Permutation Recovery against Deletion Errors for DNA Data Storage

Charchit Gupta[†], **Shubhransh Singhvi**[*], **Avital Boruchovsky**[‡],
Yuval Goldberg[‡], **Han Mao Kiah**[§] and **Eitan Yaakobi**[‡]
[*]Signal Processing & Communications Research Center, IIIT Hyderabad, India
[†]IIIT Hyderabad, India
[‡]Department of Computer Science, Technion—Israel Institute of Technology, Haifa 3200003, Israel
[§]School of Physical and Mathematical Sciences, Nanyang Technological University, Singapore

Abstract

Owing to its immense storage density and durability, DNA has emerged as a promising storage medium. However, due to technological constraints, data can only be written onto many short DNA molecules called *data blocks* that are stored in an unordered way. To handle the unordered nature of DNA data storage systems, a unique *address* is typically prepended to each data block to form a *DNA strand*. However, DNA storage systems are prone to errors and generate multiple noisy copies of each strand called *DNA reads*. Thus, we study the problem of *permutation recovery* for DNA data storage.

The permutation recovery problem for DNA data storage requires one to reconstruct the addresses or in other words to uniquely identify the noisy reads. By successfully reconstructing the addresses, one can essentially determine the correct order of the data blocks, effectively solving the clustering problem.

We first show that we can almost surely identify all the noisy reads under certain mild assumptions. We then propose a permutation recovery procedure and demonstrate that on average the procedure uses only a fraction of $\mathcal{M}^2$ data comparisons(when $\mathcal{M}$ is the number of reads).

I. Introduction

The need for a more durable and compact storage system has become increasingly evident with the explosion of data in modern times. While magnetic and optical disks have been the primary solutions for storing large amounts of data, they still face limitations in terms of storage density and physical space requirements. Storing a zettabyte of data using these traditional technologies would necessitate a vast number of units and considerable physical space.

The idea of using macromolecules for ultra-dense storage systems was recognized as early as the 1960s when physicist Richard Feynman outlined his vision for nanotechnology in his talk 'There is plenty of room at the bottom'. Using DNA is an attractive possibility because it is extremely dense (up to about 1 exabyte per cubic millimeter) and durable (half-life of over 500 years). Since the first experiments conducted by Church et al. in 2012 [**?**] and Goldman et al. in 2013 [**?**], there have been a flurry of experimental demonstrations (see [**?**], [**?**] for a survey). Amongst the various coding design considerations, in this work, we study the unsorted nature of the DNA storage system [**?**], [**?**].

A DNA storage system consists of three important components. The first is the DNA synthesis which produces the oligonucleotides, also called *strands*, that encode the data. The second part is a storage container with compartments which stores the DNA strands, however without order. Finally, to retrieve the data, the DNA is accessed using next-generation sequencing, which results in several noisy copies, called *reads*. The processes of synthesizing, storing, sequencing, and handling strands are all error prone. Due to this unordered nature of DNA-based storage systems, when the user retrieves the information, in addition to decoding the data, the user has to determine the identity of the data stored in each strand. A typical solution is to simply have a set of addresses and store this address information as a prefix to each DNA strand. As the addresses are also known to the user, the user can identify the information after the decoding process. As these addresses along with the stored data are prone to errors, this solution needs further refinements.

In [**?**], the strands (strand = address + data) are first clustered with respect to the edit distance. Then the authors determine a consensus output amongst the strands in each cluster and finally, decode these consensus outputs using a classic concatenation scheme. For this approach, the clustering step is computationally expensive. When there are $\mathcal{M}$ reads, the usual clustering method involves $\mathcal{M}^2$ pairwise comparisons to compute distances. This is costly when the data strands are long, and the problem is further exacerbated if the metric is the edit distance. Therefore, in [**?**], a distributed approximate clustering algorithm was proposed and the authors clustered 5 billion strands in 46 minutes on 24 processors.

In [**?**], the authors proposed and investigated an approach that avoids clustering, by studying a generalisation of the *bee identification problem*. Informally, the bee identification problem requires the receiver to identify M "bees" using a set of M unordered noisy measurements [**?**]. Later, in [**?**], the authors generalized the setup to multi-draw channels where every bee (addresses) results in N noisy outputs (noisy addresses). The task then is to identify each of the M bees from the MN noisy outputs and it turns out that this task can be reduced to a minimum-cost network flow problem. In contrast to previous

works, the approach in [**?**] utilizes only the address information, which is of significantly shorter length, and the method does not take into account the associated noisy data. Hence, this approach involves no data comparisons.

