An Information Theory for Out-of-Order Media With Applications in DNA Data Storage

Aditya Narayan Ravi, *Graduate Student Member, IEEE*, Alireza Vahid, *Senior Member, IEEE*, and Ilan Shomorony, *Member, IEEE*

***Abstract*—Recent advancements in DNA-based storage prototypes focus on encoding information across multiple DNA molecules. This approach utilizes high-throughput sequencing technologies, leading to outputs that are out-of-order. We study the shuffling channel, where input codewords are split into fixed-size fragments. We show that achieving channel capacity uses index-based coding, which assigns unique indices to each fragment. We also introduce two more complex channels, which aim to model popular sequencing strategies in DNA sequencing. In the torn-paper channel, the input codeword is torn up into fragments of random sizes, while in the shotgun sequencing channel, fixed-length random substrings of the input codeword are observed at the output. In both of these channels, the lack of ordering cannot be circumvented by simply adding unique indices to the fragments. We show how the capacity of both of these channels can be achieved using random codes. We introduce and analyze code constructions based on index sequences. While these codes are computationally efficient, they are not capacity-achieving, and we leave the questions of finding efficient capacity-achieving codes for these settings as open problems.**

***Index Terms*—Biological information theory, DNA, Channel coding, Decoding.**

I. Introduction

MOST standard models for communication channels assume that the channel does not affect the order of the input symbols. For example, a *discrete memoryless channel* [1], described by a conditional probability distribution $p(y|x)$, takes a sequence of inputs $x_1, \ldots, x_n$ and outputs the (ordered) sequence $y_1, \ldots, y_n$, where y_i is generated via the conditional distribution $p(y_i|x_i)$, for $i = 1, \ldots, n$. Even for some channels with memory, such as the deletion channel, the insertion channel, or sticky channels [2], where input symbols may be lost and new random symbols may be inserted into the sequence, the order of the input symbols that "survive" the channel is maintained. The assumption of an "ordered" channel makes sense for most practical communication systems, where symbols are transmitted one at a time. This ordering assumption can be violated in some packeted network scenarios, where different packets may take different routes to achieve their destination [3], but it can be easily overcome in practice by placing a unique address in the beginning of each packet. The ordering assumption also makes sense for most practical storage systems, where bits are stored in physically well-ordered locations.

However, with the emergence of new communication and storage media based on biological systems, the assumption of ordering may no longer be a valid one. For example, the new paradigm of molecular communication has gained recent attention due to potential applications in medical treatments with nanonetworks [4]. In this setting, nanoscale devices inside the patient's body communicate with each other through molecular diffusion, in which case the received "signals" are out-of-order with respect to what is transmitted [5].

Another emerging technology that can be seen as "out-of-order media" is the idea of macromolecular based storage (and in particular DNA-based storage), which has received significant research attention in the last few years [6], [7], [8]. By storing data on DNA molecules, one can achieve very high storage density and longevity, and several DNA storage system prototypes have recently been implemented [9], [10], [11], [12]. In most of these prototypes, due to technological barriers in DNA synthesis, one stores data across a very large number of short DNA strands. At the time of reading, one utilizes high-throughput sequencing platforms, which randomly sample DNA strands from the pool and read them. For these reason, the DNA storage system can effectively be seen as an out-of-order communication channel. This channel has been the topic of several papers in information theory whose goal is to characterize their channel capacity, optimal coding schemes, and error exponents [7], [13], [14], [15].

A first step in the analysis of out-of-order channels is to understand how the amount of stored data scales with the size of the output fragments. Suppose the channel input is a codeword of length n and the channel output is an unordered multiset $\mathcal{Y}$ of strings (or fragments) of (average) length ℓ_n. Depending on how ℓ_n scales with n, the number of data bits that can be reliably stored in the system scales as different functions of n. Based on prior works (see, for instance, [7]), it can be verified that the number of bits $g(n)$ that can be reliably

Manuscript received 31 December 2023; revised 6 April 2024; accepted 10 May 2024. Date of publication 21 May 2024; date of current version 17 June 2024. The work of Aditya Narayan Ravi and Ilan Shomorony was supported in part by the National Science Foundation (NSF) under Grant CCF-2007597 and Grant CCF-2046991. The work of Alireza Vahid was supported in part by the National Science Foundation (NSF) under Grant CNS-2343964. The associate editor coordinating the review of this article and approving it for publication was H. M. Kiah. *(Corresponding author: Aditya Narayan Ravi.)*

Aditya Narayan Ravi and Ilan Shomorony are with the Electrical and Computer Engineering Department, University of Illinois at Urbana–Champaign, Urbana, IL 61801 USA (e-mail: anravi2@illinois.edu; ilans@illinois.edu).

Alireza Vahid is with the Electrical and Microelectronic Engineering Department, Rochester Institute of Technology, Rochester, NY 14623 USA (e-mail: alireza.vahid@rit.edu).

Digital Object Identifier 10.1109/TMBMC.2024.3403759

TABLE I
SCALING OF THE NUMBER OF BITS THAT CAN BE RELIABLY STORED

fragment length ℓ_n	number of bits $g(n)$
constant	$\Theta(\log n)$
$\bar{L}\log n, \bar{L} < 1$	$\Theta(n^{\bar{L}})$
$\bar{L}\log n, \bar{L} > 1$	$\Theta(n)$
$\omega(\log n)$	$\Theta(n)$

stored scales with the average length ℓ_n of the strings in $\mathcal{Y}$, as shown in Table I. Notice that, if ℓ_n scales as $\bar{L}\log n$ for some constant $\bar{L} > 1$, or faster, the number of stored bits scales linearly in n, allowing us to define storage capacity in the usual sense of $\lim g(n)/n$. On the other hand, if ℓ_n is a constant, one may consider the ratio $\lim g(n)/\log n$ as a relevant measure of the fraction of stored bits instead. This setting was studied in [16], [17], [18], under the name of *permutation channels*.

In this work, we will focus on the slowest scaling of ℓ_n for which $g(n) = \Theta(n)$, namely $\ell_n = \bar{L}\log n$, for $\bar{L} > 1$. A natural starting point is the deterministic length scenario, where each fragment has length exactly $\ell_n = \bar{L}\log n$. This setting was first studied in the context of the *shuffling channel* [19]. The shuffling channel takes a length-n binary string x^n as its input. The channel tears x^n into n/ℓ_n pieces of size ℓ_n each, and outputs these fragments as a multiset $\mathcal{Y}$. In [19], it is shown that the capacity of this channel is given by

$$C_{\text{shuf}} = 1 - 1/\bar{L},$$

when $\bar{L} > 1$. This capacity can be achieved by a simple index-based coding scheme, where we encode a unique identifier in the first $\log(n/\ell_n)$ bits of each fragment. We can do this because the fragments have deterministic lengths, and thus we know a priori the starting location of each fragment in the input sequence x^n.

The impact of the lack of ordering in the output of a DNA-based storage channel can be aggravated by two practically motivated reasons:

(1) Depending on the physical storage conditions (such as temperature and pressure), the stored DNA molecules may be subject to random breaks, causing the readouts to correspond to random substrings (of random lengths) of the stored DNA strings.
(2) If the data is written onto long DNA strands but read using short-read sequencing technologies (such as the widely popular Illumina platforms), then the readouts correspond to random substrings (of a fixed length) of the stored DNA strings.

An important aspect about the lack of ordering produced by (1) and (2) above is that it cannot be easily circumvented via the addition of addresses/indices as we could for the shuffling channel. This is because, in the situations created by (1) and (2), we do not have a priori knowledge of the starting points for the sequence pieces we will observe at the output. Hence, one cannot place unique addresses at the beginning of each block to help with the reordering of the output blocks. As illustrated in Figure 1, if one attempts to spread evenly spaced addresses throughout the input sequence x^n, and the channel output $\mathcal{Y}$ corresponds to random substrings (possibly of random lengths), identifying where the address is in each substring is not a straightforward task, and it is possible that addresses are only partially included at the ends of the substrings.

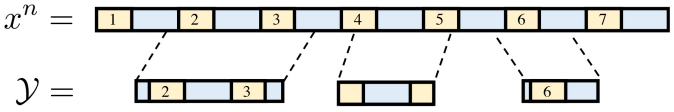

Fig. 1. Consider a channel where one observes a set $\mathcal{Y}$ of random substrings of the input sequence x^n. One attempt to combat the loss of ordering at the output could be to place unique addresses throughout x^n (shown here as the numbered segments). However, since the starting point of the observed strings is unknown, the location of the addresses in the observed strings is unknown, and they may be only partially included in the observed strings.

This leads to an interesting coding question for such out-of-order channels: How should one encode a message into the codeword x^n for out-of-order communication channels? In particular, when the starting point of the observed fragments are unknown, is there a way to use addresses to allow the reordering of the channel output? We will investigate two channel models that impose such a loss of ordering to the channel output, and study the design of *reordering codes* that can be used to communicate reliably in out-of-order settings.

The two channel models we will consider are inspired by the aforementioned practical considerations (1) and (2) and are illustrated in Figure 2. We refer to these two channels as the Torn-Paper Channel (TPC) [20], [21], [22], shown in Figure 2(a), and the Shotgun Sequencing Channel (SSC) [23], [24], shown in Figure 2(b). The TPC breaks the input sequence x^n into pieces of random sizes, and a subset of these pieces is observed at the output. This is motivated by the setting where physical conditions make the DNA molecules subject to random breaks, and a long-read sequencing technology such as nanopore sequencing [25] is used to read entire pieces. It can also be motivated by applications in fingerprinting and forensics, where one may wish to encode a serial number into a physical object (such as a weapon), which should be recoverable even from a small set of pieces left over from the original object [26], [27]. The SSC extracts many reads of a fixed length from random locations of the codeword x^n, which is motivated by the setting where one utilizes Illumina sequencing platforms [28] to read a long synthesized DNA molecule.

Next, we describe both of these models more formally. We start with the TPC, which takes a length-n binary string x^n as its input. We point out that all results in this paper can be extended in a straightforward manner to non-binary alphabets, but we focus on the binary case for simplicity. The TPC tears x^n into fragments of random sizes $N_1, N_2, \ldots$ (these fragments are referred to as *reads*). We assume that $N_1, N_2, \ldots$ are i.i.d. random variables with expected fragment length $E[N_1] = \ell_n$. (We are yet to define a stopping criteria to count the number of pieces. This is discussed formally in Section II). The output of the channel is then the unordered set of the resulting pieces (possibly with some missing ones, as we discuss later).

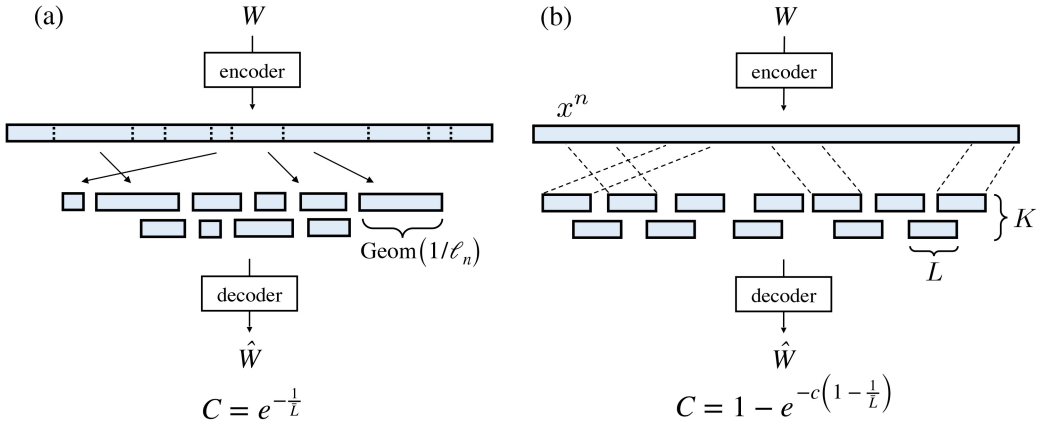

Fig. 2. Comparison between the (a) Torn Paper Channel (TPC) and the (b) Shotgun Sequencing Channel (SSC). The input to the SSC is a single (binary) string x^n and the output are K random substrings of length L.

