(n^{1/3}))$. The lemma then follows by combining these two steps. Given that the second step follows from the Hoeffding bound (with $Y_i^* = O(n)$ and $t = n^{2/3}$), we focus on the first part in the rest of the proof.

First we analyze the lower tail, i.e., the probability of $\boldsymbol{Y}_i^* - \mathbf{E}[\boldsymbol{Y}_i^*] \leq -\varepsilon t/(2p)$. Because $\boldsymbol{Y}_i^* \geq 0$ we may assume $\mathbf{E}[\boldsymbol{Y}_i^*] > \varepsilon t/(2p)$ (otherwise $\boldsymbol{Y}_i^* \geq \mathbf{E}[\boldsymbol{Y}_i^*] - \varepsilon t/(2p)$ trivially). Let

$$\lambda = 1 - \frac{\varepsilon t}{2p\mathbf{E}[\boldsymbol{Y}_i^*]} \quad \text{and} \quad x = \lambda - 1 = -\frac{\varepsilon t}{2p\mathbf{E}[\boldsymbol{Y}_i^*]},$$

so that $\lambda \mathbf{E}[\boldsymbol{Y}_i^*] = \mathbf{E}[\boldsymbol{Y}_i^*] - \varepsilon t/(2p)$. By Theorem 10 and Part (1) of Claim 11, we have

$$\mathbf{Pr}\left[\boldsymbol{Y}_i^* \leq \mathbf{E}[\boldsymbol{Y}_i^*] - \frac{\varepsilon t}{2p}\right] \leq \exp\left(-\Omega\left(\mathbf{E}[\boldsymbol{Y}_i^*] \cdot \frac{t^2}{\mathbf{E}[\boldsymbol{Y}_i^*]^2}\right)\right) \leq \exp\left(-\Omega\left(\frac{t^2}{n}\right)\right) = \exp\left(-\Omega(n^{1/3})\right).$$

For the second inequality, we used the fact that $\mathbf{E}[\boldsymbol{Y}_i^*] = O(n)$. Similarly, we analyze the upper tail. Let

$$\lambda = 1 + \frac{\varepsilon t}{2p\mathbf{E}[\boldsymbol{Y}_i^*]} \quad \text{and} \quad x = \lambda - 1 = \frac{\varepsilon t}{2p\mathbf{E}[\boldsymbol{Y}_i^*]}.$$

If $\lambda \leq 2$, Theorem 10 and Part (1) of Claim 11 imply that

$$\mathbf{Pr}\left[\boldsymbol{Y}_i^* \geq \mathbf{E}[\boldsymbol{Y}_i^*] + \frac{\varepsilon t}{2p}\right] \leq \exp\left(-\Omega\left(\mathbf{E}[\boldsymbol{Y}_i^*] \cdot \frac{t^2}{\mathbf{E}[\boldsymbol{Y}_i^*]^2}\right)\right) \leq \exp\left(-\Omega\left(\frac{t^2}{n}\right)\right) = \exp\left(-\Omega(n^{1/3})\right).$$

On the other hand, if $\lambda \geq 2$, then $x \geq 1$. By Theorem 10 and Part (2) of Claim 11, we have

$$\mathbf{Pr}\left[\boldsymbol{Y}_i^* \geq \mathbf{E}[\boldsymbol{Y}_i^*] + \frac{\varepsilon t}{2p}\right] \leq \exp\left(-\Omega\left(\mathbf{E}[\boldsymbol{Y}_i^*] \cdot \frac{t}{\mathbf{E}[\boldsymbol{Y}_i^*]}\right)\right) \leq \exp\left(-\Omega(t)\right) = \exp\left(-\Omega(n^{2/3})\right).$$

This finishes the proof of the lemma.   ◀

We are ready to prove Lemma 7.

Proof of Lemma 7. We work on the three events separately and apply a union bound at the end.

(0): It follows from Claim 12 that property (0) holds with probability at least $1 - \exp(-\Omega(n))$.