However, as evident, the clustering and bee identification based approaches do not completely take into account the nature of the DNA storage system. In particular, the clustering approaches do not utilise the uncorrupted set of addresses which can be accessed by the receiver and the bee identification approach uses solely the information stored in address and neglects the noisy data strands.

In this work, we overcome these shortcomings by utilising both the address and data information to identify the noisy reads. Specifically for the binary deletion channel, we first show that we can almost surely correctly identify all the reads under certain mild assumptions. Then we propose our permutation recovery procedure and demonstrate that on average the procedure uses only a fraction of $\mathcal{M}^2$ data comparisons (when there are $\mathcal{M}$ reads).

II. Problem Formulation

Let N and M be positive integers. Let $[M]$ denote the set $\{1, 2, \ldots, M\}$. An N-permutation ψ over $[M]$ is an NM-tuple $(\psi(j))_{j\in[MN]}$ where every symbol in $[M]$ appears exactly N times, and we denote the set of all N-permutations over $[M]$ by $\mathbb{S}_N(M)$. Let the set of addresses be denoted by $\mathcal{A} \subseteq \{0,1\}^n$ and $M \triangleq |\mathcal{A}|$. We will use the terms addresses and codewords interchangeably. We assume that every codeword $\boldsymbol{x}_i \in \mathcal{A}$ is attached to a length-L data part $\boldsymbol{d}_i \in \{0,1\}^L$ to form a strand, which is the tuple, $(\boldsymbol{x}_i, \boldsymbol{d}_i)$. Let the multiset of data be denoted by $D = \{\{\boldsymbol{d}_i : i \in [M]\}\}$ and the set of strands by $R = \{(\boldsymbol{x}_i, \boldsymbol{d}_i) : i \in [M]\}$. Throughout this paper, we assume that D is drawn uniformly at random over $\{0,1\}^L$. Let $\mathcal{S}_N((\boldsymbol{x}, \boldsymbol{d}))$ denote the multiset of channel outputs when $(\boldsymbol{x}, \boldsymbol{d})$ is transmitted N times through the channel $\mathcal{S}$. Assume that the entire set R is transmitted through the channel $\mathcal{S}$, hence an unordered multiset, $R' = \{\{(\boldsymbol{x}'_1, \boldsymbol{d}'_1), (\boldsymbol{x}'_2, \boldsymbol{d}'_2), \ldots, (\boldsymbol{x}'_{MN}, \boldsymbol{d}'_{MN})\}\}$, of MN noisy strands (reads) is obtained, where for every $j \in [MN]$, $(\boldsymbol{x}'_j, \boldsymbol{d}'_j) \in \mathcal{S}_N((\boldsymbol{x}_{\pi(j)}, \boldsymbol{d}_{\pi(j)}))$ for some N-permutation π over $[M]$, which will be referred to as the *true N-permutation*. Note that the receiver, apart from the set of reads R', has access to the set of addresses $\mathcal{A}$ but does not know the set of data D.

Let $\boldsymbol{x}, \boldsymbol{y} \in \{0,1\}^*$. We denote the shortest common supersequence between $\boldsymbol{x}, \boldsymbol{y}$ by $\mathsf{SCS}(\boldsymbol{x}, \boldsymbol{y})$ and the longest common subsequence by $\mathsf{LCS}(\boldsymbol{x}, \boldsymbol{y})$.

In [], the authors introduced the following problems:

Problem 1. *Given $\mathcal{S}, \epsilon$ and $\mathcal{A}$, find the region $\mathcal{R} \in \mathbb{Z}_+^2$, such that for $(N, L) \in \mathcal{R}$, it is possible to identify the true permutation with probability at least $1 - \epsilon$ when the data D is drawn uniformly at random.*

For $(N, L) \in \mathcal{R}$, we can find the true permutation by making at least $(NM)^2$ data comparisons. This may be expensive when the data parts are long, i.e., when L is large. Therefore, our second objective is to reduce the number of data comparisons.