Notice that we allow the distribution of N_1 to change with n since letting $E[N_1]$ scale as $\log n$ is the regime of interest from a channel capacity standpoint. In particular, when the lengths $N_1 \sim \text{Geom}(1/\ell_n)$ (which implies $E[N_1] = \ell_n$), the capacity of this channel is

$$C_{\text{TPC}} = e^{-1/\bar{L}}, \tag{1}$$

where $\bar{L} = \lim_{n\to\infty} \frac{\ell_n}{\log n}$, as first shown in [20]. Notice that this agrees with our intuition that the channel capacity should increase towards 1 as $\bar{L}$ grows.

As it turns out, for the general case where the distribution of N_i is arbitrary, but still i.i.d., the capacity has an intuitive form that can be expressed as "coverage"−"reordering-cost". The precise mathematical forms of these quantities will be discussed in the section dedicated to the TPC. At a high level, "coverage" is the fraction of bits in the input string present in the set of output reads, after discarding all pieces of size at most $\log n$, which can be shown to carry no useful information from a capacity standpoint. The reordering cost is intuitively the cost in terms of fraction of bits effectively used to reorder the unordered set (again after throwing out pieces of size at most $\log n$), and will be made more explicit later.

The Shotgun Sequencing Channel (SSC), shown in Figure 2(b), also takes a length-n binary string as its input. But unlike the TPC, it picks points uniformly at random on the input string and samples K fixed-length reads of size $L = \bar{L}\log n$ from this string. This unordered set is the channel output. The capacity of the SSC is given by

$$C_{\text{SSC}} = 1 - e^{-c\left(1-\frac{1}{\bar{L}}\right)}, \tag{2}$$

where $c = KL/n$ is the *coverage depth*. The coverage depth c, assumed to be a fixed constant, is a standard parameter in practical sequencing experiments [29] and corresponds to the expected number of times an input symbol is read.

A key difference between the SSC and the TPC is that the SSC in general creates reads that may have overlaps with each other. In other words, two reads with starting locations that differ by less than L will have a matching prefix and suffix. Intuitively, these overlaps can be used to merge reads, forming longer pieces of sequence. Therefore, in order to characterize the capacity of the SSC, one must understand the optimal way to utilize overlaps for merging reads (while being careful not to make incorrect merges due to spurious matches between prefixes and suffixes). As we will discuss, the optimal coding scheme for the SSC is vastly more complicated to analyze than the optimal coding scheme for the TPC, since it requires a careful approach to merge reads.

New Contributions: In this paper, we discuss the SSC and TPC channel models in detail and survey existing capacity results. While the capacity expressions have been previously studied separately, here we study them within the same framework, providing qualitative and quantitative comparisons.

Our new contributions are fourfold: (i) We present the capacity expressions in a unified framework, which allows them to be compared under the same choice of parameters such as coverage and expected fragment length (e.g., see Figures 4 and 5). This provides insight into the impact of the fragment length distribution and overlaps in the capacity of DNA storage systems. (ii) We present an explicit address-based code construction for the SSC. A version of this coding scheme had originally been proposed for the TPC, but here we show that the approach can be seen as a general way to code for out-of-order channels, which can be adapted to the specifics of the channel (and in particular the SSC). (iii) We study the zero-error capacity of a TPC with bounded fragment lengths, which allows us to draw comparisons between the probabilistic TPC and a recently studied TPC in an adversarial setting. (iv) We introduce and study a noisy version of the TPC. While an exact characterization of the capacity is challenging, we present novel achievable rates for this channel.

The organization of this paper is as follows. In the next section, we will discuss the channel models considered, state their capacities and compare them. This is followed by Section III, where we consider index-based (address-based) *reordering codes*. We pay special attention to its application for the SSC, which is unexplored in existing literature. We further discuss unresolved open questions. We also discuss an efficient coding scheme from [30], a code with a linear run-time decoder for the TPC within an adversarial setting and a method involving Varshamov-Tenengolts codes that embed indices within the torn pieces of the TPC [31], [32]. We then

conclude in Section IV by introducing a noisy extension of the TPC, studying achievable rates, and posing some open questions about this channel.

Related Work: The analysis of DNA storage channels started with the initial prototypes of DNA-based storage systems by Church et al. [9] and Goldman et al. [10]. Subsequent advances demonstrated the feasibility of long-term storage through error-correcting codes [11], selective data access [33], high information densities [34], and scaled-up storage capabilities [35]. All these prototypes utilized short DNA molecules due to the high cost of synthesizing longer strands. For that reason, most of the literature on DNA storage channels has focused on short fragments. In that context, the work we present on the SSC can be understood as an investigation into potential storage rates achievable if we were to use long DNA molecules instead.

The results presented in this paper are aligned with the recent literature on modeling the DNA storage channel and studying its capacity, which includes studies on the noisy shuffling channel model [7], [19], [36] and multi-draw sampling channels [13], [37], [38]. A comprehensive survey on this topic can be found in [15]. Out-of-order channels have also been considered in the context of the noisy permutation channel [18], [39], which is a channel that breaks codewords into individual symbols and shuffles them. The bee identification problem [40], [41], [42], [43] is another example of an out-of-order channel, where barcodes are output in a noisy and unordered fashion.

Concurrent research efforts have been dedicated to developing explicit codes catering to DNA storage specifics, addressing synthesis constraints [33], [34], [44], asymmetric sequencing errors [45], insertion error correction [46] and deletion error correction [47].

II. Channel Models and Capacity Expressions

In this section we formally define the TPC and the SSC, state and compare the capacity expressions, and discuss the rates achieved by random coding schemes. We utilize a standard definition of capacity:

Definition 1: A rate R is achievable if there exists a sequence of codes with rate R and blocklength n whose error probability satisfies $P(\mathcal{E}) \to 0$ as $n \to \infty$. The capacity C is the supremum of achievable rates R.

A. Capacity of the Torn-Paper Channel

In the TPC, let W be the message to be transmitted. The transmitter encodes a message $W \in \{1, 2, \ldots, 2^{nR}\}$ into a length-n binary codeword $X^n \in \{0,1\}^n$. The output $\mathcal{Y}$ of the channel is obtained as follows: The channel tears the input codeword X^n into pieces (reads) of size $N_1, N_2, \ldots$ where N_is are i.i.d. random variables. We assume that $E[N_i] = \ell_n = \bar{L} \log n$ is the expected length of the pieces, and we let K_n be the smallest index such that $\sum_{i=1}^{K_n} N_i \geq n$. Notice that K_n is also a random variable. We let $\vec{X}_i$ be the ith piece of length N_i, for $i = 1, \ldots, K_n - 1$. As illustrated in Figure 2(a), the channel output is the multiset

$$\mathcal{Y} \coloneqq \{\vec{X}_1, \vec{X}_2, \ldots, \vec{X}_{K_n}\}, \tag{3}$$

where the segments $\vec{X}_1, \ldots, \vec{X}_{k_n}$, are given by

$$\vec{X}_i = \left[X_{1+\sum_{j=1}^{i-1} N_j}, \ldots, X_{\sum_{j=1}^{i} N_j}\right], \tag{4}$$

for $i = 1, \ldots, K_n - 1$ and

$$\vec{X}_{K_n} = \left[X_{1+\sum_{j=1}^{K-1} N_j}, \ldots, X_n\right]. \tag{5}$$

The following theorem, which is a special case of the main result in [22], characterizes the capacity of the TPC.

Theorem 1: The capacity C_{TPC} of the Torn-Paper Channel is

$$C_{\mathrm{TPC}} = \lim_{n\to\infty} \frac{1}{\ell_n} E\left[N_1 \mathbf{1}_{\{N_1 \geq \log n\}}\right] - \lim_{n\to\infty} \frac{\log n}{\ell_n} E\left[\mathbf{1}_{\{N_1 \geq \log n\}}\right]. \tag{6}$$

We refer to the first term in (6) as the "coverage" and to the second term as the "reordering-cost". Notice that both the coverage and the reordering-cost terms involve the indicator function $\mathbf{1}_{\{N_1 \geq \log n\}}$, which essentially discards reads in $\mathcal{Y}$ of length less than $\log n$. But why are reads of size less than $\log n$ discarded? An intuitive reason for this is as follows: Suppose the encoder knew where the tearing points occurred in the string. Then an optimal way to encode messages (at least intuitively) would be to allocate a certain portion of bits in each individual read to encode ordering information, by using unique binary addresses that correspond to the position where each read starts. The number of bits per read that needs to be allocated to achieve this (in expectation) is $\log(n/\ell_n) \approx \log n$, since we expect the total number of reads to be n/ℓ_n. Therefore pieces of lengths smaller than $\log n$ cannot store any information about the message, and are hence useless.

We point out that under very mild conditions on the first and second moments of N_1 [22], we can rewrite (6) as

$$\lim_{n\to\infty}\left(\frac{1}{\ell_n} E\left[N_1 \mathbf{1}_{\{N_1 \geq \log n\}}\right] - \frac{\log n}{\ell_n} E\left[\mathbf{1}_{\{N_1 \geq \log n\}}\right]\right) = \frac{1}{\bar{L}} \int_1^\infty (\beta - 1) h(\beta) d\beta, \tag{7}$$

where $\lim_{n\to\infty} \Pr(N_1 = \beta \log n) \log n \triangleq h(\beta)$. This allows the capacity of the TPC to be computed explicitly for several choices for the distribution of N_1. The case when $N_i \sim \mathrm{Geom}(1/\ell_n)$ is particularly interesting, since it corresponds to the situation where a tearing occurs between any two consecutive bits in x^n independently and with a fixed probability $1/\ell_n$. When $N_1 \sim \mathrm{Geom}(1/\ell_n)$, we have

$$h(\beta) = \lim_{n\to\infty} \Pr(N_1 = \beta \log n) \log n = \lim_{n\to\infty} \frac{\log n}{\ell_n}\left(1 - \frac{1}{\ell_n}\right)^{\beta \log n - 1} = \frac{e^{-\beta/\bar{L}}}{\bar{L}}. \tag{8}$$

Therefore, from (7) we have that the capacity expression in this case is given by

$$C_{\mathrm{TPC}} = \frac{1}{\bar{L}^2} \int_1^\infty (\beta - 1) e^{-\beta/\bar{L}} d\beta = e^{-1/\bar{L}}, \tag{9}$$

where $\bar{L} \coloneqq \lim_{n\to\infty} \ell_n / \log n$.

TABLE II
CAPACITY EXPRESSIONS FOR VARIOUS FRAGMENT LENGTHS (N_i) AND READ DELETION FUNCTIONS ($\hat{d}$) OF THE TPC-LP

$\hat{d}(\beta)$	N_i	$h(\beta)$	C_{TPC}
0	Geometric$(1/\ell_n)$	$e^{-\beta/\bar{L}}/\bar{L}$	$e^{-1/\bar{L}}$
ϵ	Geometric$(1/\ell_n)$	$e^{-\beta/\bar{L}}/\bar{L}$	$(1-\epsilon)e^{-1/\bar{L}}$
$e^{-\gamma\beta}$	Geometric$(1/\ell_n)$	$e^{-\beta/\bar{L}}/\bar{L}$	$\left(1-\frac{e^{-\gamma}}{(1+\bar{L}\gamma)^2}\right)e^{-1/\bar{L}}$
0	U$[0:\gamma\log n]$, $\gamma\geq 1$	$1/\gamma$	$((\gamma-1)/\gamma)^2$
0	Fixed(ℓ_n), $\ell_n \geq \log n$	NA[1]	$1-\frac{1}{\bar{L}}$

[1] $h(\cdot)$ does not exist, hence we directly employ Theorem 1.

