Capacity of the Shotgun Sequencing Channel

Aditya Narayan Ravi
University of Illinois, Urbana-Champaign
anravi2@illinois.edu

Alireza Vahid
University of Colorado, Denver
alireza.vahid@ucdenver.edu

Ilan Shomorony
University of Illinois, Urbana-Champaign
ilans@illinois.edu

2022 IEEE International Symposium on Information Theory (ISIT) | 978-1-6654-2159-1/22/$31.00 ©2022 IEEE | DOI: 10.1109/ISIT50566.2022.9834409

***Abstract*—Most DNA sequencing technologies are based on the shotgun paradigm: many short reads are obtained from random unknown locations in the DNA sequence. A fundamental question, studied in [1], is what read length and coverage depth (i.e., the total number of reads) are needed to guarantee reliable sequence reconstruction. Motivated by DNA-based storage, we study the coded version of this problem; i.e., the scenario in which the DNA molecule being sequenced is a codeword from a predefined codebook. Our main result is an exact characterization of the capacity of the resulting *shotgun sequencing channel* as a function of the read length and coverage depth. In particular, our results imply that while in the uncoded case, $O(n)$ reads of length greater than $2\log n$ are needed for reliable reconstruction of a length-n binary sequence, in the coded case, only $O(n/\log n)$ reads of length greater than $\log n$ are needed for the capacity to be arbitrarily close to 1.**

***Index Terms*—DNA shotgun sequencing, DNA-based storage, channel capacity**

I. Introduction

At the heart of the DNA sequencing revolution was the development of *high-throughput shotgun sequencing* platforms. These platforms extract a large number of short reads from random locations of the target DNA sequence (e.g., the genome of an organism), in a massively parallel fashion. Sequencing must then be followed by an *assembly* step, where the reads are merged together based on regions of overlap with the intention of reconstructing the original DNA sequence.

A natural question thus arises: When is it possible, from an information-theoretic standpoint, to reconstruct a sequence from a random set of its substrings? Formalizing this question, suppose we observe K random reads (i.e., substrings) of length L from an unknown length-n sequence x^n. What conditions on x^n, K and L guarantee that x^n can be reliably reconstructed from the observed reads? This problem was first studied from an information-theoretic point of view in [1]. The authors considered the asymptotic regime where $n \to \infty$ and the read length L scales as $L = \bar{L}\log n$, for a constant $\bar{L}$. They also defined $c = KL/n$ to be the *coverage depth*; i.e., the average number of times each symbol in x^n is sequenced. This appropriate scaling of the read length allowed the authors of [1] to show a surprising critical phenomenon: if x^n is an i.i.d. Ber$(1/2)$ sequence, when $\bar{L} < 2$, reconstruction is impossible for any coverage depth c, but if $\bar{L} > 2$, reconstruction is possible as long as the coverage depth is at least $c_{LW} = \ln(n/\epsilon)$, which is the Lander-Waterman coverage [2], i.e. the minimum coverage needed to guarantee that all symbols in x^n are sequenced at least once with probability $1-\epsilon$. The result in [1] established a *feasibility region* for the shotgun sequencing problem, illustrated in blue in Figure 1(a).

One key aspect about the framework studied in [1] is that the sequence x^n is chosen "by nature" [1, 3, 4]). However, in recent years, significant advances in DNA synthesis technologies have enabled the idea of storing data in DNA, and several groups demonstrated working DNA-based storage systems [5–11]. In these systems, information was encoded into DNA molecules via state-of-the-art synthesis techniques, and later retrieved via sequencing. This emerging technology motivates the following question: How do the fundamental limits of shotgun sequencing from [1] change in the coded setting where x^n is chosen from a codebook?

Motivated by this question, in this paper we introduce the Shotgun Sequencing Channel (SSC). As illustrated in Figure 1(b), the channel input is a (binary) length-n sequence x^n, and the channel output are K random reads of length L from x^n. Each read is assumed to be drawn independently and uniformly at random from x^n and we consider the read length scaling as $L = \bar{L}\log n$. Notice that this is essentially the same setup as in [1], except that the "genome" x^n is chosen from a codebook rather than decided by nature. In this paper, we characterize the capacity of this channel, provide an achievable scheme that achieves this capacity and a converse that matches the achievable rates. Some details are deferred to a longer version of the paper [13].

In order to build intuition it is worth considering the related setting of the *shuffling-sampling channel* [12], illustrated in Figure 1(c). In this case the input are M strings of length L, and the output are K strings, each chosen uniformly at random from the set of input strings. If we define the coverage depth for this setting as $c = \frac{KL}{ML} = K/M$, the result in [12] implies that, for $\bar{L} > 1$, the capacity of this channel is

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

and $C_{\text{shuf}} = 0$ for $\bar{L} \leq 1$. The term $(1-e^{-c})$ captures the loss due to unseen input strings and $(1-1/\bar{L})$ captures the loss due to the unordered nature of the output strings (which becomes more severe the shorter the strings are).