Problem 2. *Let $\kappa < 1$. Given $\mathcal{S}, \epsilon, \mathcal{A}$ and $(N, L) \in \mathcal{R}$, design an algorithm to identify the true permutation with probability at least $1 - \epsilon$ using $\kappa(NM)^2$ data comparisons. As before, D is drawn uniformly at random.*

In Section **??**, we demonstrate that the algorithm in [] identifies the true permutation with a vanishing probability as n grows. In Section **??**, we address Problem 1 and identify the region $\mathcal{R}$ for which there exists only one valid permutation, viz. the true permutation. In Section **??**, we describe our algorithm that identifies the true-permutation with probability at least $1 - \epsilon$ when $(N, L) \in \mathcal{R}$. In Section **??**, we analyse the expected number of data comparisons performed by the algorithm.

III. Bee-Identification over Multi-Draw Deletion Channel

The algorithm in [] uses solely the information stored in the addresses to identify the true-permutation, and does not take into consideration the noisy data that is also available to the receiver. The first step in their algorithm is to construct a bipartite graph $\mathcal{G} = (\mathcal{X} \cup \mathcal{Y}, E)$.

(i) Nodes. The left nodes are the addresses ($\mathcal{X} = \mathcal{A}$) and the right nodes are the noisy reads ($\mathcal{Y} = R'$).
(ii) Demands. For each left node / codeword $\boldsymbol{x} \in \mathcal{X}$, we assign the demand $\boldsymbol{\delta}(\boldsymbol{x}) \triangleq -N$, while for each right node / output $(\boldsymbol{y}, \boldsymbol{d}') \in \mathcal{Y}$, we assign the demand $\boldsymbol{\delta}(\boldsymbol{y}) \triangleq 1$.
(iii) Edges. There exists an edge between $\boldsymbol{x} \in \mathcal{X}$ and $(\boldsymbol{y}, \boldsymbol{d}') \in \mathcal{Y}$ if and only if $P(\boldsymbol{y}|\boldsymbol{x}) > 0$, where $P(\boldsymbol{y}|\boldsymbol{x})$ is the likelihood probability of observing $\boldsymbol{y}$ given that $\boldsymbol{x}$ was transmitted. For $\boldsymbol{x} \in \mathcal{X}$ and $(\boldsymbol{y}, \boldsymbol{d}') \in \mathcal{Y}$, let $E_{\boldsymbol{x}}$ and $E_{(\boldsymbol{y},\boldsymbol{d}')}$ denote the multiset of neighbours of $\boldsymbol{x}$ and the set of neighbours of $(\boldsymbol{y}, \boldsymbol{d}')$ in $\mathcal{G}$, respectively, i.e., $E_{\boldsymbol{x}} = \{\{(\boldsymbol{y}, \boldsymbol{d}') | (\boldsymbol{x}, (\boldsymbol{y}, \boldsymbol{d}')) \in E\}\}$, $E_{(\boldsymbol{y},\boldsymbol{d}')} = \{\boldsymbol{x} | (\boldsymbol{x}, (\boldsymbol{y}, \boldsymbol{d}')) \in E\}$. Note that the degree of every left node is at least N as $\mathcal{S}_N((\boldsymbol{x}, \boldsymbol{d})) \subseteq E_{\boldsymbol{x}}$.
(iv) Costs. For an edge $(\boldsymbol{x}, \boldsymbol{y}) \in \mathcal{X} \times \mathcal{Y}$, we assign the cost $\gamma(\boldsymbol{x}, \boldsymbol{y}) = -\log P(\boldsymbol{y} \mid \boldsymbol{x})$. Note that the cost is well-defined as the value $P(\boldsymbol{y} \mid \boldsymbol{x})$ is necessarily positive.

Let us now show that the above algorithm (Chrisnata et al.[]) identifies the true permutation with a vanishing probability as n grows, when $M = 2^n$.