The capacity of the TPC can also be generalized to the case where some pieces are "lost" and not observed at the output $\mathcal{Y}$. This is reasonable from a practical standpoint as, during the sequencing process (including nanopore sequencing), not all the DNA in the pool is sequenced. If we assume that a d fraction of the pieces are lost uniformly at random, it is straightforward to modify the analysis used to prove Theorem 1 and show that the capacity of the TPC with lost pieces is given by

$$C_{\text{TPC}} = (1-d)e^{-1/\bar{L}} \tag{10}$$

when $N_i \sim \text{Geom}(1/\ell_n)$ [22].

Remark: The parameter d can in general be a function of the length of the reads, and can be described as $d(.)$. It turns out that the capacity of this new channel (referred to as the TPC with Lost Pieces or TPC-LP), can be characterized as

$$C_{\text{TPC-LP}} = \frac{1}{\bar{L}}\int_1^\infty (\beta-1)\Big(1-\hat{d}(\beta)\Big)h(\beta)d\beta, \tag{11}$$

where the *deletion function* $\hat{d}$ is defined as $\hat{d}(\beta) := \lim_{n\to\infty} d(\beta\log n)$. The following table computes capacity explicitly for many choices of N_i and $d(.)$

Achievability via random coding: The achievability for Theorem 1 is based on a random coding argument. More precisely, a codebook with 2^{nR} codewords of length n is generated by picking every entry independently as Bern(1/2). The decoder then tosses out all pieces in $\mathcal{Y}$ of size less than $\log n$ and looks for the codeword that contains all the pieces in set $\mathcal{Y}$, with no overlaps between the pieces. If there is only one codeword satisfying this, the decoder declares that codeword as having been sent; else it declares an error.

Without loss of generality let us assume that $W=1$ was transmitted. Let $\mathcal{Y}'$ be the set of all reads with length at least $\log n$. If the elements of $\mathcal{Y}'$ exist as non-overlapping substrings in some $\mathbf{x}_i$ in the codebook, it declares the index of that codeword as the message transmitted, i.e., $\hat{W}=i$. We can bound the error $\mathcal{E}$ averaged over all codebook choices as

$$\begin{aligned}\Pr(\mathcal{E}) &= \Pr(\mathcal{E}|\,W=1)\\ &= \Pr\big(\exists j: x_j \neq 1\text{contains all strings in } \mathcal{Y}'\big|\,W=1\big).\end{aligned} \tag{12}$$

To bound this error probability we need to calculate the total number of different permutations on the order of the reads in $\mathcal{Y}'$ and see what the probability is that any such ordering can be placed in any codeword corresponding to $W\neq 1$. However, all these are random quantities, and the cardinality of $\mathcal{Y}'$ is also random. As it turns out, we have a handle on how many such fragments exist in $\mathcal{Y}'$ as $n\to\infty$, with high probability. The following lemma [22] formalizes this.

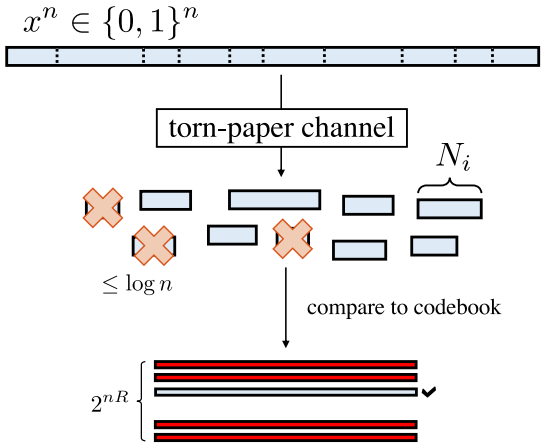

Fig. 3. Illustration of the decoding process for the achievable scheme. The decoder tosses out reads of size at most $\log n$, and picks the codeword from the codebook that contains all the remaining reads as substrings. As long as $R \leq C_{\text{TPC}} - \epsilon$, the probability that a wrong codeword is decoded vanishes as $n\to\infty$.

Lemma 1: For any $\epsilon>0$, as $n\to\infty$,

$$\Pr\left(\left|\sum_{i=1}^{K_n}\mathbf{1}_{\{N_i\geq\log n\}} - \frac{nE\Big[\mathbf{1}_{\{N_1\geq\log n\}}\Big]}{\ell_n}\right| \geq \frac{n\epsilon'}{\ell_n}\right)\to 0, \tag{13}$$

where $\epsilon' := \epsilon E[\mathbf{1}_{\{N_1\geq\log n\}}]$.

Notice that the quantity $\sum_{i=1}^{K_n}\mathbf{1}_{\{N_i\geq\log n\}}$ just counts whether a fragment $\vec{X}_i$ from $\mathcal{Y}$ is in $\mathcal{Y}'$. The lemma explicitly states this is close (cumulatively) to the reordering-cost $E[K_n]\times E[\mathbf{1}_{N_1\geq\log n}] = \frac{nE[\mathbf{1}_{\{N_1\geq\log n\}}]}{\ell_n}$ with high probability.

Since picking a codeword in error implies that an independently generated codeword (which has nothing to do with the codeword generated for $W=1$), contains all elements in $\mathcal{Y}'$ (in a certain order), the cumulative total number of bits amongst all reads in $\mathcal{Y}'$ for some order must match the bits in the same order in the wrong codeword. To calculate this we need a similar lemma [22] on the total number of bits in $\mathcal{Y}'$.

Lemma 2: For any $\epsilon>0$, as $n\to\infty$,

$$\Pr\left(\left|\sum_{i=1}^{K_n}\frac{N_i\mathbf{1}_{\{N_i\geq\log n\}}}{n} - \frac{E\Big[N_1\mathbf{1}_{\{N_i\geq\log n\}}\Big]}{\ell_n}\right| > \epsilon'\right) \to 0, \tag{14}$$

where $\epsilon' := \epsilon E[N_1\mathbf{1}_{\{N_i\geq\log n\}}]/\ell_n$.

Again note that the quantity $\frac{1}{n}\sum_{i=1}^{K_n}N_i\mathbf{1}_{\{N_i\geq\log n\}}$ calculates the total fraction of bits contained within the reads in $\mathcal{Y}'$ and the lemma explicitly states this is close to the coverage $(\frac{E[N_1\mathbf{1}_{\{N_i\geq\log n\}}]}{\ell_n})$ with high probability. These lemmas are proved in [22].

Lemmas 1 and 2 let us define "bad" events that have a vanishing probability as $n\to\infty$. Let $B_1 = (1+$

$\epsilon)\frac{nE[\mathbf{1}_{\{N_i\geq\log n\}}]}{\ell_n}$ and $B_2=(1-\epsilon)\frac{E[N_1\mathbf{1}_{\{N_i\geq\log n\}}]}{\ell_n}$ and define the event

$$\mathcal{B}=\left\{\sum_{i=1}^{K_n}\mathbf{1}_{\{\vec{X}_i\in\mathcal{Y}'\}}>B_1\right\}\cup\left\{\frac{1}{n}\sum_{i=1}^{K_n}N_i\mathbf{1}_{\{\vec{X}_i\in\mathcal{Y}'\}}<B_2\right\}. \tag{15}$$

Lemmas 1 and 2 imply that $\Pr(\mathcal{B})\to 0$ as $n\to\infty$. Therefore

$$\begin{aligned}\Pr(\mathcal{E})&=\Pr\big(\exists x_j\neq x_1\text{containing all strings in }\mathcal{Y}'|\,W=1\big)\\&\leq\Pr\big(\exists x_j\neq x_1\text{containing all strings in }\mathcal{Y}'|\,W=1,\overline{\mathcal{B}}\big)\\&\quad+\Pr(\mathcal{B})\overset{(a)}{\leq}|\mathcal{C}|\frac{n^{B_1}}{2^{nB_2}}+\Pr(\mathcal{B})\\&\leq 2^{nR}2^{B_1\log n}2^{-nB_2}+o(1).\end{aligned}$$

Inequality (a) follows from the union bound and the fact that given $\overline{\mathcal{B}}$, there are at most n^{B_1} ways to align $\mathcal{Y}'$ to a codeword x_j. To see this note that, given $|\mathcal{Y}'|<B_1$, there are at most n places the fragments can start from to align each piece and at most B_1 such pieces. Since a non-overlapping alignment of the strings in $\mathcal{Y}'$ to a codeword x_j covers at least nB_2 positions of x_j, the probability that it matches x_j on all covered positions is at most 2^{-nB_2}.

Now $\Pr(\mathcal{E})\to 0$ if

$$\begin{aligned}R\leq&\lim_{n\to\infty}(1-\epsilon)\frac{1}{\ell_n}E\left[N_1\mathbf{1}_{N_1\geq\log n}\right]\\&-\lim_{n\to\infty}(1+\epsilon)\frac{\log n}{\ell_n}E\left[\mathbf{1}_{N_1\geq\log n}\right].\end{aligned} \tag{16}$$

Letting $\epsilon\to 0$ we prove the achievable part of Theorem 1.

As we can see, the above scheme can be used to achieve rates arbitrarily close to the capacity. This scheme is however intractable from a computational standpoint since it requires matching the set $\mathcal{Y}'$ to an exponentially large number of candidate codewords, and is hence not practically feasible. In Section III, we will discuss a more practical scheme which separates bits used to transmit message and ordering information.

The converse of Theorem 1 involves partitioning the set $\mathcal{Y}$ into pieces of "roughly" the same size. We can then divide the TPC into a set of parallel channels that produce pieces of almost fixed lengths at the output. Each of these "channels" acts like a shuffling channel [7], [36]. The details of this converse argument are available in [22].

B. Capacity of the Shotgun Sequencing Channel

Recall that the SSC models a setting where a long DNA strand is synthesized and sequenced using a short-read sequencing platform, which extracts fixed-length reads from random locations. As in the TPC, the transmitter encodes a message $W\in\{1,2,\ldots,2^{nR}\}$ into a length-n binary codeword $X^n\in\{0,1\}^n$. The channel then chooses (fixed) k_n starting points uniformly at random, represented by the random vector $T^{k_n}\in[1:n]^{k_n}$. The vector T^{k_n} is assumed to be sorted in a non-decreasing order. Length-L reads ($L=\bar{L}\log n$) are then sampled with T_i, $i=1,\ldots,k_n$ as their starting points. For simplicity, we allow the reads to "wrap around" X^n; i.e., if for any i, $T_i+L>n$, we concatenate bits from the start of X^n to form length-L reads. For example if $T_i=n-2$ and $L=5$, then the read $\vec{X}$ associated with this starting location is $\vec{X}=[X_{n-2},X_{n-1},X_n,X_1,X_2]$. As illustrated in Figure 2(b), the unordered multi-set $\mathcal{Y}=\{\vec{X}_1,\vec{X}_2,\ldots,\vec{X}_{k_n}\}$ of reads resulting from this sampling process is the channel output. The following theorem, presented in [24], establishes the capacity of this channel.

Theorem 2: The capacity C_{SSC} of the SSC is given by

$$C_{\text{SSC}}=\left(1-e^{-c\left(1-\frac{1}{L}\right)}\right)^+, \tag{17}$$

where $c:=\lim_{n\to\infty}KL/n$, and $x^+=\max(x,0)$.