Intuitively, the capacity of the SSC should depend on c and $\bar{L}$ in a similar way as in (1). The expected fraction of symbols in x^n that are read at least once can be shown to be $1-e^{-c}$, which provides an upper bound to the capacity of the SSC. But it is not clear a priori which of the channels in Figure 1(b,c) should have the larger capacity. Our main result establishes

Authorized licensed use limited to: INDRAPRASTHA INSTITUTE OF INFORMATION TECHNOLOGY. Downloaded on May 15,2023 at 07:36:42 UTC from IEEE Xplore. Restrictions apply.

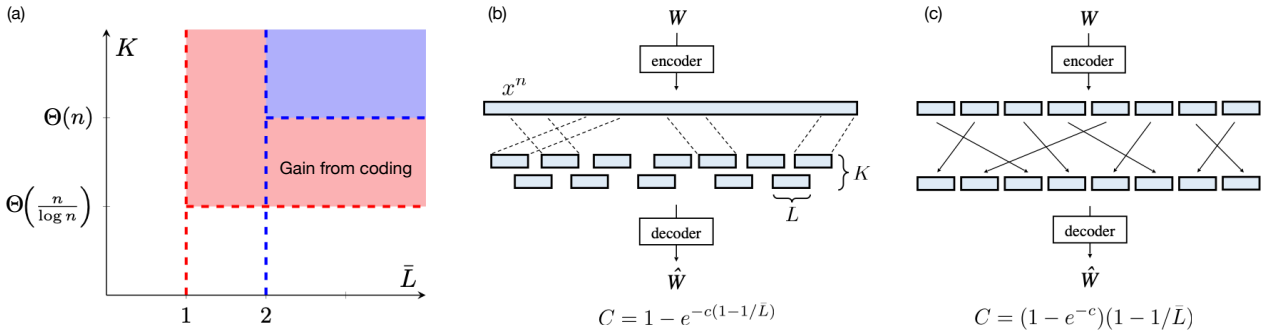

Fig. 1. (a) The feasibility region for reconstruction of a string under the uncoded setting (blue) and the coded setting (red) (b) The Shotgun Sequencing Channel (SSC) and the (c) Shuffling-Sampling channel from [12]. The input to the SSC is a single (binary) string x^n and the output are K random substrings of length L. In the Shuffling-Sampling channel, the input are M strings of length L, which are sampled with replacement to produce the channel output. Both capacity expressions can be written in terms of the expected coverage depth c and the normalized read length $\bar{L}$.

that, for $\bar{L} \geq 1$, the capacity of the SSC is given by

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

Notice that the dependence on $\bar{L}$ appears as the term $(1-1/\bar{L})$ in the exponent and, as $c \to \infty$, $C_{\text{SSC}} \to 1$ for any $\bar{L} > 1$. This is in contrast to the shuffling-sampling channel, where $C_{\text{shuf}} \to 1 - 1/\bar{L}$ as we increase the coverage depth c to infinity. Therefore, even in the high coverage depth regime, if $\bar{L} \approx 1$, $C_{\text{shuf}} \approx 0$. Furthermore, it can be verified that $C_{\text{shuf}} < C_{\text{SSC}}$ for any c and $\bar{L}$, establishing the advantage (from a capacity standpoint) of storing data on a long molecule of DNA as opposed to on many short molecules.

The above result also allows for an interesting comparison with the uncoded setting (i.e., the genome sequencing problem) of [1]. When we allow coding over the string, the critical threshold on the read length reduces to $\bar{L} > 1$, compared to $\bar{L} > 2$ for the uncoded setting. Moreover, in the SSC it is possible to achieve a capacity close to 1 by having the coverage depth be a large constant, while in the uncoded case the coverage c needs to grow as $\log n$.

Background and Related Work: The idea of coding over a set of strings that are shuffled and sampled was studied in several settings [12, 14–19]. Many works have focused on developing explicit codes tailored to specific aspects of DNA storage. These include DNA synthesis constraints such as sequence composition [8, 9, 20], the asymmetric nature of the DNA sequencing error channel [21], the need for codes that correct insertion errors [22], and the need for techniques to allow random access [8].

Several works studied the problem of genome sequencing and assembly from an information-theoretic standpoint [1, 3, 4, 23]. The trace reconstruction problem is another related setting where one observes (non-contiguous) subsequences of the input sequence and attempt to reconstruct it [24–26].

A very relevant related setting is the problem of reconstructing a string from its substring spectrum [27, 28]. We discuss the differences between the methods used in this setting and the setting considered in this paper in Section V.

II. Problem Setting

We consider the Shotgun Sequencing Channel (SSC), shown in Figure 1(b). The transmitter sends a length-n binary string $X^n \in \{0,1\}^n$, corresponding to a message $W \in [1 : 2^{nR}]$. The channel output is a set of length-L binary strings $\mathcal{Y}$. The channel chooses K starting points uniformly at random, represented by the random vector $T^K \in [1 : n]^K$. The vector T^K is assumed to be sorted in a non-decreasing order. Length-L reads are then sampled with T_i, $i = 1, \ldots, K$ as their starting points. 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. The unordered multi-set $\mathcal{Y} = \{\vec{Y}_1, \vec{Y}_2, \ldots, \vec{Y}_K\}$ of reads resulting from this sampling process is the channel output. The expected number of times a given symbol from X^n is sequenced is defined as the coverage depth c. This is given by the expression $c := KL/n$.