Let $a \in \mathcal{Y}$ and $x \in \mathcal{X}$ and let $|a| = n - k, k \in [0, n]$. Consider the cost of an edge between a and x

$$\text{cost}(x, a) = -\log(P(a \mid x)),$$

For a deletion channel with deletion probability p, we have that

$$P(a \mid x) = \text{Emb}(x, a) \cdot p^k (1-p)^{n-k}.$$

A sufficient condition for the algorithm to fail is the existence of $x, y \in \{0,1\}^n,\ a \in \{0,1\}^k, b \in \{0,1\}^{k'}$ where a, b are noisy reads of x, y, respectively and $0 \leq k, k' \leq n$. Now consider the following event:

$$\begin{aligned} \text{cost}(y, a) < \text{cost}(x, a) \wedge \text{cost}(x, b) < \text{cost}(y, b) \\ P(a \mid y) > P(a \mid x) \wedge P(b \mid x) > P(b \mid y) \end{aligned}$$

If the above event is true then the minimum-cost flow algorithm will necessarily assign a, b wrongfully. However, note that it might not assign b to x and a to y. Since,

$$\begin{aligned} P(a \mid x) = \text{Emb}(x, a) \cdot p^k (1-p)^{n-k} \\ P(a \mid y) = \text{Emb}(y, a) \cdot p^k (1-p)^{n-k} \end{aligned}$$

Therefore, $P(a \mid y) > P(a \mid x)$ if and only if $\text{Emb}(y, a) > \text{Emb}(x, a)$.

We now prove that there exists a set of pairs $x, y \in \{0,1\}^n$ for a large enough $n \in \mathbb{N}$ that have noisy reads a, b (respectively) such that

$$\text{Emb}(y, a) > \text{Emb}(x, a) \wedge \text{Emb}(x, b) > \text{Emb}(y, b).$$

Let us denote $x_{m,n} = 0^m 10^{n-m-3}11$ and $y_{m,n} = 0^m 10^{n-m-4}100$ for each $m, n \in \mathbb{N}$ such that $m \leq n-4$. Also, denote for each $0 \leq k \leq n-3 : a_k = 0^k, b_k = 0^{k-1}1$. For each m, n, k, it holds that

$$\begin{aligned} \text{Emb}\left(x_{m,n}, a_k\right) &= \binom{n-3}{k} < \binom{n-2}{k} = \text{Emb}\left(y_{m,n}, a_k\right) \\ \text{Emb}\left(x_{m,n}, b_k\right) &= \begin{cases} 2\binom{n-3}{k-1} & k-1 > m \\ 2\binom{n-3}{k-1} + 3\binom{m}{k-1} & k-1 \leq m \end{cases} \\ \text{Emb}\left(y_{m,n}, b_k\right) &= \begin{cases} \binom{n-4}{k-1} & k-1 > m \\ \binom{n-4}{k-1} + 2\binom{m}{k-1} & k-1 \leq m \end{cases} \end{aligned}$$

Therefore,

$$\text{Emb}\left(x_{m,n}, b_k\right) > \text{Emb}\left(y_{m,n}, b_k\right).$$

Therefore, when a_k is a noisy version of $x_{m,n}$ and $b_{k'}$ is a noisy version of $y_{m,n}$ (respectively, for some fixed m, n), the min cost algorithm wrongfully assigns a_k and b_k.

Let P_0 denote the probability that the min cost algorithm fails to identify the true permutation. As shown above, P_0 is lower bounded by the probability that the noisy versions of $x_{m,n}$ and $y_{m,n}$ ($m \leq n-4$) are $a_k, b_{k'}$ ($k, k' \leq n-3$). It can be verified that

$$\begin{aligned} P\left(a_k \mid x_{m,n}\right) &= p^3, \\ P\left(b_{k'} \mid y_{m,n}\right) &\geq p^3(1-p). \end{aligned}$$

Therefore, we have that

$$\begin{aligned} P_0 &\geq 1 - \prod_{m=0}^{n-4} \left(1 - P\left(a_k \mid x_{m,n}\right) P\left(b_{k'} \mid y_{m,n}\right)\right) \\ &\geq 1 - \left(1 - p^6(1-p)\right)^{n-3} \\ &\underset{n \to \infty}{\longrightarrow} 1. \end{aligned}$$

IV. Uniqueness of N-permutation

The task of identifying the true permutation π, can be split into two steps. We can first identify the *partitioning* $\{\mathcal{S}_N((\boldsymbol{x}_i, \boldsymbol{d}_i)) : i \in [M]\}$ and then for each *partition* $(\mathcal{S}_N((\boldsymbol{x}_i, \boldsymbol{d}_i)))$ identify the *label*, viz. the channel input $(\boldsymbol{x}_i)$, where $i \in [M]$. Hence, given R' and $\mathcal{A}$, we are able to find the true permutation if and only if there exists only one valid partitioning and one valid labelling.