The number of times we sample a read from the long string affects the coverage depth, and hence the capacity. In particular, if we look at the expression of C_{SSC} in a high coverage depth regime (where $c\to\infty$, which essentially happens if K, the number reads sampled grows faster than $n/\log n$), $C_{\text{SSC}}\to 1$. This is where the SSC differs from the TPC. Notice that the TPC can never achieve a capacity 1 when $L=\Theta(\log n)$, while the SSC can, provided we sample enough reads. We will discuss this comparison in more detail in Section II-C.

Achievability via random coding. Proving that rates close to C_{SSC} are achievable is fairly involved, and utilizes a coding scheme with a decoding rule that merges reads based on their overlaps in an intricate manner. The key steps of this achievable scheme are: (i) merging the reads correctly to form *islands*, which are variable-length substrings of the input string and (ii) noticing that these islands are actually non-overlapping substrings so that we can use a decoder very similar to the optimal decoder described for the TPC.

The first step involves a brute-force enumeration of all possible potential overlaps between reads in set $\mathcal{Y}$. Each potential set of admissible merges is stored. We then employ the rate optimal decoder that we developed for the TPC on all the sets that this brute-force search admits. The reordering cost for this setting is due to the fact that only one out of all possible admissible merges within each set is the true merge. Therefore the form of capacity we discussed previously ("coverage"−"reordering-cost"), also applies to the SSC. This scheme, analyzed in a careful way, can be used to show that all rates $R<C_{\text{SSC}}$ are achievable. The full details of this proof are available in [24]. We point out that this coding scheme is also based on random coding and is therefore computationally very expensive, just as in the TPC case.

The converse to Theorem 2 is closely related to the converse for the TPC. Specifically it involves utilizing a genie that merges all reads that have an overlap size greater than a certain threshold. Now the variable-length "islands" are variable-length substrings of the codeword and the problem can thus be viewed as a TPC with lost reads [22]. Careful optimization (with respect to what threshold on the merge length the genie would be allowed to know) of the bound obtained when this principle is applied leads to the converse proof of Theorem 2.

C. Comparing Capacity Expressions

The main value of the capacity expressions for the out-of-order channels described in Sections II-A and II-B is that they

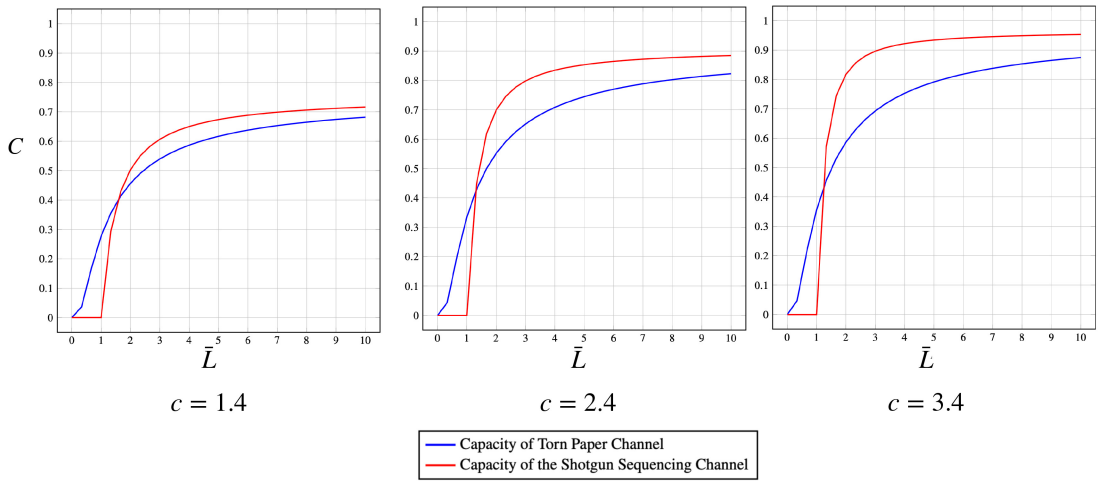

Fig. 4. Comparison between the capacity of the TPC (with lengths having Geometric Distributions) and the SSC, for varying coverage depths and fixed expected read lengths.

capture the impact of key parameters (such as read length, tearing probability, and coverage depth) in the storage capacity. This can provide guidelines for system design, for parameter tuning, and in the selection of synthesis and sequencing technologies. In particular, one can think of a direct comparison between the capacity expressions of the TPC and of the SSC as the comparison between two system architectures: one that uses Illumina sequencing (which is usually preceded by a PCR amplification step [48], which creates many copies of the DNA molecule, allowing reads to have overlaps), and one based on nanopore sequencing (without a PCR amplification step).

Notice that a priori, it is not obvious which of the two systems illustrated in Figure 2 should have the higher capacity. In order to do a fair comparison, we tune parameters d (the probability that a piece is deleted in the TPC) and c (the number of bits covered by reads of a string that is shotgun sequenced) Specifically, by choosing $d = e^{-c}$, we fix the coverage depth to be equal in both models. One can then compare the capacity expressions as a function of the normalized read length parameter $\bar{L}$. Notice that this corresponds to the average (normalized) length for the TPC and the deterministic fixed (normalized) read length for the SSC.

Figure 4 compares the capacity of the TPC and SSC as a function of $\bar{L}$. We can see that in general, the capacity of the TPC dominates the capacity of the SSC in the short read length paradigm, while the capacity of the SSC is higher when the length of the reads increase. This is intuitively because when the read lengths are larger (for a fixed coverage depth), there are more overlaps of a larger length between reads in the SSC. Intuitively larger overlaps are easier to merge than smaller overlaps. This means that if we can merge these overlaps, we obtain much larger pieces that cover a larger portion of the string, which boosts the capacity. This however is not the case when $\bar{L}$ is small. Notice that the range where the TPC capacity is larger than the SSC capacity decreases as c increases. Intuitively this is because as the coverage c increases, more overlaps occur, which allows for more merges and longer resulting pieces, even for small values of $\bar{L}$. In particular, for any $\bar{L} > 1$, the capacity of the SSC tends to 1 as $c \to \infty$, which does not happen to the capacity of the TPC.

Figure 5 shows the capacities of both systems as a function of coverage depth for a fixed read length. As expected, as the coverage depth increases, since more overlaps are likely to be present in the output of the SSC, many merges can be made, and the SSC has a higher capacity at high coverage depths. Moreover, when the fixed length of the reads is larger, the threshold on coverage depth above which the SSC has a larger capacity than the TPC is smaller, again because longer reads imply larger overlaps and thus easier merges.

III. Index-Based Coding Schemes

In Section II, we discussed the exact capacity expressions of the TPC and SSC. Since the achievability arguments of Theorem 1 and Theorem 2 rely on random codes and brute-force decoding, they provide little insight into the design of efficient codes for the out-of-order channels considered.

As discussed in Section I, standard approaches to deal with lack of ordering, such as the placing of unique indices in each fragment, cannot be used in a straightforward manner for the TPC and SSC. This is because, in both cases, we do not know the starting point of the observed fragments. Even if we place evenly spaced indices throughout the input codeword x^n, it is not obvious how to find an address in the output fragments.

A natural question is whether it is still possible to utilize an index-based approach to perform the reordering of the pieces at the output of channels such as the TPC and the SSC. The main idea of index-based coding schemes is to separate bits that convey actual information about the message from bits whose purpose is only to help with the reordering.

We will first look at a simple case where the reads have deterministic tearing lengths. Then we will look at an "interleaving" approach. In this approach a predetermined "pilot" sequence is interleaved with the information bits in a way that we can uniquely align reads extracted from this codeword to a generic "skeleton" codeword. This construction is used to

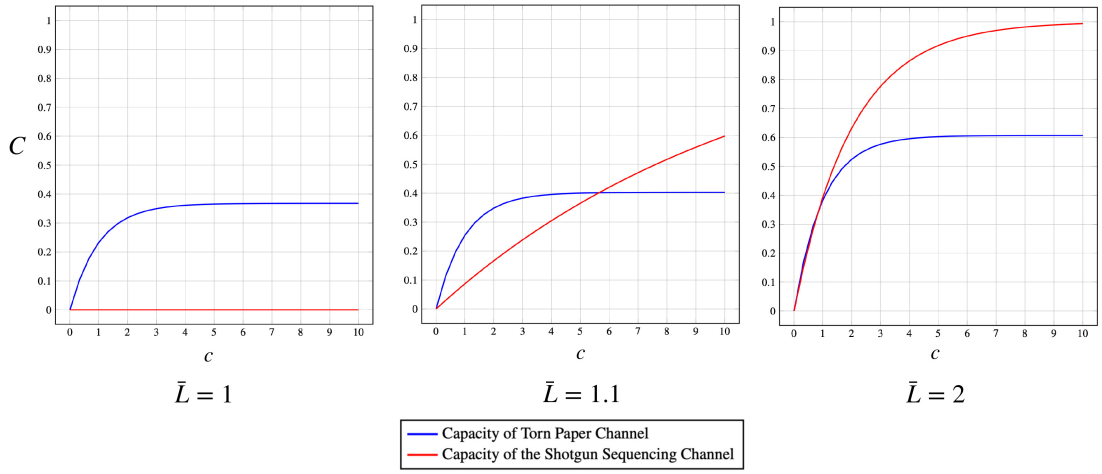

Fig. 5. Comparison between the capacity of the TPC (with lengths having Geometric Distributions) and the SSC, for different coverage depths as the expected read lengths vary.

handle the fact that the read locations are extracted from a priori unknown starting points on the codeword.

Finally we will look at another index-based approach which involves concatenating uniquely identifiable strings after each index, which is then followed by information bits. This scheme is described in detail in [30] and it was originally proposed for an *adversarial* setting of the TPC, where the read lengths are constrained to be between fixed parameters.

A. Index-Based Coding for the Shuffling Channel

In order to build some intuition, let us first consider a simple TPC with deterministic tearing lengths. More precisely, this channel breaks the binary codeword X^n into pieces of length $N_i = \ell_n = \bar{L}\log n$ for $i = 1, \ldots, n/\ell_n$. Since in this case the tearing points are known, the effective input can be seen as M strings of length-ℓ_n, where $M\ell_n = n$. The capacity C of this channel is (as seen in Table II)

$$C = 1 - \frac{1}{\bar{L}}. \tag{18}$$

A very simple index-based coding scheme for this channel is allocating ℓ'_n out of ℓ_n bits of each read to store ordering information and using the remaining $\ell_n - \ell'_n$ bits to store actual message information. There are $M = n/\ell_n$ pieces. Therefore we require $\ell'_n = \log M = \log n - \log \ell_n$ bits to encode ordering information. The way this is done is by encoding (in binary) the position of the piece in the first ℓ'_n bits. For the remaining bits, we can just store message bits directly in an uncoded fashion. The total rate thus achieved by this scheme is just the fraction of bits used to store message information, which is

$$R = \lim_{n\to\infty} \frac{\ell_n - \ell'_n}{\ell_n} \to 1 - \frac{1}{\bar{L}}, \tag{19}$$

Clearly, this is a very efficient coding scheme compared to a random code. Specifically this scheme does not require the decoder to have any a priori knowledge of the codebook to find the codeword sent, which means it does not have to search an exponentially large list to find the transmitted codeword.

B. Index-Based Coding via Interleaving for the Shotgun Sequencing Channel

The problem with using index-based coding schemes for the SSC is that the sampling locations are picked uniformly at random. Thus, we do not know a priori how to allocate bits for indexing purposes, since if we place them at evenly separated points, they will appear at random locations on the reads.

To solve this problem we will create a "pilot sequence" (or synchronization sequence) $\vec{p}$ and interleave it with message bits in order to create codewords. The idea is that any short sequence of consecutive bits of $\vec{p}$ is unique and allows us to determine the location of a read in the sent codeword.