(1): Fix $\ell \in [\tilde{s}]$. Recall that $I_\ell = [(2\ell - 2)t + 1, (2\ell - 1)t] \cap \mathbb{Z}$. Let

$$I_\ell^{(0)} := \left[\frac{p'}{p}(2\ell - 1.9)t + 1, \frac{p'}{p}(2\ell - 1.1)t \right] \cap \mathbb{Z}.$$

Then $I_\ell^{(0)} \subset [n]$. We will show that with probability at least $1 - \exp(-\Omega(n^{1/3}))$, both properties below hold:

(a) At least $0.7tpp'$ elements in $I_\ell^{(0)}$ survive in both $\mathbf{r}$ and $\mathbf{r}'$;

(b) If an element $i \in I_\ell^{(0)}$ survives in both $\mathbf{r}$ and $\mathbf{r}'$, then $i \in B_\ell(\mathbf{r}) \cap B_\ell(\mathbf{r}')$.

Given that (a) and (b) together imply property (1), we have that property (1) holds for $\ell \in [\tilde{s}]$ with probability at least $1 - \exp(-\Omega(n^{1/3}))$. A union bound over all $\ell \in [\tilde{s}]$ implies that property (1) holds for all $\ell \in [\tilde{s}]$ with probability at least $1 - \tilde{s} \cdot \exp(-\Omega(n^{1/3})) = 1 - \exp(-\Omega(n^{1/3}))$.

So it suffices to show that (a) and (b) happen with probability at least $1 - \exp(-\Omega(n^{1/3}))$. For (a), it follows from a standard Chernoff bound that (a) holds with probability at least $1 - \exp(-\Omega(n^{2/3}))$. For (b), let i_0 and i_1 be the left and right endpoints of $I_\ell^{(0)}$, respectively. Let $\boldsymbol{Y}_{i_0}, \boldsymbol{Y}_{i_1}$ ($\boldsymbol{Y}'_{i,0}, \boldsymbol{Y}'_{i_1}$) be as defined earlier with respect to $\mathbf{r}$ ($\mathbf{r}'$). Note that $\mathbf{E}[\boldsymbol{Y}_{i_0}] = ((i_0/p') - 1)p$ and $\mathbf{E}[\boldsymbol{Y}_{i_1}] = ((i_1/p') - 1)p$. Then by Lemma 13 (and a union bound), with probability at least $1 - 4\exp(-\Omega(n^{1/3}))$, we have:

$$\boldsymbol{Y}_{i_0} \geq \mathbf{E}[\boldsymbol{Y}_{i_0}] - 0.05t > (2\ell - 2)t \quad \text{and} \quad \boldsymbol{Y}_{i_1} \leq \mathbf{E}[\boldsymbol{Y}_{i_1}] + 0.05t < (2\ell - 1)t,$$

and the same holds for $\boldsymbol{Y}'_{i_0}$ and $\boldsymbol{Y}'_{i_1}$. When all these events occur, then clearly all characters in $I_\ell^{(0)}$ that survive in $\mathbf{r}, \mathbf{r}'$ are in I_ℓ in both n-patterns. This finishes the analysis of property (1).

(2): Suppose a character $i \in [n]$ appears in $B_\ell(\mathbf{r})$ and $B_{\ell'}(\mathbf{r}')$ for some $\ell \neq \ell'$. Let $\boldsymbol{Y}_i, \boldsymbol{Y}'_i$ denote the number of characters before i in $\mathbf{r}, \mathbf{r}'$ respectively. Then $\mathbf{E}[\boldsymbol{Y}_i] = \mathbf{E}[\boldsymbol{Y}'_i]$. As any two distinct blocks are separated by at least t positions in the n-patterns, we have $|\boldsymbol{Y}_i - \boldsymbol{Y}'_i| \geq t$. Triangle inequality implies that $|\boldsymbol{Y}_i - \mathbf{E}[\boldsymbol{Y}_i]| \geq t/2$ or $|\boldsymbol{Y}'_i - \mathbf{E}[\boldsymbol{Y}'_i]| \geq t/2$. Assume without loss of generality that $|\boldsymbol{Y}_i - \mathbf{E}[\boldsymbol{Y}_i]| \geq t/2 > 0.05t$. Instantiating Lemma 13, we conclude that this event happens with probability at most $n \cdot \exp(-\Omega(n^{1/3}))$ which remains $\exp(-\Omega(n^{1/3}))$.

The lemma follows from a union bound.   ◄