In this section, we study Problem 1 when $\mathcal{S} = \mathsf{BDL}(p)$. Specifically, in Lemmas **??** and **??**, we determine the values L_{Th} and N_{Th}, respectively, such that for all $L \geq L_{\mathsf{Th}}$ and $N \geq N_{\mathsf{Th}}$, we are able to find the true permutation with high probability. The result is formally stated in Theorem **??**.

Before formally defining partitioning and labelling, we introduce some notations. For $\boldsymbol{a}, \boldsymbol{b} \in \{0,1\}^n$, let $\boldsymbol{a}', \boldsymbol{b}'$ be the channel outputs through $\mathsf{BDL}(p)$ of $\boldsymbol{a}, \boldsymbol{b}$, respectively. We say that $\boldsymbol{a}'$ and $\boldsymbol{b}'$ are *congruent*, denoted by $\boldsymbol{a}' \cong \boldsymbol{b}'$, if $|\mathsf{SCS}(\boldsymbol{a}', \boldsymbol{b}')| \leq n$. Further, by abuse of notation, we would denote the event of all sequences in $A \subseteq \mathcal{S}_N(\boldsymbol{a})$ being congruent with all sequences in $B \subseteq \mathcal{S}_N(\boldsymbol{b})$ by $A \cong B$, where $\mathcal{S}_N(\boldsymbol{x}_i)$ denotes the multiset of channel outputs when $\boldsymbol{x}_i$ is transmitted N times through the channel $\mathcal{S}$, $i \in \{1, 2\}$.

A read $(\boldsymbol{y}, \boldsymbol{d}') \in R'$ is said to be **faulty** if there exists some other read $(\widetilde{\boldsymbol{y}}, \widetilde{\boldsymbol{d}}') \in R' \backslash \{(\boldsymbol{y}, \boldsymbol{d}')\}$ with $(\boldsymbol{y}, \boldsymbol{d}') \in \mathcal{S}_N((\boldsymbol{x}, \boldsymbol{d}))$ and $(\widetilde{\boldsymbol{y}}, \widetilde{\boldsymbol{d}}') \in \mathcal{S}_N((\widetilde{\boldsymbol{x}}, \widetilde{\boldsymbol{d}}))$ such that $(\boldsymbol{y}, \boldsymbol{d}') \cong (\widetilde{\boldsymbol{y}}, \widetilde{\boldsymbol{d}}')$. Let $\mathsf{R}_{\mathsf{faulty}}$ denote the multiset of such faulty reads. In the next lemma, we give a bound on the length of the longest common subsequence of two uniformly chosen binary strings, which we would be crucial in analyzing the probability of a read being faulty.

Lemma 1. *[] Let $X_{k,\ell}$ denote the length of the longest common subsequence of two uniformly chosen binary strings of length k and ℓ, respectively and $\lambda > 0$, then*

$$\mathbb{E}[X_{k,\ell}] = \gamma_2 \max\{k, \ell\}, \tag{1}$$

where γ_2[bounds –Shubhransh]. Further,

$$P\left(|X_{k,\ell} - \mathbb{E}\left(X_{k,\ell}\right)| \geq \lambda\right) \leq 2e^{-\frac{\lambda^2}{8\max\{k,\ell\}}}. \tag{2}$$

Let A^* denote the event that all noisy reads have lengths between $[(1-p)L - c_L\sqrt{L}, (1+p)L + c_L\sqrt{L}]$.