1) Codebook Construction: To construct the codebook, we will create a single pilot sequence $\vec{p}$ and several message blocks $\vec{s}_i$. Let $m = \frac{\bar{L}}{2+\delta}$ and assume for simplicity that $\frac{\bar{L}}{2+\delta}$ is an integer greater than 1. We design these sequences such that no string of length $2\log n$ appears in both the pilot sequence $\vec{p}$ and $\vec{s}_i$ for any i. For some $\delta > 0$, we let $\frac{n}{m} = \frac{(2+\delta)n}{\bar{L}}$ be the length of $\vec{p}$ and $\vec{s}_i$.

We construct $\vec{p}$ as a *de Bruijn sequence* [49] of order $\log(n/m)$. This sequence has length $2^{\log(n/m)} = n/m$ and it has the property that each binary string of length $\log(n/m)$ appears in $\vec{p}$ exactly once. For example, a de Bruijn sequence of order 4 is $S = 0000100110101111$. Notice that each binary string of length 4 appears exactly once (when we view S as a cyclic sequence).

We proceed to then interleave codewords from an erasure code with $\vec{p}$. Let $\mathcal{C}_{\text{er}}$ be an erasure code with block length n/m. We generate a single i.i.d. $\text{Bern}(1/2)$ sequence and compute its letter-wise modulo-2 sum with each codeword in $\mathcal{C}_{\text{er}}$ to form a new codebook $\tilde{\mathcal{C}}_{\text{er}}$. This implies that the original erasure code codewords can be obtained by computing the modulo-2 sum of the new shifted codewords with the fixed $\text{Bern}(1/2)$ sequence. The probability that a randomly shifted codeword $\vec{s} \in \tilde{\mathcal{C}}_{\text{er}}$ shares an identical length-k segment with the pilot sequence can be upper bounded as

$$\Pr(\vec{p}[i : i+k-1] = \vec{s}[j : j+k-1],$$

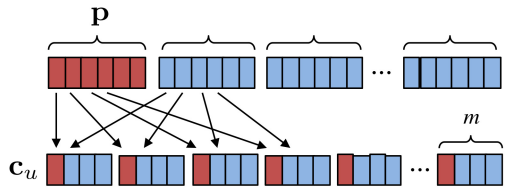

Fig. 6. Interleaving a pilot sequence $\vec{p}$ with codewords $\vec{s}_{u(1)}, \ldots, \vec{s}_{u(m-1)}$ to form the codeword $\vec{c}_u$.

$$\text{for } 1 \le i \le n/m - k,\ 1 \le j \le n/m - k) \le (n/m)^2 2^{-k}. \quad (20)$$

Therefore, if we let $k = (2+\delta)\log n$ for $\delta > 0$,

$$\Pr(\vec{p}[i : i+k-1] = \vec{s}[j : j+k-1], \text{ for } 1 \le i \le n/m - k,\ 1 \le j \le n/m - k) \to 0, \quad (21)$$

as $n \to \infty$. We can then claim that, with high probability, no string of length $(2+\delta)\log n$ appears in both $\vec{p}$ and $\vec{s}_i$. This means that for any $\epsilon > 0$, for n large enough, it is possible to choose the fixed binary sequence so that at least a $(1-\epsilon)$ fraction of the shifted codewords in $\tilde{\mathcal{C}}_{\text{er}}$ contain no length-$(2+\delta)\log n$ segment that is also in the pilot sequence $\vec{p}$. Hence, if the codebook $\mathcal{C}_{\text{er}}$ has a rate R_{er} and a blocklength of n/m (and thus a total of $2^{(n/m)R_{\text{er}}}$ codewords), we can choose $(1-\epsilon)2^{(n/m)R_{\text{er}}}$ shifted codewords in $\tilde{\mathcal{C}}_{\text{er}}$ that contain no length-$(2+\delta)\log n$ segment that is also in $\vec{p}$.

Let $\vec{S} = \{\vec{s}_1, \ldots, \vec{s}_{|\vec{S}|}\} \subset \tilde{\mathcal{C}}_{\text{er}}$ be a set with $(1-\epsilon)2^{\frac{n}{m}R_{\text{er}}}$ such codewords. We build each codeword $\vec{c}_u$ by taking $m-1$ sequences $\vec{s}_{u(1)}, \ldots, \vec{s}_{u(m-1)}$ from $\vec{S}$ and interleaving their symbols with the symbols from $\vec{p}$. More precisely, for each $u \in \{1, \ldots, |\vec{S}|\}^{m-1}$ we build the codeword $\vec{c}_u = (\vec{c}_u[0], \ldots, \vec{c}_u[n-1])$ as

$$\vec{c}_u[mt+j] = \begin{cases} \vec{p}[t], & \text{for } j = 0, \\ \vec{s}_{u(j)}[t], & \text{for } j = 1, \ldots, m-1, \end{cases} \quad (22)$$

for $t = 0, \ldots, n/m - 1$. The interleaving procedure is shown in Figure 6. We use this as the codebook for storage. We refer to the resulting codebook as $\mathcal{C}$, and it has $|\vec{S}|^{m-1} = (1-\epsilon)^{m-1}2^{(1-1/m)nR_{\text{er}}}$ codewords. This yields a coding rate of approximately $(1-1/m)R_{\text{er}}$.

2) Decoding and Analysis: As illustrated in Figure 6, a given codeword will contain one pilot bit from $\vec{p}$ for every m bits. Hence, a given fragment of size $\bar{L}\log n$ contains at least

$$\frac{\bar{L}\log n}{m} = (2+\delta)\log n$$

pilot bits. Since the pilot $\vec{p}$ is a priori fixed, we claim that as long as $\bar{L} > (2+\delta)$, the location of every read can be uniquely identified by aligning each read to a generic codeword, with only the pilot bits specified. Suppose by contradiction that the fragment can be properly aligned to $\vec{c}_i$ at an incorrect location. Note that if $L > (2+\delta)\log n$, then $\bar{L} > (2+\delta)$ or $m > 1$. Therefore since $\vec{p}$ is a de Bruijn sequence of size *n/m*, any substring of $\vec{p}$ which is of size at least $\log(n/m)$ is unique. In particular, sequences of size $(2+\delta)\log n$ are unique, since $(2+\delta)\log n > \log n > \log(n/m)$.

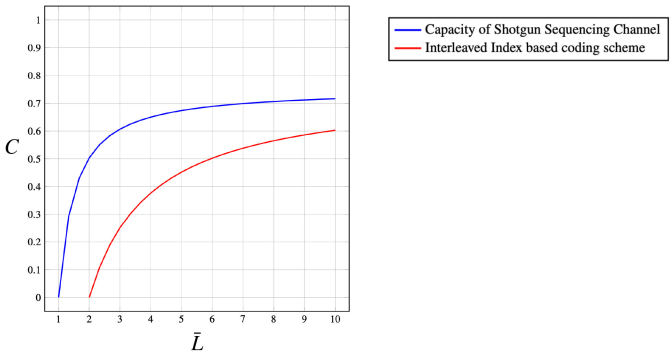

Fig. 7. Comparison between the capacity of the SSC and the rate achieved on the SSC by the explicit code construction based on index coding. We choose a coverage depth of $c = 1.4$.

Since sequences of $(2+\delta)\log n$ consecutive symbols of $\vec{p}$ are unique, it must be the case that pilot symbols of $\vec{c}_i$ align with $(2+\delta)\log n$ non-pilot symbols of the fragment. However, these $(2+\delta)\log n$ symbols must correspond to consecutive symbols in one of the sequences $\vec{s}_i$ from $\vec{S}$. Since no block of length $(2+\delta)\log n$ of $\vec{p}$ appears in any $\vec{s}_i \in \vec{S}$, this is a contradiction.

The rate is now the total number of information bits conveyed. Since we allocate *n/m* bits for indexing, the rate can be evaluated as

$$R = \left(1 - \frac{1}{m}\right)R_{\text{er}}. \quad (23)$$

Under the scaling we choose ($K = cn/L$ and $L = \bar{L}\log n$), the total fraction of bits (in the codeword) sequenced by the SSC is $1 - e^{-c}$ as $n \to \infty$ [24]. Therefore the total fraction of bits erased will be e^{-c}, as $n \to \infty$, and we can choose the rate of the erasure code to be $R_{\text{er}} = 1 - e^{-c}$. Letting $\delta \to 0$, we conclude that rate

$$R = \left(1 - \frac{2}{\bar{L}}\right)\left(1 - e^{-c}\right), \quad (24)$$

is achieved by this coding scheme. This rate is compared to C_{SSC} in Figure 7.

As is evident from the figure, the rates achieved by the index-based coding scheme are significantly lower than C_{SSC}. In fact, as $c \to \infty$, the achievable rate *R* tends to $(1 - \frac{2}{\bar{L}})$ which is strictly less than 1. In the short length regime this scheme therefore performs poorly, even if the coverage depth is high, in comparison to the capacity of the SSC, which satisfies $C_{\text{SSC}} \to 1$ as $c \to \infty$ for all $\bar{L} > 1$.

3) Coding Complexity: When analyzing the complexity of the coding scheme, we consider three distinct steps: code generation (done once, independent of the message sent), encoding, and decoding.

Code Generation: This involves construction of the pilot sequence and the erasure code.

- *Generation of the De Bruijn Pilot Sequence $\vec{p}$:* A single De Bruijn sequence of length *n/m* is generated and stored. This process is independent of the message bits and occurs only once.
- *Selection of Erasure Code:* Choose an existing erasure code with rate R_{er} and blocklength *n/m* is selected.

We can choose existing codes with efficient encoding/decoding, such as a Reed-Solomon code or another efficient erasure code [50].

- *Generation of a consistent Random Binary Shift String:* This ensures no match between shifted codewords and the pilot. To generate a string we require *O(n)*. For each string, we compare the string to the pilot sequence $\vec{p}$. The complexity of this process is $O(n^2)$, for each string we compare to the pilot. It may be required to do this multiple times, but once we obtain a binary shift string with no matches, we can fix it over all message transmissions.

While the total complexity involved in generating the fixed code amounts may big in the worst-case sense, it is important to emphasize that this process is conducted only once at the outset. When viewed in the context of many message transmissions, this initial computational expense becomes negligible.

Encoding: For the actual encoding, we can use the fixed code from above as follows:

- *Division of Message Bits:* The message bits $W = [1{:}2^{nR}]$ are divided into $m-1$ blocks.
- *Application of Erasure Code:* Each block is encoded using the erasure code, which has a linear complexity [50] *O(n/m)*.
- *Applying the Shift and Interleaving:* The encoded blocks are shifted and interleaved with the pilot sequence. Both shifting and interleaving have linear complexity *O(n)*.

The overall encoding complexity of the algorithm would thus be *O(n)*.

Decoding: Using the fixed code from above, we have the following decoding steps:

- *Alignment to the Skeleton Codeword:* This involves matching each piece with the skeleton codeword, with a complexity of *O(n)* per piece. As there are *K* pieces, the total complexity is $O(Kn) = O(n^2/\log n)$.
- *Filling with Erasure Symbols and Undoing Interleaving:* This is performed in linear time *O(n)*.
- *Applying Shift and Erasure Decoding:* Each piece undergoes a shift and then erasure decoding, both having linear complexities, leading to an overall decoding complexity of $O(Kn) = O(n^2/\log n)$. The erasure codes designed in [50] have linear-time decoding, but are based on a randomized construction. Polar codes [51] are deterministic and also exhibit near-linear time decoding complexity, and achieve rates arbitrarily close to capacity.

The overall decoding complexity of the algorithm would thus be $O(n^2/\log n)$.