We focus on the regime where the length of the reads sampled is much smaller than the block length n. In particular, as shown in previous works [12, 29–33], the regime $L = \Theta(\log n)$ is of interest from a capacity standpoint. Hence, as in [1], we fix a normalized length $\bar{L}$ and define $L := \bar{L} \log n$.

Notice that, in this regime, the total number of reads is $K = cn/(\bar{L} \log n) = \Theta(n/\log n)$, which is a $\log n$ factor smaller than what is needed in the uncoded setting from [1]. We define achievable rates and capacity in the usual way [34].

Notation: $\log(\cdot)$ represents the logarithm in base 2. For functions $a(n)$ and $b(n)$, we say $a(n) = o(b(n))$ if $a(n)/b(n) \to 0$ as $n \to \infty$ and $a(n) = O(b(n))$ if $a(n)/b(n) < \infty$ as $n \to \infty$. Further, we say that a function $a(n) = \Theta(f(n))$ if there exist $n_0 \in \mathbb{N}, k_1, k_2 \in (0, \infty)$, such that $k_1 f(n) \leq a(n) \leq k_2 f(n)$ $\forall n \geq n_0$. For an event A, we let $\mathbf{1}_A$ be the binary indicator of A. For a set B, $|B|$ indicates the cardinality of that set. For any a, $a^+ = \max(a, 0)$.

III. Main Results

The DNA storage problem considered here has two important properties: (i) the reads in general overlap with each other and (ii) the set of reads is unordered. Property (i) was

Authorized licensed use limited to: INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY. Downloaded on May 15,2023 at 07:36:42 UTC from IEEE Xplore. Restrictions apply.

explored in the context of genome sequencing [1]. Intuitively, the overlaps between the reads allow them to be merged in order to reconstruct longer substrings of X^n. Property (ii) has been analyzed before in the context of several works on DNA storage. In particular, in the context of the shuffling-sampling channel from [12], illustrated in Figure 1(c), the input to the channel is a set of strings of length L, and the capacity is given by $C_{\text{shuf}} = (1 - e^{-c})(1 - 1/\bar{L})$.

Notice that, in the case of the shuffling-sampling channel, the output strings have no overlaps (they can only be non-overlapping or identical). In the context of the SSC, on the other hand, the overlaps can provide useful information to fight the lack of ordering of the output strings. Our main result captures the capacity gains that can be achieved by optimally exploiting the overlaps, for any coverage depth c and normalized read length $\bar{L}$.

Theorem 1. *For any $c > 0$ and $\bar{L} > 0$, the capacity of the Shotgun Sequencing Channel is*

$$C = \left(1 - e^{-c\left(1-1/\bar{L}\right)}\right)^{+}. \tag{3}$$

Notice that the capacity of the SSC given in Theorem 1 is zero when $\bar{L} < 1$. An intuitive reason for this is that when $\bar{L} < 1$, the number of possible distinct length-L sequences is just $2^{\bar{L}\log n} = n^{\bar{L}} = o(n/\log n) = o(K)$, and many reads must be identical. This can be used to show that the decoder cannot discern any meaningful information from $\mathcal{Y}$. The longer version of this paper further discusses this issue [13].

In order to interpret the capacity expression in (3) notice that the probability that a given symbol in X^n is not sequenced by any of the K reads is $(1 - L/n)^K = (1 - L/n)^{cn/L} \to e^{-c}$, as $n \to \infty$. Hence the expected fraction of symbols in X^n covered by at least one read is asymptotically close to $1 - e^{-c}$.

If instead of reads of length $L = \bar{L}\log n$ we had reads of length $(\bar{L} - 1)\log n$, the new coverage depth would be $c' = K(\bar{L}-1)\log n/n = c(1-1/\bar{L})$, and the expected fraction of symbols in X^n that would be sequenced would be $1 - e^{-c'} = 1 - e^{-c(1-1/\bar{L})}$. Hence, the capacity expression in Theorem 1 suggests that, on average, $\log n$ bits from each read are used for ordering information, while the remaining $(\bar{L} - 1)\log n$ bits provide new data information.

We prove the achievability of Theorem 1 in the next section.

IV. Achievability

We use a random coding argument to prove the achievability of Theorem 1. We generate a codebook with 2^{nR} codewords of length n, independently picking each letter Ber$(1/2)$. Let the codebook be $\mathcal{C} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{2^{nR}}\}$. The encoder chooses the codeword corresponding to the message $W \in [1 : 2^{nR}]$, and sends $\mathbf{x}_W$ across the Shotgun Sequencing Channel. The output, $\mathcal{Y}$, is presented to the decoder. For the analysis and without loss of generality, we assume $W = 1$.

The optimal decoder looks for a codeword that contains all the reads in $\mathcal{Y}$. Analyzing the error probability of this optimal decoder, however, is hard. We therefore develop a decoding rule that is simple enough to analyze. It turns out that it is critical to consider the overlaps between reads to do this. Unfortunately, merging reads is not a straightforward process because reads $\vec{Y}_i$ and $\vec{Y}_j$ may have an overlap even if they do not correspond to overlapping segments of X^n. In general, the merging process will be prone to errors and we need to develop a decoding algorithm that considers merges in a careful way.