Lemma 2. *The probability of the event A^* is at least*

$$P(A^*) \geq \left(1 - 2e^{\left(-2c_L^2\right)}\right)^{N2^n}.$$

Proof. For $c_L > 0$, Let $d_1' \in \{0,1\}^*$, then it can be verified that $\mathbb{E}(|d_1'|) = (1-p)L$. Further, using Hoeffding's inequality, we have that

$$P\left(\left|\,|d_1'| - (1-p)L\right| \geq c_L\sqrt{L}\right) \leq 2e^{\left(-\frac{2(c_L\sqrt{L})^2}{L}\right)},$$
$$P\left(\left|\,|d_1'| - (1-p)L\right| \leq c_L\sqrt{L}\right) \geq 1 - 2e^{\left(-2c_L^2\right)}.$$

Therefore, the probability of the event A^* is at least

$$P(A^*) \geq \left(1 - 2e^{\left(-2c_L^2\right)}\right)^{N2^n}.$$

□

In the next lemma, we calculate the probability of a read being faulty.

Lemma 3. *For $(\boldsymbol{y}, \boldsymbol{d}') \in \mathcal{S}_N((\boldsymbol{x}, \boldsymbol{d}))$,*

$$P(\mathbb{I}_{(\boldsymbol{y},\boldsymbol{d}') \in \mathsf{R}_{\mathsf{faulty}}} | A^*) =$$

Proof. Let $|d'| \triangleq \ell \in [(1-p)L - c_L\sqrt{L}, (1+p)L + c_L\sqrt{L}]$. Consider another noisy strand d_1' obtained from $d_1 \in \{0,1\}^L$ and let $|d_1'| \triangleq \ell_1 \in [(1-p)L - c_L\sqrt{L}, (1+p)L + c_L\sqrt{L}]$. From Lemma**??**, we have that

$$\mathbb{E}\left(|\mathsf{LCS}\left(d', d_1'\right)| \;\middle|\; |d'| = \ell, |d_1'| = \ell_1\right) = \gamma_2 \max\left\{\ell, \ell_1\right\},$$

where $\gamma_2 \approx 0.8$. Further, it is known that $|\mathsf{SCS}\left(d_1', d_2'\right)| = |d_1'| + |d_2'| - |\mathsf{LCS}\left(d_1', d_2'\right)|$. Therefore,

$$\mathbb{E}\left(|\mathsf{SCS}\left(d', d_1'\right)| \;\middle|\; |d'| = \ell, |d_1'| = \ell_1\right) = \ell + \ell_1 - \gamma_2 \max\left\{\ell, \ell_1\right\}.$$

Given A^* and the lengths of the noisy strands, we have that

$$\gamma_2\left((1-p)L + c_L\sqrt{L}\right) \geq \mathbb{E}\left(|\mathsf{LCS}\left(d', d_1'\right)| \;\middle|\; |d'| = \ell, |d_1'| = \ell_1, A^*\right), \tag{3}$$

and

$$|\mathsf{SCS}\,(d', d_1')| \geq 2\left((1-p)L - c_L\sqrt{L}\right) - |\mathsf{LCS}\,(d_1', d_2')|\,.$$

We now analyze the probability that $|\mathsf{SCS}\,(d', d_1')| > L$ given A^* and the lengths of the noisy strands.

$$\begin{aligned}
&P\Big(\,|\mathsf{SCS}\,(d', d_1')| > L \;\Big|\; |d'| = \ell, |d_1'| = \ell_1, A^*\Big)\\
&\leq P\Big(2\left((1-p)L - c_L\sqrt{L}\right) - |\mathsf{LCS}\,(d', d_1')| > L \;\Big|\; |d'| = \ell, |d_1'| = \ell_1, A^*\Big)\\
&= P\Big(\mathbb{E}[|\mathsf{LCS}\,(d', d_1')|] - |\mathsf{LCS}\,(d', d_1')| > \mathbb{E}[|\mathsf{LCS}\,(d', d_1')|] - 2\left((1-p)L - c_L\sqrt{L}\right) + L \;\Big|\; |d'| = \ell, |d_1'| = \ell_1, A^*\Big)\\
&\leq P\Big(\mathbb{E}[|\mathsf{LCS}\,(d', d_1')|] - |\mathsf{LCS}\,(d', d_1')| > L - 2\left((1-p)L - c_L\sqrt{L}\right) - \gamma_2\left((1-p)L + c_L\sqrt{L}\right) \;\Big|\; |d'| = \ell, |d_1'| = \ell_1, A^*\Big).
\end{aligned}$$