In summary, while setting up the code at the beginning has a complexity of $O(n^2)$, this step is done just once. The key parts to focus on are the encoding and decoding steps. The encoding complexity is linear in *n*, while the decoding rule has complexity of $O(n^2/\log n)$.

C. Index-Based Coding via Interleaving for the Torn-Paper Channel

In the case of the TPC, we can construct a similar scheme as the one for the SSC. However we need to modify it, because the TPC produces reads of a random length, many of which might be too short to contain enough bits from the pilot sequence. We avoid this issue by discarding all the pieces of size 2*m*log*n* or less and then using the previous scheme. More precisely, just like before we create a pilot sequence $\vec{p}$ and several message blocks $\vec{s}_i$. And as before we construct these sequences such that no string of length 2log*n* appears in the pilot sequence and the message blocks. The codebook construction and encoding remain the same.

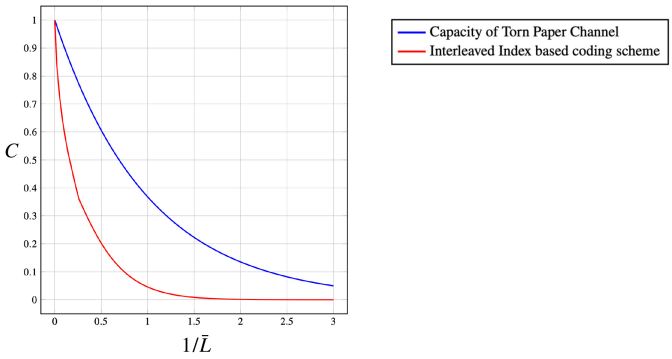

Fig. 8. Comparison between the capacity of the torn-paper channel $C = e^{-1/\bar{L}}$ and the rate achieved on the torn-paper channel by the explicit code construction based on index coding. Note that the x-axis is $1/\bar{L}$ for convenience, since we require $\bar{L} \to \infty$ for $C_{\mathrm{TPC}} \to 1$.

The decoding rule however varies slightly. Since only reads length of size at least 2*m*log*n* can be uniquely aligned to a skeleton codeword, we discard all pieces of size at most 2*m*log*n*. This effectively converts the channel into an erasure channel with total number of erasures

$$\sum_{i=1}^{K} N_i \mathbf{1}_{\{N_i < 2m \log n\}} = n(1 - c_{2m}), \tag{25}$$

where c_{2m} is the fraction of bits covered by fragments of length at least 2*m*log*n*. It can be shown that the rate achieved is thus

$$R = \left(1 - \tfrac{1}{m}\right)\left(\frac{2m}{\bar{L}} + 1\right) e^{-\frac{2m}{\bar{L}}}, \tag{26}$$

where *m* is an integer. We can then optimize over integer values of *m* to obtain the required achievable rates. See [21] for more details. This rate is compared to C_{TPC} in Figure 8. As before, we notice that the gap between this efficient interleaved-pilot approach and the actual capacity is still quite significant.

Thus, when we consider using index-based coding for the torn-paper channel (TPC) and the shotgun sequencing channel (SSC), a key question comes up: Can these methods achieve the capacity of these channels? To tackle this, we might need to go beyond the usual ways of index coding. This could involve developing new indexing strategies that are specifically designed to tackle the random nature of the TPC and SSC. Subsequent research in this domain is essential to unravel the potential of efficient index-based coding in realizing the capacities for this out-of-order media.

D. Index-Based Coding via Unique Headers for the Torn-Paper Channel

The idea of index-based coding was used in [30] to develop an encoding-decoding scheme for the following version of the TPC: The input sequence X^n to the encoder is torn into pieces of size between $L_{\min}$ and $L_{\max}$. Instead of the piece lengths drawn according to a distribution (as discussed previously), they can be chosen in an adversarial manner. As before, the unordered set of pieces is fed to the decoder. In this subsection we shift our focus from the probabilistic framework discussed previously to this adversarial problem setting. This shift is underscored by an emphasis on analyzing the redundancy of a codebook that achieves an error probability exactly zero, in comparison to the previous subsections centered around coding schemes that achieve a vanishing error probability. First we define the zero-error capacity:

Definition 2: The zero-error capacity C_{ZE} is the maximum asymptotic rate that can be achieved with error probability exactly zero.

In the context of probabilistic channels, this concept can be reinterpreted through the lens of a genie that can select any of the possible channel outputs for a given input. Here, the zero-error capacity is essentially the capacity under the worst-case error of this genie channel. For probabilistic channels like the standard TPC [21], there is a non-zero probability of breaking a codeword into pieces of size 1. This implies that the zero-error capacity is zero (since breaking a codeword into pieces of size 1 converts the channel into the permutation channel [16], for which the capacity with standard normalization is zero). However in the adversarial channel described above the minimum length of pieces is given by $L_{\min}$. An interesting question is whether this channel has a positive zero-error capacity. As it turns out, this is true. In [30], an optimal codebook that achieves the positive zero-error capacity for this setting is constructed.

The main idea for the codebook construction is as follows: Represent the messages $W = [1 : 2^{nR}]$ in binary form. Then divide each message into equally spaced substrings of length m. Create a set of indices to concatenate with these substrings and pad these indices with a sequence of a fixed, easily identifiable header ([30] uses a string of zeros). Then convert each substring of length m to another substring (in general having a different length) that does not contain this header anywhere. Now concatenate the indices, the header and the modified substrings in that order, to obtain a codeword. Notice that given a fragment from such a codeword, one can search for the header (a long string of zeros) and then use the index placed right next to it to identify the location of the fragment.

In [30], a specific construction for a $(L_{\min}, L_{\max})$-*single strand torn-paper code* is provided. For a codeword $\vec{x}^n$, let $S_{\vec{x}^n}$ be the set of all possible unordered sets of fragments that can potentially be obtained such that each read has size within $[L_{\min}, L_{\max}]$. A $(L_{\min}, L_{\max})$-*single strand torn-paper code* has the property that any $\vec{x}_i^n$ and $\vec{x}_j^n$ such that $\vec{x}_i^n \neq \vec{x}_j^n$, $S_{\vec{x}_i^n}$ and $S_{\vec{x}_j^n}$ are disjoint. The following theorem is proved in [30] for the proposed construction:

Fig. 9. The figure shows a part of the codeword: It involves an index (in red), concatenated with a header (in green) and the information bits (in blue). The index and information part are constructed such that no zero runs of size $f(n)$ appear, so as it allow the decoder to identify the 0 runs of size $f(n)$, and thus find the position of the indices even if the tearing locations are random.

Theorem 3: If $L_{min} = a \log n + o(\log n)$ for some $a > 1$, then there exists a $(L_{\min}, L_{\max})$-*single strand torn-paper code* that achieves the zero-error capacity $1 - 1/a$. Moreover, there exists such a code with encoder and decoder that operates in run-time which is linear in n.

The encoding rule involves a two-step process. First, an index is generated in a careful manner using *Gray codes* [52]. Then it utilizes the concept of *Run-Length Limited Encoding* (RLL) [53], where the encoder receives a length-m string and converts it to a variable-length string (N) with the property that it does not contain zero runs of size greater than a chosen number $f(n)$. Specifically, in [30] it is shown that when $f(n) \coloneqq \sqrt{\log n}$ and $N \coloneqq L_{\min} - \alpha - f(n) - 2$, where α is the length of the encoded index, we can lower bound m as

$$m \geq L_{\min} - \log n - f(n) - \frac{\log n}{f(n) - 1} - 9 - \frac{2}{f(n) - 1} - \frac{a \log n}{2^{(f(n)-1)}}. \tag{27}$$

Now, a careful concatenation of the index, the RLL encoded string and the header (described in [30, Algorithm 1] and Figure 9), leads to a codebook which is a $(L_{\min}, L_{\max})$-single strand torn-paper code, which achieves a zero-error rate of $1 - 1/a$. Moreover, [30] shows that this achievable rate is the zero-error capacity of the adversarial TPC.

A $(L_{\min}, L_{\max})$-*single strand torn-paper code* can be uniquely decoded since, for any $\vec{x}_i^n \neq \vec{x}_j^n$, $S_{\vec{x}_i^n}$ and $S_{\vec{x}_j^n}$ are disjoint, which means that any fragmentation of a codeword into pieces with lengths in $[L_{\min}, L_{\max}]$ uniquely determines the codeword. Moreover, [30] proves that this decoding can be done in linear time. This code can also be extended to a *multi-strand* setting, where the input to the channel is a set of sequences as opposed to just a single sequence, and the output is the union of the fragments of all input sequences. This is motivated by the fact that DNA storage systems encode the information across a large number of DNA molecules, and the information is retrieved by sequencing the entire DNA pool. Furthermore, the scheme from [30] can be extended to a noisy setting, which models errors that occur during DNA synthesis and sequencing.

Comparison to the probabilistic TPC: To draw a comparison with the probabilistic setting, we define $C_{\text{ZE,aTPC}}$ as the zero-error capacity in the adversarial setting with fragment lengths in $[a \log n, b \log n]$ and C_{pTPC} as the (standard) capacity in the probabilistic setting where N_i has a distribution supported in $[a \log n, b \log n]$. From Theorem 3, we know that $C_{\text{ZE,aTPC}} = (1 - 1/a)^+$. Moreover, it is clear that $C_{\text{ZE,aTPC}} \leq C_{\text{pTPC}}$, as the adversarial setting represents a worst-case scenario.

Applying Theorem 1 to this channel, we establish that:

Corollary 1: For the probabilistic TPC with fragment length distribution supported in $[a \log n, b \log n]$,

$$C_{\mathrm{pTPC}} = \left(1 - \lim_{n\to\infty} \frac{\log n}{\ell_n}\right)^+. \tag{28}$$

Specifically, setting N_1 deterministically equal to $L_{\min} = a \log n$, we find $C_{\mathrm{pTPC}} = 1 - 1/a$, which implies $C_{\mathrm{ZE,aTPC}} \leq 1 - 1/a$. This provides a tight upper bound $C_{\mathrm{ZE,aTPC}} \leq (1 - 1/a)^+$ for the capacity $C_{\mathrm{ZE,aTPC}} = (1 - 1/a)^+$ established in [30]. We note that $C_{\mathrm{ZE,aTPC}} = 0$ when $L_{\min} < \log n$. Therefore we focus on the regime where $L_{\min} \geq \log n$. The gap between the rate achieved by this scheme and the capacity is given by

$$\lim_{n\to\infty} \left(\frac{\log n}{L_{\min}} - \frac{\log n}{\ell_n}\right) := \frac{1}{a} - \frac{1}{\bar{L}}, \tag{29}$$

where a is defined in Theorem 3 and $\bar{L}$ refers to the average length normalized by $\log n$. In cases where the mean piece length is close to $L_{\min}$, the scheme is more effective, as expected. When the distribution is uniform over $[L_{\min}, L_{\max}]$, the gap becomes $\frac{1}{a} - \frac{2}{a+b} = \frac{a-b}{a(a+b)}$, with $L_{\max} = b \log n + o(\log n)$, similar to $L_{\min}$. This suggests that this scheme is more effective when the minimum length of the pieces are longer, since the value of the rate gap is inversely dependent on a.

E. Embedded Indices for the Torn-Paper Channel

One can also adapt existing codes to suit the Torn-Paper Channel. For instance, Varshamov-Tenengolts (VT) codes were developed for wireless (Z and deletion) channels [54], [55] and later found applications in computer memory technology such as Racetrack [56]. Then, the authors in [31], [32] respectively present concatenated and nested versions of the VT codes for the TPC. These codes are defined as follows.