In Section II, we defined the unknown vector T^K to be the ordered starting positions of the reads in $\mathcal{Y}$. Thus, without loss of generality we assume that $\vec{Y}_i$ starts at T_i. We define the *successor* of $\vec{Y}_i$ as $\vec{Y}_{i+1}$. We assume Y_1 is the successor of Y_K. Now we need a consistent definition to characterize how large the overlap of a given read is.

Definition 1. *(Overlap size) The overlap size of a read is defined as the number of bits the suffix of the read shares with its successor. It has an overlap size of 0 if no bits are shared (i.e., if a read and its successor have no overlap).*

The above definition implies that the overlap size of $\vec{Y}_i$ is $(L - (T_{i+1} - T_i))^+$. Notice that some reads might share some of their prefix bits with a predecessor read, but we do not consider this a contribution to the overlap size of that read. Intuitively speaking, since each bit of the string X^n was generated independently as a Ber$(1/2)$ random variable, we would expect larger overlap sizes to be easily discerned as compared to smaller ones. Therefore, we would need to know: (a) how many pieces exist of particular overlap sizes, and (b) given an overlap size, how "easy" it is to merge a read with its successor.

To handle (a), we define $G(\gamma)$ as a random variable that counts the number of reads with an overlap size of $\gamma \log n$, where $\gamma \in \Gamma := \left[\frac{1}{\log n}, \frac{2}{\log n}, \ldots, \bar{L}\right]$. Thus, γ is chosen from a finite set that depends on n. We can say $G(\gamma) := \sum_{i=1}^{K} G(\gamma)_i$, where $G(\gamma)_i := \mathbf{1}_{\{\vec{Y}_i \text{ has an overlap size of } \gamma \log n\}}$.

To capture (b), given a binary string $\vec{z}$, we define the random variable $M_{\vec{z}}$ as the number of times $\vec{z}$ appears as the prefix of a read in $\mathcal{Y}$. Note that the length of $\vec{z}$ is in $[1 : \bar{L}\log n]$. Let $\mathcal{Z}$ be the set of all binary strings with lengths in $[1 : \bar{L}\log n]$.

First let's define a quantity that captures the fraction of total bits that are actually sequenced. We say that the ith bit of X^n is *covered* if there is a read with starting position in $\{i - L + 1, i - L + 2, \ldots, i\}$, where the indices wrap around X^n. We then define the *coverage* as the random variable $\Phi = \frac{1}{n}\sum_{i=1}^{n} \mathbf{1}_{\{i\text{th bit is covered}\}}$.

Lemma 1. *(Coverage) For any $\epsilon > 0$, the coverage Φ satisfies*

$$\Pr\left(\left|\Phi - \left(1 - e^{-c}\right)\right| > \epsilon\left(1 - e^{-c}\right)\right) \to 0, \tag{4}$$

as $n \to \infty$.

The proof of this lemma is available in the longer verison of this paper [13]. If we can identify the merges correctly, we are left with a set of variable-length strings called islands, which we formally define next.

Definition 2. *(Islands) The set of non-overlapping substrings*

Authorized licensed use limited to: INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY. Downloaded on May 15,2023 at 07:36:42 UTC from IEEE Xplore. Restrictions apply.

that are obtained after merging all the reads to their successors based on positive overlap sizes are called islands.

Let K' be the number of islands. Then, we have the following concentration result for K'.

Lemma 2. *(Number of islands) For any $\epsilon > 0$, as $n \to \infty$, the number of islands K' satisfies*

$$\Pr\left(\left|K' - Ke^{-c}\right| \geq \epsilon K e^{-c}\right) \to 0. \tag{5}$$

The proof of this lemma is available in the longer verison of this paper [13]. Similar to the previous lemma, Lemma 2 guarantees that the number of islands K' is concentrated around its expectation Ke^{-c}. However, to use these results, the decoder would first need to obtain the non-overlapping islands (the decoder only has the reads, not their locations). The following lemmas give the decoder some guidelines on how to construct these islands from the reads.

Lemma 3. *(Number of potential overlaps) For any $\epsilon > 0$,*

$$\Pr\left(\bigcup_{z\in\mathcal{Z}:\gamma(\vec{z})\leq 1-\epsilon}\left\{\left|M_{\vec{z}} - Kn^{-\gamma(\vec{z})}\right| \geq \epsilon K n^{-\gamma(\vec{z})}\right\}\right) \to 0,$$

$$\Pr\left(\bigcup_{z\in\mathcal{Z}:\gamma(\vec{z})>1-\epsilon}\left\{M_{\vec{z}} \geq n^{\epsilon}\right\}\right) \to 0, \tag{6}$$

as $n \to \infty$, where we define $\gamma(\vec{z}) := |z|/\log n$.

Lemma 3 considers two separate cases for binary strings based on their length. For strings $\vec{z}$ with length at most $(1-\epsilon)\log n$, Lemma 3 states that $M_{\vec{z}}$ is close to its mean $Kn^{-\gamma(\vec{z})} = cn^{1-\gamma(\vec{z})}/\bar{L}\log n$. For strings $\vec{z}$ with length greater than $(1-\epsilon)\log n$, Lemma 3 states that $M_{\vec{z}} < n^{\epsilon}$ with high probability.