For $c_L > \max\left(0, \dfrac{(2+\gamma_2)(1-p)-1}{2-\gamma_2}\right)\sqrt{L}$, from Lemma **??**, it follows that

$$\begin{aligned}
P\Big(\,|\mathsf{SCS}\,(d', d_1')| > L \;\Big|\; |d'| = \ell, |d_1'| = \ell_1, A^*\Big) &\leq 2e^{-\frac{\left(L - 2\left((1-p)L - c_L\sqrt{L}\right) - \gamma_2\left((1-p)L + c_L\sqrt{L}\right)\right)^2}{8\max\{\ell, \ell_1\}}}\\
&\leq 2e^{-\frac{\left(L(1-(2+\gamma_2)(1-p)) + c_L\sqrt{L}(2-\gamma_2)\right)^2}{8\gamma_2\left((1-p)L + c_L\sqrt{L}\right)}}
\end{aligned}$$

□

Definition 1. *A **partitioning** $\mathcal{P} = \{P_1, P_2, \ldots, P_M\}$ of $\mathcal{Y}$ is defined as the collection of disjoint submultisets of $\mathcal{Y}$, each of size N, such that for $i \in [M]$, for $(j,k) \in \binom{[N]}{2}$, $|\mathsf{SCS}(\boldsymbol{y}_j, \boldsymbol{y}_k)| \leq n$ & $|\mathsf{SCS}(\boldsymbol{d}_j', \boldsymbol{d}_k')| \leq L$, where $(\boldsymbol{y}_j, \boldsymbol{d}_j'), (\boldsymbol{y}_k, \boldsymbol{d}_k') \in P_i$.*

V. Permutation Recovery Algorithm

Algorithm 1 Permutation Recovery Algorithm

procedure PRUNE($\mathcal{G}, (\widetilde{\boldsymbol{y}}, \widetilde{\boldsymbol{d}}')$)
  $(\widetilde{\boldsymbol{y}}, \widetilde{\boldsymbol{d}}') \longrightarrow \mathsf{Pruned}$, $\mathcal{T} = \{\}$
  for $(\boldsymbol{y}, \boldsymbol{d}') \in \mathcal{N}_{(\widetilde{\boldsymbol{y}}, \widetilde{\boldsymbol{d}}')}$ **do**
    if $(\boldsymbol{y}, \boldsymbol{d}') \cong (\widetilde{\boldsymbol{y}}, \widetilde{\boldsymbol{d}}')$ **then**
      $(\boldsymbol{y}, \boldsymbol{d}') \longrightarrow \mathcal{T}$
  if $|\mathcal{T}| = N - 1$ **then**
    Let $\mathcal{X}^* = \bigcap_{(\boldsymbol{y}, \boldsymbol{d}') \in \mathcal{T}} E_{(\boldsymbol{y}, \boldsymbol{d}')}$
    for $(\boldsymbol{y}, \boldsymbol{d}') \in \mathcal{T}$ **do**
      Remove $\{(\boldsymbol{x}, (\boldsymbol{y}, \boldsymbol{d}')) : \boldsymbol{x} \notin \mathcal{X}^*\}$ from E
      $(\boldsymbol{y}, \boldsymbol{d}') \longrightarrow \mathsf{Pruned}$

procedure PRUNING ALGORITHM($\mathcal{P}_{\mathcal{G}}, \mathcal{G}$)
  $\mathsf{Pruned} = \{\}$
  while $|\mathsf{Pruned}| < N2^n$ **do**
    $(\widetilde{\boldsymbol{y}}, \widetilde{\boldsymbol{d}}') = \arg\min\{|\mathcal{N}_{(\boldsymbol{y}, \boldsymbol{d}')}| : (\boldsymbol{y}, \boldsymbol{d}') \in \mathcal{Y}\}$
    PRUNE $(\mathcal{G}, (\widetilde{\boldsymbol{y}}, \widetilde{\boldsymbol{d}}'))$
  return $\mathsf{PMA}(\mathcal{P}_{\mathcal{G}}, \mathcal{G})$

Lemma 4. *Let $(\boldsymbol{x}, \boldsymbol{d}) \in R, (\boldsymbol{y}, \boldsymbol{d}') \in \mathcal{S}((\boldsymbol{x}, \boldsymbol{d}))$. Then, we have that*

$$\mathbb{E}[\mathcal{N}_{(\boldsymbol{y}, \boldsymbol{d}')}] = NW_{\mathcal{A}, \boldsymbol{x}}(2p - p^2) - 1$$