Definition 3 [32], [57]: For $0 \leq r \leq n$, the Varshamov-Tenengolts code, $\mathrm{VT}_r(n)$, is a set of binary encoded strings with length n, which is given by:

$$\mathrm{VT}_r(n) = \left\{\mathbf{x} \in \{0,1\}^n : \sum_{i=1}^{n} i x_i \equiv r \mod (n+1)\right\}, \tag{30}$$

where x_i is the i^{th} element of $\mathbf{x}$, the sum is evaluated as an ordinary integer summation, and r is referred to as the residue.

The key idea is to use the Varshamov-Tenengolts code structure as a way to embed the "index" within the codeword. In essence, [31], [32] use different residues in different parts of the overall code as embedded indices, which will be used later as signatures to put the fragments back together at the output of the TPC, and numerically show the error declines as the code lengths increase. However, there is no analytical analysis in these references to show the error of such designs vanishes as the block length tends to infinity nor have the presented codes been studied for TPC with lost pieces [22] or the SSC; nonetheless, the codes provide acceptable error rate and complexity in the context of the TPC.

IV. Concluding Remarks and Extensions

The disparity between the channel capacity and the rate achieved by the index-based scheme can be attributed to several factors. Firstly, the index-based approach necessitates the creation of pieces that exceed the size of $2 \log n$, whereas capacity-achieving random codes do not have this requirement. It is intriguing to speculate if the size requirement could be reduced to just $\log n$. Indeed, this is the case when the tears are deterministic and a priori known. The scheme can be reduced to just fixing indices for each piece. Since we know the tearing locations a priori, we can align these pieces to the pilot codeword easily. In [21], the authors show that we can achieve capacity when tearing locations are known, via index based schemes. This points to potential areas for further improvement in the structure of the index-based scheme. Secondly, for scenarios like the SSC, where overlaps allow for the merging of pieces, the bits used for indexing can become redundant, leading to inefficiencies. Hence, it may be possible to further optimize the integration of indexing information into the codewords. For instance, the development of an efficient scheme that forms "composite" codes, where a single bit carries both indexing and message information, could be a lead to potential improvement.

Specifically if we look at the random coding based achievable scheme described in Section II, the codes inherently contain both ordering and message information, without any specific requirement to separate the bits capturing this information. Therefore it might be of interest to develop codes that do not separate these bits. However it is currently unclear how we would do that efficiently.

The models considered in this manuscript capture specific aspects of DNA storage systems that lead to lack of ordering at the output. However, a complete model for a DNA storage system also needs to incorporate the per-base noise observed at the output sequences, which may be in the form of substitutions, insertions and deletions (commonly referred to as indels). Some aspects of these errors can modeled by concatenating traditional Binary Symmetric Channels or Binary Erasure Channels to the SSC or TPC.

For illustration, we briefly discuss one such model here (Figure 10). Let us consider a TPC with geometric tearing lengths, as considered in Section II. Before the string X^n is passed through this channel, it is passed through a BEC(p) channel, which erases each bit independently with probability p. Then it is passed through a TPC, and the process of tearing remains exactly the same. As mentioned before for simplicity, let us assume that $N_i \sim \mathrm{Geom}(1/\ell_n)$

Achievability via random coding: The achievable scheme is closely related to the achievable scheme discussed in Section II. Just as before we construct the codebook with i.i.d. $\mathrm{Bern}(1/2)$ entries and pick (without loss of generality) the codeword corresponding $W = 1$.

The decoding rule remains almost the same too. The difference is just that, since the reads are now corrupted with erasures, we will have to consider all codewords which have these modified reads as "potentially" correct substrings of the codeword, since the decoder does not know what the erased

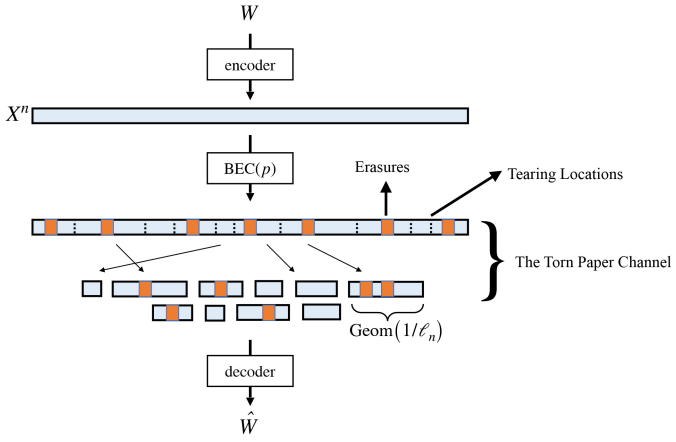

Fig. 10. A TPC with preceded by a Binary Erasure Channel with erasure probability p.

bits were. Moreover it is unclear if the decoder should throw out any of the reads. To circumvent this we say the decoder discards all reads of size $\gamma \log n$ and lower, where $\gamma \in [0, \infty)$ is a parameter. Let the new set obtained be $\mathcal{Y}'_\gamma$.

Following the same steps in calculating the probability of error, we can say

$$\begin{aligned}\Pr(\mathcal{E}) &= \Pr(\mathcal{E} \,|\, W = 1) \\ &= \Pr\big(\exists j \neq 1 : x_j \text{ covers all strings in } \mathcal{Y}'_\gamma \,\big|\, W = 1\big).\end{aligned} \tag{31}$$

Notice that, in the context of reads with erasures, "containing" a string in $\mathcal{Y}'_\gamma$ means that a segment from x_j matches the string in $\mathcal{Y}'_\gamma$ except for in the erased positions. Now, in order to analyze this error probability as we did in Section II-A, we need a handle on the number of bits from a codeword x_j that are "covered" by the strings in $\mathcal{Y}'_\gamma$. But in this case, "covered" refers to only positions of x_j that are covered by non-erased bits from strings in $\mathcal{Y}'_\gamma$. Let $Z_{\text{BEC},\gamma}$ be the random variable which counts the total fraction of erasures in all reads longer than $\gamma \log n$. We then have that

$$\Pr\big(|Z_{\text{BEC},\gamma} - p| > \epsilon p\big) \to 0, \tag{32}$$

as $n \to \infty$ for any $\gamma \in [0, \infty)$. This can be easily proved using basic concentration inequalities.

Now in a very similar way as in Section II, we condition on bad events that look at deviations from this concentration. We define $\alpha = 1/\bar{L}$ and c_γ as the total fraction of bits covered by the non-erased bits in the reads. By letting $B_1 = (1 + \epsilon)e^{-\alpha\gamma} n p_n$ and $B_2 = (1 - \epsilon)(\alpha\gamma + 1)e^{-\alpha\gamma}(1 - p)$, we have the bad events

$$\mathcal{B} = \left\{ \sum_{i=1}^{K} \mathbf{1}_{\{\vec{X}_i \in \mathcal{Y}'_\gamma\}} > B_1 \right\} \cup \Big\{ c_\gamma < B_2 \Big\}, \tag{33}$$

and $\Pr(\mathcal{B}) \to 0$ as $n \to \infty$. Therefore we can further bound $\Pr(\mathcal{E})$ as

$$\begin{aligned}\Pr(\mathcal{E}) &= \Pr\big(\exists x_j \neq x_1 \text{ covering all strings in } \mathcal{Y}' \,|\, W = 1\big) \\ &\leq \Pr\big(\exists x_j \neq x_1 \text{ covering all strings in } \mathcal{Y}' \,|\, W = 1, \overline{\mathcal{B}}\big) \\ &\quad + \Pr(\mathcal{B}) \overset{(a)}{\leq} |\mathcal{C}| \frac{n^{B_1}}{2^{nB_2}} + \Pr(\mathcal{B}) \\ &\leq 2^{nR - e^{-\alpha\gamma}[(1+\alpha\gamma)(1-p) - \alpha]} + o(1),\end{aligned}$$

which tends to 0 if (taking $\epsilon \to 0$)

$$R < e^{-\alpha\gamma}[(1 + \alpha\gamma)(1 - p) - \alpha]. \tag{34}$$

Now optimizing over $\gamma \in [0, \infty)$, we note that $\gamma = \frac{1}{1-p}$ gives the highest achievable rates. This implies that all rates R such that

$$R < e^{-\frac{1}{\bar{L}(1-p)}}(1 - p) \tag{35}$$

are achievable.

An interesting observation in the above decoding rule is that the optimal discarding threshold is $\gamma = \frac{1}{1-p}$. This agrees with our intuition in the following way: On average each read of, say, length N_1 contains only a $(1 - p)$ fraction of its original bits. Therefore we can think of each read having an "effective" length of $(1 - p)N_1$. In the noise-free case, we needed to discard all reads of size at most $\log n$. Now if we discard all pieces of "effective" length $\log n$, or true length $\log n/(1-p)$, we obtain the decoding rule described above.

It is currently unclear whether the achievable scheme described above is the capacity of this channel. Developing a converse for noisy out-of-order channels, even with a discrete memoryless noise channel such as the BEC noise described above, is generally quite difficult. Works such as [14], [36], [37] attempt to develop converse techniques for noisy shuffling channels, but they only work in limited parameter regimes and do not generalize to the systems considered here.

Also note that this scheme uses a random code, which is a highly inefficient computationally. One could consider implementing an index-based scheme for this system too. The problem, however, is that to align a read to its template codeword (containing only the pilot bits), we need the pilot bits to be preserved. This is not guaranteed in general for the channel described above, since pilot bits can be erased. Therefore, in the presence of noise, developing efficient codes for out-of-order channels that achieve rates close to the capacity is likely very challenging.

REFERENCES

[1] T. M. Cover and J. A. Thomas, *Elements of Information Theory*. Hoboken, NJ, USA: Wiley, 2012.

[2] M. Cheraghchi and J. Ribeiro, "Sharp analytical capacity upper bounds for sticky and related channels," *IEEE Trans. Inf. Theory*, vol. 65, no. 11, pp. 6950–6974, Nov. 2019.

[3] D. Tandjaoui, N. Badache, H. Bettahar, A. Bouabdallah, and H. Seba, "Performance enhancement of smooth handoff in mobile IP by reducing packets disorder," in *Proc. 8th IEEE Symp. Comput. Commun. (ISCC)*, 2003, pp. 149–154.

[4] D. Malak and O. B. Akan, "Molecular communication nanonetworks inside human body," *Nano Commun. Netw.*, vol. 3, no. 1, pp. 19–35, 2012.

[5] T. Nakano, A. W. Eckford, and T. Haraguchi, *Mol. Commun.*. Cambridge, U.K.: Cambridge Univ. Press, 2013.

[6] H. T. Yazdi, H. M. Kiah, E. Garcia-Ruiz, J. Ma, H. Zhao, and O. Milenkovic, "DNA-based storage: Trends and methods," *IEEE Trans. Mol., Biological Multi-Scale Commun.*, vol. 1, no. 3, pp. 230–248, Sep. 2015.