Lemma 4. *(Number of reads of a given overlap size) For all $\epsilon > 0$,*

$$\Pr\left(\bigcup_{\gamma\in\Gamma}\left\{|G(\gamma) - \bar{G}(\gamma)| \geq \epsilon\bar{G}(\gamma)\right\}\right) \to 0, \tag{7}$$

as $n \to \infty$, where $\bar{G}(\gamma) := E[G(\gamma)]$.

Lemma 4 give us a handle on the expected number of overlaps of each size, which will be used by the decoder when trying to construct the islands from the reads. Lemmas 3 and 4 are proved in the longer version of this paper [13].

The decoding procedure starts with a brute-force search over ways to merge the reads into islands, which we refer to as the Partition and Merge (PM) algorithm. First, the decoder considers all possible partitions of the reads into L groups, by assigning potential overlap sizes to each read. This can be done by looking at all ways of assigning a number in $[0:L]$ to each of the reads. To make this precise, we can look at all possible vectors of the form $\vec{p} := (p_1, p_2, \ldots, p_K) \in [0:L]^K$ and call them *partition vectors*. Each element p_i of the vector corresponds to an assigned overlap size of read $\vec{Y}_{\sigma(i)}$ for some permutation σ of the elements of $\mathcal{Y}$. Thus, each partition vector along with a permutation σ can be viewed as assigning an overlap size to each read. It is easy to see the total number of such partition vectors (and hence the total possible partitions) will be $P := (L+1)^K$.

Rather than considering all P partitions, we will only consider partitions that satisfy the bounds implied by Lemmas 1–4. To make this requirement precise, we define for a partition vector $\vec{p}$, $G(\vec{p},\gamma)$ to be the number of reads in $\mathcal{Y}$ that would have an overlap size of $\gamma\log n$ according to partition vector $\vec{p}$, which can be written as $G(\vec{p},\gamma) = |\{i : p_i = \gamma\log n\}|$.

Note that since the number of potential islands is exactly equal to the number of reads with overlap size zero, the total number of islands according to $\vec{p}$ is $G(\vec{p},0)$. Moreover we define $\Phi(\vec{p})$ as the total coverage of the reads according to $\vec{p}$, which is given by $\Phi(\vec{p}) := KL - \sum_{i=1}^{K} p_i$.

We then define the set $\mathcal{P}$ as the set of all $\vec{p}$ such that (for a fixed $\epsilon > 0$): $|\Phi(\vec{p}) - (1-e^{-c})| \leq \epsilon(1-e^{-c})$ (i.e., coverage is close to expected coverage), $|G(\vec{p},0) - Ke^{-c}| \leq \epsilon Ke^{-c}$ (i.e., number of islands is close to expected number of islands) and $|G(\vec{p},\gamma) - \bar{G}(\gamma)| \leq \epsilon\bar{G}(\gamma)$ for all $\gamma \in \Gamma$ (i.e., number of reads with overlap size $\gamma\log n$ is close to the expected number). Therefore, $\mathcal{P}$ restricts the total number of partition vectors to a smaller set of partition vectors that are admissible according to Lemmas 1, 2 and 4.

Now, for each partition vector $\vec{p} \in \mathcal{P}$, we take all possible $K!$ permutations σ of the reads. For each permutation, all of the reads are compared to their successors. If every read can be successfully merged with its successor with the assigned overlap size, we retain the set of substrings formed after these merges as a Candidate Island set, and add it to the set CI. Notice that CI is a set of sets of variable-length strings. This procedure is summarized in Figure 2 and Algorithm 1. After completion of Algorithm 1, the decoder checks, for each set of candidate islands in CI, whether there exists a codeword that contains all the candidate islands as substrings. If only one such codeword is found, the decoder outputs its index. Otherwise, an error is declared.

Algorithm 1: Partition and Merge

```
for each partition vector p⃗ ∈ P do
    for each permutation σ of [1 : K] do
        check if suffix of length p_i of Y⃗_{σ(i)} matches
          prefix of Y⃗_{σ(i+1)}, for i = 1, ..., K
        if prefix and suffix match for i = 1, ..., K then
            Merge reads according to overlaps
            Add set of resulting islands to CI
return CI
```

Let the event that the decoder makes an error be $\mathcal{E}$. An error occurs if more than one codeword contains any of the CI sets as substrings. We define $B_1 := (1+\epsilon)Ke^{-c}$, $B_2 := (1-\epsilon)(1-e^{-c})$, $B_3(\gamma) := (1+\epsilon)n^{1-\gamma}$ for $\gamma \leq 1-\epsilon$, $B_3(\gamma) = n^{\epsilon}$ for $\gamma > 1-\epsilon$ and $B_4(\gamma) := (1+\epsilon)\bar{G}(\gamma)$, and we define the corresponding undesired events as: $\mathcal{B}_1 = \{K' > B_1\}, \mathcal{B}_2 = \{\Phi < B_2\}, \mathcal{B}_3 = \bigcup_{\vec{z}\in\mathcal{Z}}\{M_{\vec{z}} > B_3(\gamma(\vec{z}))\}, \mathcal{B}_4 =$

Authorized licensed use limited to: INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY. Downloaded on May 15,2023 at 07:36:42 UTC from IEEE Xplore. Restrictions apply.