Proof. Let $(\widetilde{\boldsymbol{x}}, \tilde{\boldsymbol{d}}) \in R$, such that $d_L(\boldsymbol{x}, \widetilde{\boldsymbol{x}}) = r$, i.e. $\mathsf{LCS}(\boldsymbol{x}, \tilde{\boldsymbol{x}}) = n - r$. Let $(\tilde{\boldsymbol{y}}, \tilde{\boldsymbol{d}}') \in \mathcal{S}_N((\widetilde{\boldsymbol{x}}, \tilde{\boldsymbol{d}}))$. Then for $(\boldsymbol{y}, \boldsymbol{d}') \in \mathcal{N}_{(\tilde{\boldsymbol{y}}, \tilde{\boldsymbol{d}}')}$, there must be at least one erasure in $\boldsymbol{x}$ or $\widetilde{\boldsymbol{x}}$ at all the positions where they differ, which happens with probability $(1-(1-p)^2)^r = (2p-p^2)^r$. Hence, $\mathbb{E}[\mathcal{N}_{(\boldsymbol{y}, \boldsymbol{d}')}] = \sum_{r=1}^{n} W_{r,\boldsymbol{x}}(2p-p^2)^r N + (N-1) = N W_{\mathcal{A},\boldsymbol{x}}(2p-p^2) - 1$. □

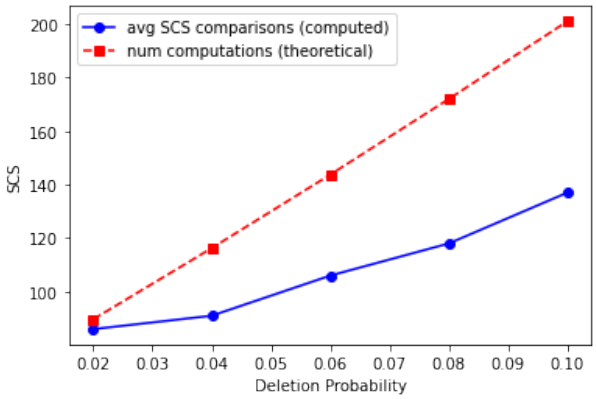

Fig. 1: Computed vs Theoretical SCS comparisons.

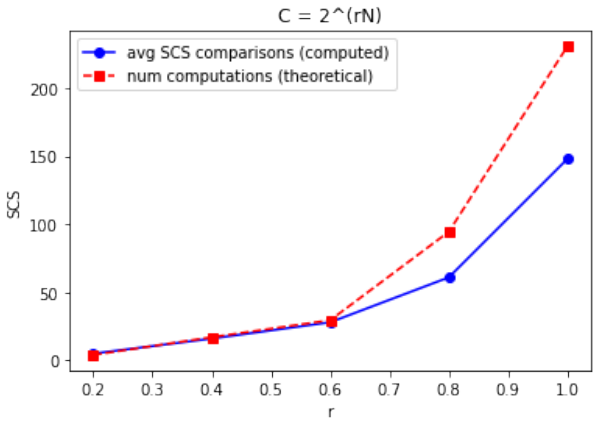

Fig. 2: Computed vs Theoretical SCS comparisons.

VI. References

[1] S. Singhvi, A. Boruchovsky, H. M. Kiah and E. Yaakobi, Data-Driven Bee Identification for DNA Strands, 2023.

[2] J. Chrisnata, H. M. Kiah, A. Vardy and E. Yaakobi, "Bee Identification Problem for DNA Strands," IEEE Journal on Selected Areas in Information Theory, vol. 4, pp. 190-204, 2023.

[3] C. Rashtchian, K. Makarychev, M. Racz, S. Ang, D. Jevdjic, S. Yekhanin, L. Ceze and K. Strauss, "Clustering Billions of Reads for DNA Data Storage," in Advances in Neural Information Processing Systems, 2017.

[4] G. S. Lueker, "Improved bounds on the average length of longest common subsequences," Journal of the ACM (JACM), vol. 56, p. 1-38, 2009.