[7] I. Shomorony and R. Heckel, "DNA-based storage: Models and fundamental limits," *IEEE Trans. Inf. Theory*, vol. 67, no. 6, pp. 3675–3689, Jun. 2021.
[8] S. Pattabiraman, R. Gabrys, and O. Milenkovic, "Reconstruction and error-correction codes for polymer-based data storage," in *Proc. IEEE Inf. Theory Workshop (ITW)*, 2019, pp. 1–5.
[9] G. M. Church, Y. Gao, and S. Kosuri, "Next-generation digital information storage in DNA," *Science*, vol. 337, no. 6102, pp. 1628–1628, 2012.
[10] N. Goldman et al., "Towards practical, high-capacity, low-maintenance information storage in synthesized DNA," *Nature*, vol. 494, no. 7435, pp. 77–80, 2013.
[11] R. Grass, R. Heckel, M. Puddu, D. Paunescu, and W. J. Stark, "Robust chemical preservation of digital information on DNA in silica with error-correcting codes," *Angewandte Chemie Int. Ed.*, vol. 54, no. 8, pp. 2552–2555, 2015.
[12] H. T. Yazdi, R. Gabrys, and O. Milenkovic, "Portable and error-free DNA-based data storage," *Sci. Rep.*, vol. 7, no. 1, p. 5011, 2017.
[13] A. Lenz, P. H. Siegel, A. Wachter-Zeh, and E. Yaakohi, "Achieving the capacity of the DNA storage channel," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, 2020, pp. 8846–8850.
[14] N. Weinberger and N. Merhav, "The DNA storage channel: Capacity and error probability," 2021, *arXiv:2109.12549*.
[15] I. Shomorony, R. Heckel et al., "Information-theoretic foundations of DNA data storage," *Found. Trends® Commun. Inf. Theory*, vol. 19, no. 1, pp. 1–106, 2022.
[16] A. Makur, "Coding theorems for noisy permutation channels," *IEEE Trans. Inf. Theory*, vol. 66, no. 11, pp. 6723–6748, Nov. 2020. [Online]. Available: https://doi.org/10.1109
[17] J. Tang and Y. Polyanskiy, "Capacity of noisy permutation channels," 2021, *arXiv:2111.00559*.
[18] J. Tang and Y. Polyanskiy, "Capacity of noisy permutation channels," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, 2022, pp. 1987–1992.
[19] R. Heckel, I. Shomorony, K. Ramchandran, and D. N. C. Tse, "Fundamental limits of DNA storage systems," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, 2017, pp. 3130–3134.
[20] I. Shomorony and A. Vahid, "Communicating over the torn-paper channel," in *Proc. IEEE Glob. Commun. Conf.*, 2020, pp. 1–6.
[21] I. Shomorony and A. Vahid, "Torn-paper coding," *IEEE Trans. Inf. Theory*, vol. 67, no. 12, pp. 7904–7913, Dec. 2021.
[22] A. N. Ravi, A. Vahid, and I. Shomorony, "Capacity of the torn paper channel with lost pieces," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, 2021, pp. 1937–1942.
[23] A. N. Ravi, A. Vahid, and I. Shomorony, "Capacity of the shotgun sequencing channel," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, 2022, pp. 210–215.
[24] A. N. Ravi, A. Vahid, and I. Shomorony, "Coded shotgun sequencing," *IEEE J. Sel. Areas Inf. Theory*, vol. 3, no. 1, pp. 147–159, Mar. 2022.
[25] T. Laver et al., "Assessing the performance of the oxford nanopore technologies minion," *Biomol. Detect. Quantif.*, vol. 3, pp. 1–8, Mar. 2015.
[26] E. Brooks, M. Prusinowski, S. Gross, and T. Trejos, "Forensic physical fits in the trace evidence discipline: A review," *Forensic Sci. Int.*, vol. 313, Aug. 2020, Art. no. 110349. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0379073820302115
[27] C. Wang, J. Sima, and N. Raviv, "Break-resilient codes for forensic 3D fingerprinting," 2023.
[28] M. Schirmer, R. D'Amore, U. Z. Ijaz, N. Hall, and C. Quince, "Illumina error profiles: Resolving fine-scale variation in metagenomic sequencing data," *BMC Bioinf.*, vol. 17, p. 125, Mar. 2016.
[29] D. Sims, I. Sudbery, N. E. Ilott, A. Heger, and C. P. Ponting, "Sequencing depth and coverage: Key considerations in genomic analyses," *Nat. Rev. Genet.*, vol. 15, no. 2, pp. 121–132, 2014.
[30] D. Bar-Lev, S. Marcovich, E. Yaakobi, and Y. Yehezkeally, "Adversarial torn-paper codes," 2022, *arXiv:2201.11150v3*.
[31] S. Nassirpour and A. Vahid, "Embedded codes for reassembling non-overlapping random DNA fragments," *IEEE Trans. Mol., Biol. Multi-Scale Commun.*, vol. 7, no. 1, pp. 40–50, Mar. 2021.
[32] S. Nassirpour, I. Shomorony, and A. Vahid, "DNA merge-sort: A family of nested Varshamov-Tenengolts reassembly codes for out-of-order media," *IEEE Trans. Commun.*, vol. 72, no. 3, pp. 1303–1317, Mar. 2024.
[33] H. T. Yazdi, Y. Yuan, J. Ma, H. Zhao, and O. Milenkovic, "A rewritable, random-access DNA-based storage system," *Sci. Rep.*, vol. 5, Sep. 2015, Art. no. 14138.
[34] Y. Erlich and D. Zielinski, "DNA fountain enables a robust and efficient storage architecture," *Science*, vol. 355, no. 6328, pp. 950–954, 2017.
[35] L. Organick et al., "Random access in large-scale DNA data storage," *Nat. Biotechnol.*, vol. 36. pp. 242–248, Feb. 2018.
[36] I. Shomorony and R. Heckel, "Capacity results for the noisy shuffling channel," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, 2019, pp. 1–6.
[37] A. Lenz, P. Siegel, A. Wachter-Zeh, and E. Yaakobi, "An upper bound on the capacity of the DNA storage channel," in *Proc. IEEE Inf. Theory Workshop*, 2019, pp. 1–5.
[38] N. Weinberger and N. Merhav, "The DNA storage channel: Capacity and error probability bounds," *IEEE Trans. Inf. Theory*, vol. 68, no. 9, pp. 5657–5700, Sep. 2022.
[39] A. Makur, "Information capacity of BSC and BEC permutation channels," in *Proc. 56th Annu. Allerton Conf. Commun., Control, Comput. (Allerton)*, 2018, pp. 1112–1119.
[40] A. Tandon, V. Y. F. Tan, and L. R. Varshney, "The bee-identification problem: Bounds on the error exponent," *IEEE Trans. Commun.*, vol. 67, no. 11, pp. 7405–7416, Nov. 2019.
[41] H. M. Kiah, A. Vardy, and H. Yao, "Efficient algorithms for the bee-identification problem," *IEEE J. Sel. Areas Inf. Theory*, vol. 4, pp. 205–218, Jul. 2023.
[42] J. Chrisnata, H. M. Kiah, A. Vardy, and E. Yaakobi, "Bee identification problem for DNA strands," *IEEE J. Sel. Areas Inf. Theory*, vol. 4, pp. 190–204, Jul. 2023.
[43] S. Singhvi, A. Boruchovsky, H. M. Kiah, and E. Yaakobi, "Data-driven bee identification for DNA strands," 2023, *arXiv:2305.04597v1*.
[44] H. M. Kiah, G. J. Puleo, and O. Milenkovic, "Codes for DNA sequence profiles," *IEEE Trans. Inf. Theory*, vol. 62, no. 6, pp. 3125–3146, Jun. 2016.
[45] R. Gabrys, H. M. Kiah, and O. Milenkovic, "Asymmetric lee distance codes: New bounds and constructions," in *Proc. IEEE Inf. Theory Workshop (ITW)*, 2015, pp. 1–5.
[46] F. Sala, R. Gabrys, C. Schoeny, and L. Dolecek, "Exact reconstruction from insertions in synchronization codes," *IEEE Trans. Inf. Theory*, vol. 63, no. 4, pp. 2428–2445, Apr. 2017.
[47] J. Chrisnata, H. M. Kiah, and V. L. P. Pham, "Deletion correcting codes for efficient DNA synthesis," 2023, *arXiv:2305.07274*.
[48] J. M. Ruijter et al., "Amplification efficiency: Linking baseline and bias in the analysis of quantitative PCR data," *Nucl. Acids Res.*, vol. 37, no. 6, 2009, p. e45.
[49] N. G. De Bruijn, "A combinatorial problem," in *Proc. Koninklijke Nederlandse Academie van Wetenschappen*, vol. 49, 1946, pp. 758–764.
[50] M. A. Shokrollahi, "New sequences of linear time erasure codes approaching the channel capacity," in *Appl. Algebra, Algebraic Algorithms Error-Correcting Codes*, M. Fossorier, H. Imai, S. Lin, and A. Poli, Eds. Berlin, Heidelberg: Springer, 1999, pp. 65–76.
[51] E. Arikan, "Channel polarization: A method for constructing capacity-achieving codes for symmetric binary-input memoryless channels," *IEEE Trans. Inf. Theory*, vol. 55, no. 7, pp. 3051–3073, Jul. 2009.
[52] R. Doran, "The gray code," *J. Univ. Comput. Sci.*, vol. 13, pp. 1573–1597, Jan. 2007.
[53] M. Levy and E. Yaakobi, "Mutually uncorrelated codes for DNA storage," *IEEE Trans. Inf. Theory*, vol. 65, no. 6, pp. 3671–3691, Jun. 2019.
[54] R. Varshamov and G. Tenengolts, "Codes which correct single asymmetric errors (in Russian)," *Automatika i Telemkhanika*, vol. 161, no. 3, pp. 288–292, 1965.
[55] V. I. Levenshtein, "Binary codes capable of correcting deletions, insertions, and reversals," *Sov. Phys. Doklady*, vol. 10, no. 8, 1966, pp. 707–710.
[56] G. Mappouras, A. Vahid, R. Calderbank, and D. J. Sorin, "GreenFlag: Protecting 3D-racetrack memory from shift errors," in *Proc. 49th Annu. IEEE/IFIP Int. Conf. Dependable Syst. Netw. (DSN)*, 2019, pp. 1–12.
[57] N. J. Sloane, "On single-deletion-correcting codes," in *Proc. Conf. Honor. Profess. Dijen K. Ray-Chaudhuri Occas. 65th Birth.*, 2002, pp. 273–291.

Aditya Narayan Ravi (Graduate Student Member, IEEE) received the B.Tech. and M.Tech. degrees from the Department of Electrical Engineering, Indian Institute of Technology Bombay in 2020. He is currently pursuing the graduate degree with the University of Illinois at Urbana–Champaign. His current research interests are in information theory and its applications, particularly to computational biology.

Alireza Vahid (Senior Member, IEEE) received the B.Sc. degree in electrical engineering from the Sharif University of Technology, Tehran, Iran, in 2009, and the M.Sc. and Ph.D. degrees in electrical and computer engineering from Cornell University, Ithaca, NY, USA, in 2012 and 2015, respectively. From 2015 to 2017, he worked as a Postdoctoral Research Scientist with the Information Initiative, Duke University, Durham, NC, USA. From 2017 to 2023, he was an Assistant Professor of Electrical Engineering with the University of Colorado at Denver, Denver, CO, USA. He is currently a Gleason Endowed Associate Professor of Electrical and Microelectronic Engineering with the Rochester Institute of Technology, Rochester, NY, USA. His research interests include network information theory, wireless communications, and applications of coding theory in high-performance computer memory systems. He received the 2015 Outstanding Ph.D. Thesis Research Award, the 2010 Director's Ph.D. Teaching Award, the Jacobs Scholar Fellowship in 2009 from Cornell University, the 2013 Qualcomm Innovation Fellowship, the 2019 Lab Venture Challenge Award, and the 2021 SONY Faculty Innovation Award. He currently serves as an Associate Editor for IEEE COMMUNICATIONS LETTERS.

Ilan Shomorony (Member, IEEE) received the Ph.D. degree in electrical and computer engineering from Cornell University in 2014. He was a Postdoctoral Scholar with the University of California at Berkeley through the NSF Center for Science of Information until 2017. After that, he spent a year working as a Researcher and a Data Scientist with Human Longevity Inc., a personal genomics company. He is currently an Assistant Professor of Electrical and Computer Engineering with the University of Illinois at Urbana–Champaign, where he is a member of the Coordinated Science Laboratory. His research interests include information theory, communications, and computational biology. He received the NSF CAREER Award in 2021.