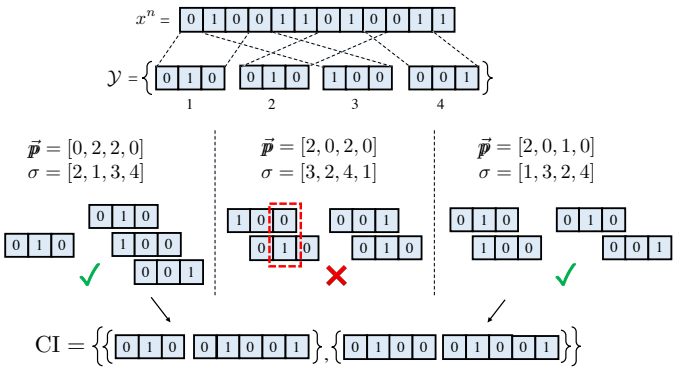

Fig. 2. The decoder receives the shotgun sequenced reads $\mathcal{Y}$ and performs the Partition and Merge procedure. For each partition $\vec{p} \in [0 : L]^K$ of the K reads according to overlap size, and each ordering of the reads σ, the decoder attempts to merge the reads into islands based on σ and $\vec{p}$. The figure shows this procedure for three choices of $\vec{p}$ (out of $(1+L)^K$) and three choices of σ (out of $K!$). If the merging of all reads is successful for some p and σ, the set of resulting islands is added to the set CI.

$\bigcup_{\gamma \in \Gamma} \{G(\gamma) > B_4(\gamma)\}$.

From Lemmas 1, 2, 3 and 4, we let $\mathcal{B} = \mathcal{B}_1 \cup \mathcal{B}_2 \cup \mathcal{B}_3 \cup \mathcal{B}_4$, and we have $\Pr(\mathcal{B}) \to 0$. Note that conditioned on $\bar{\mathcal{B}}$, we are guaranteed that the CI set outputs the true island set. This is because exactly one partition and one arrangement given that partition correspond to the true order in which the reads were sampled. Before we use this to bound the probability of error, notice that the error event depends on the total number of CI sets output by the PM algorithm. In general, this is not a deterministic value. We will define $\overline{CI}_n$ as an upper bound on the number of CI sets conditioned on $\bar{\mathcal{B}}$. We claim that conditioned on $\bar{\mathcal{B}}$, after the PM algorithm, the resulting CI (which is a set of sets of binary strings) satisfies

$$|\text{CI}| \leq P \times \prod_{\gamma \leq 1-\epsilon} B_3(\gamma)^{B_4(\gamma)} \prod_{\gamma > 1-\epsilon} n^{\epsilon B_4(\gamma)} := \overline{CI}_n. \quad (8)$$

To see this, first we notice that $|\mathcal{P}| \leq P$. According to a given partition vector $\vec{p} \in \mathcal{P}$, there are at most $B_4(\gamma)$ reads with overlap size $\gamma \log n$. Given a read $\vec{Y}_i$ with assigned overlap size $\gamma \log n$, when $\gamma \leq 1 - \epsilon$, there are at most $B_3(\gamma)$ reads whose prefix matches the $(\gamma \log n)$-suffix of $\vec{Y}_i$ and, therefore, at most $B_3(\gamma)$ potential valid merges. Therefore, for a given overlap size $\gamma \log n$, $\gamma \leq 1$, there at most $B_3(\gamma)^{B_4(\gamma)}$ merge possibilities. However, when $\gamma > 1 - \epsilon$, we know that for the given read, there at most n^ϵ potential valid merges. Therefore there are at most $n^{\epsilon B_4(\gamma)}$ merge possibilities. We thus bound the probability of error averaged over all codebooks as

$$\begin{aligned}
&\Pr(\mathcal{E}) = \Pr(\mathcal{E}|W=1) \leq \Pr(\mathcal{E}|W=1, \bar{\mathcal{B}}) + \Pr(\mathcal{B}) \\
&\overset{(a)}{\leq} 2^{nR} \times \overline{CI}_n \times n^{B_1} \times \frac{1}{2^{nB_2}} + o(1) \\
&= 2^{nR + \log \overline{CI}_n + B_1 \log n - nB_2} + o(1) \\
&= 2^{nR + \log \overline{CI}_n + (1+\epsilon) K e^{-c} \log n - n(1-\epsilon)(1-e^{-c})} + o(1). \quad (9)
\end{aligned}$$

This follows because an error occurs if any of the $2^{nR} - 1$ codewords ($W \neq 1$) contain any of the sets of candidate islands in CI (which is upper bounded by $\overline{CI}_n$ when conditioned on $\bar{\mathcal{B}}$). Each of the sets in CI contains at most B_1 islands and a total island length of at least nB_2 bits. Hence, there are at most n^{B_1} ways to arrange the islands on a codeword and, given one such arrangement, an erroneous codeword must match these islands on at least nB_2 bits.

In order to have $\Pr(\mathcal{E}) \to 0$ in (9), we require

$$R \leq (1-\epsilon)(1-e^{-c}) - (1+\epsilon)\frac{ce^{-c}}{\bar{L}} - \lim_{n\to\infty} \frac{1}{n} \log \overline{CI}_n.$$

The following lemma, proved in the longer version of the paper [13], evaluates the last term in the above expression.

Lemma 5. *The upper bound $\overline{CI}_n$ on $|\text{CI}|$ satisfies*

$$\lim_{n\to\infty} \frac{1}{n} \log \overline{CI}_n \leq e^{-c\left(1-\frac{1}{L}\right)} - \left(\frac{c}{\bar{L}} + 1\right) e^{-c} + f(\epsilon),$$

where $f(\epsilon) \to 0$ as $\epsilon \to 0$.

From Lemma 5, if we let $\epsilon \to 0^+$, we have that any rate

$$R < 1 - e^{-c\left(1-\frac{1}{L}\right)},$$

is achievable. This completes the proof of achievability. The converse is considered in the longer version of this paper [13].

V. Discussion

We presented a closed form capacity expression for the SSC. We provide an achievability argument based on merging reads in an intricate way, which allowed us to achieve capacity.

A relevant related setting is the problem of reconstructing a string from its substring spectrum [27, 28]. In that setting, the goal is to provide explicit code constructions that allow the reconstruction of any codeword from a noisy version of the set of its length-L substrings. The reconstruction guarantees are in terms of the maximum number of errors and missing substrings. This is in contrast to the probabilistic setting we consider, for which one can characterize the capacity.

Many of the explicit code constructions that have been proposed are related to the class of *repeat-free* codes [35, 36]. The absence of repeats of length $\ell \geq \log n$ in the codewords allows one to confidently merge reads with overlaps greater than $\log n$. However, these codes provide no guarantees for the merging of substrings with overlaps smaller than ℓ. Notice that the capacity-achieving code discussed in Section IV needs to carefully deal with the merging of small overlaps, which is needed in order to achieve capacity. To use repeat-free codes on the SSC, one would need to guarantee that successive reads overlap by at least $\log n$. To guarantee this with high probability, it can be shown that we need $K = \Theta(n)$. On the other hand we showed a coding scheme that provably requires only $\Omega(n/\log n)$ reads to achieve the channel capacity.

Acknowledgements

The work of Aditya Narayan Ravi and Ilan Shomorony was supported in part by the NSF Grant CCF-2007597 and in part by the NSF CAREER Award under Grant CCF-2046991. The work of A. Vahid was in part supported by NSF grants ECCS-2030285 and CNS-2106692.

Authorized licensed use limited to: INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY. Downloaded on May 15,2023 at 07:36:42 UTC from IEEE Xplore. Restrictions apply.

REFERENCES

[1] A. Motahari, G. Bresler, and D. Tse, "Information Theory of DNA Shotgun Sequencing," *IEEE Transactions on Information Theory*, vol. 59, pp. 6273–6289, Oct. 2013.

[2] E. S. Lander and M. S. Waterman, "Genomic mapping by fingerprinting random clones: a mathematical analysis," *Genomics*, vol. 2, no. 3, pp. 231–239, 1988.

[3] G. Bresler, M. Bresler, and D. Tse, "Optimal Assembly for High Throughput Shotgun Sequencing," *BMC Bioinformatics*, 2013.

[4] I. Shomorony, T. A. Courtade, and D. Tse, "Fundamental limits of genome assembly under an adversarial erasure model," *IEEE Transactions on Molecular, Biological and Multi-Scale Communications*, vol. 2, no. 2, pp. 199–208, 2016.

[5] G. M. Church, Y. Gao, and S. Kosuri, "Next-generation digital information storage in DNA," *Science*, vol. 337, no. 6102, pp. 1628–1628, 2012.

[6] N. Goldman, P. Bertone, S. Chen, C. Dessimoz, E. M. LeProust, B. Sipos, and E. Birney, "Towards practical, high-capacity, low-maintenance information storage in synthesized DNA," *Nature*, vol. 494, no. 7435, pp. 77–80, 2013.

[7] 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 International Edition*, vol. 54, no. 8, pp. 2552–2555, 2015.

[8] H. T. Yazdi, Y. Yuan, J. Ma, H. Zhao, and O. Milenkovic, "A rewritable, random-access DNA-based storage system," *Sci. Rep.*, vol. 5, p. 14138, 2015.

[9] Y. Erlich and D. Zielinski, "Dna fountain enables a robust and efficient storage architecture," *Science*, 2017.

[10] L. Organick, S. D. Ang, Y.-J. Chen, R. Lopez, S. Yekhanin, K. Makarychev, M. Z. Racz, G. Kamath, P. Gopalan, B. Nguyen, and et al., "Random access in large-scale DNA data storage," *Nature Biotechnology*, 2018.

[11] P. L. Antkowiak, J. Lietard, M. Z. Darestani, M. Somoza, W. J. Stark, R. Heckel, and R. N. Grass, "Low cost DNA data storage using photolithographic synthesis and advanced information reconstruction and error correction," *Nature Communications*, 2020.

[12] R. Heckel, I. Shomorony, K. Ramchandran, and D. N. C. Tse, "Fundamental limits of dna storage systems," in *IEEE International Symposium on Information Theory (ISIT)*, pp. 3130–3134, 2017.

[13] A. N. Ravi, A. Vahid, and I. Shomorony, "Coded shotgun sequencing," *https://arxiv.org/abs/2110.02868*, 2021.

[14] I. Shomorony and R. Heckel, "Capacity results for the noisy shuffling channel," in *IEEE International Symposium on Information Theory (ISIT)*, 2019.

[15] I. Shomorony and R. Heckel, "Dna-based storage: Models and fundamental limits," *IEEE Transactions on Information Theory*, vol. 67, no. 6, pp. 3675–3689, 2021.

[16] A. Lenz, P. H. Siegel, A. Wachter-Zeh, and E. Yaakobi, "Anchor-based correction of substitutions in indexed sets," *arXiv preprint arXiv:1901.06840*, 2019.

[17] A. Lenz, P. H. Siegel, A. Wachter-Zeh, and E. Yaakobi, "Coding over sets for dna storage," in *2018 IEEE International Symposium on Information Theory (ISIT)*, pp. 2411–2415, IEEE, 2018.

[18] A. Lenz, P. Siegel, A. Wachter-Zeh, and E. Yaakobi, "An upper bound on the capacity of the DNA storage channel," in *IEEE Information Theory Workshop*, 2019.

[19] A. Lenz, P. H. Siegel, A. Wachter-Zeh, and E. Yaakohi, "Achieving the capacity of the dna storage channel," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 8846–8850, IEEE, 2020.

[20] H. M. Kiah, G. J. Puleo, and O. Milenkovic, "Codes for DNA sequence profiles," *IEEE Trans. on Information Theory*, vol. 62, no. 6, pp. 3125–3146, 2016.

[21] R. Gabrys, H. M. Kiah, and O. Milenkovic, "Asymmetric Lee distance codes: New bounds and constructions," in *2015 IEEE Information Theory Workshop (ITW)*, pp. 1–5, 2015.

[22] F. Sala, R. Gabrys, C. Schoeny, and L. Dolecek, "Exact reconstruction from insertions in synchronization codes," *IEEE Transactions on Information Theory*, 2017.

[23] I. Shomorony, S. H. Kim, T. A. Courtade, and D. N. Tse, "Information-optimal genome assembly via sparse read-overlap graphs," *Bioinformatics*, vol. 32, no. 17, pp. i494–i502, 2016.

[24] T. Holenstein, M. Mitzenmacher, R. Panigrahy, and U. Wieder, "Trace reconstruction with constant deletion probability and related results.," in *SODA*, vol. 8, pp. 389–398, 2008.

[25] S. R. Srinivasavaradhan, M. Du, S. Diggavi, and C. Fragouli, "On maximum likelihood reconstruction over multiple deletion channels," in *2018 IEEE International Symposium on Information Theory (ISIT)*, pp. 436–440, IEEE, 2018.

[26] M. Cheraghchi, J. Ribeiro, R. Gabrys, and O. Milenkovic, "Coded trace reconstruction," in *2019 IEEE Information Theory Workshop (ITW)*, pp. 1–5, IEEE, 2019.

[27] R. Gabrys and O. Milenkovic, "Unique reconstruction of coded sequences from multiset substring spectra," in *2018 IEEE International Symposium on Information Theory (ISIT)*, pp. 2540–2544, IEEE, 2018.

[28] S. Marcovich and E. Yaakobi, "Reconstruction of strings from their substrings spectrum," *arXiv preprint arXiv:1912.11108*, 2019.

[29] I. Shomorony and A. Vahid, "Communicating over the torn-paper channel," in *GLOBECOM 2020 - 2020 IEEE Global Communications Conference*, pp. 1–6, 2020.

[30] A. N. Ravi, A. Vahid, and I. Shomorony, "Capacity of the torn paper channel with lost pieces," in *2021 IEEE International Symposium on Information Theory (ISIT)*, pp. 1937–1942, 2021.

[31] I. Shomorony and A. Vahid, "Torn-paper coding," *IEEE Transactions on Information Theory*, vol. 67, no. 12, pp. 7904–7913, 2021.

[32] S. Nassirpour and A. Vahid, "Embedded codes for reassembling non-overlapping random DNA fragments," *IEEE Transactions on Molecular, Biological and Multi-Scale Communications*, vol. 7, no. 1, pp. 40–50, 2020.

[33] S. Nassirpour, I. Shomorony, and A. Vahid, "Reassembly codes for the chop-and-shuffle channel," *arXiv preprint arXiv:2201.03590*, 2022.

[34] A. E. Gamal and Y.-H. Kim, *Network Information Theory*. Cambridge University Press, 2012.

[35] Y. Yehezkeally and N. Polyanskii, "On codes for the noisy substring channel," in *2021 IEEE International Symposium on Information Theory (ISIT)*, pp. 1700–1705, 2021.

[36] O. Elishco, R. Gabrys, E. Yaakobi, and M. Médard, "Repeat-free codes," *IEEE Transactions on Information Theory*, vol. 67, no. 9, pp. 5749–5764, 2021.

Authorized licensed use limited to: INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY. Downloaded on May 15,2023 at 07:36:42 UTC from IEEE Xplore. Restrictions apply.