On Coding for an Abstracted Nanopore Channel for DNA Storage

Reyna Hulett
Stanford University
450 Serra Mall, Stanford, CA 94305
Email: rmhulett@stanford.edu

Shubham Chandak
Stanford University
450 Serra Mall, Stanford, CA 94305
Email: schandak@stanford.edu

Mary Wootters
Stanford University
450 Serra Mall, Stanford, CA 94305
Email: marykw@stanford.edu

***Abstract*—In the emerging field of DNA storage, data is encoded as DNA sequences and stored. The data is read out again by sequencing the stored DNA. Nanopore sequencing is a new sequencing technology that has many advantages over other methods; in particular, it is cheap, portable, and can support longer reads. While several practical coding schemes have been developed for DNA storage with nanopore sequencing, the theory is not well understood. Towards that end, we study a highly abstracted (deterministic) version of the nanopore sequencer, which highlights key features that make its analysis difficult. We develop methods and theory to understand the capacity of our abstracted model, and we propose efficient coding schemes and algorithms.**

A full version of this paper is accessible at: https://arxiv.org/pdf/2102.01839.pdf

I. Introduction

In the emerging field of *DNA storage,* data is encoded as DNA sequences and stored; the data can be read back by sequencing the stored DNA. This technology promises high storage density and stability, as well as efficient duplication of data and random access using PCR-based technologies; we refer the reader to [1] and the references therein for an excellent overview.

Both the synthesis and sequencing processes are noisy, and as a result the data must be encoded before the synthesis stage to ensure accurate data recovery. Prior work has studied methods for encoding data in order to protect it against (aspects of) the noise introduced by these processes, for example [2]–[9].

In this work, we focus on one particular stage of this noisy process, the *nanopore sequencer.* Nanopore sequencing—and in particular the MinION sequencer developed by Oxford Nanopore Technologies [10]—is an emerging sequencing technology. While initial works in DNA storage used Illumina sequencing, nanopore sequencing has been attracting interest due to its portability, low cost, and ability to support significantly longer reads than Illumina.

At a high level, the nanopore sequencer works as follows. A single strand of DNA is passed through a pore, leading to variations in a current readout. The pore can hold k nucleotides (for our purposes, a nucleotide is just a value in $\{A, C, G, T\}$) at a time; in practice k is about six. The value of the current readout depends on which nucleotides are in the pore. For example, if the strand of DNA is ATGCCAGT, and the pore sees the sub-strand ATGC, it will output one current reading.

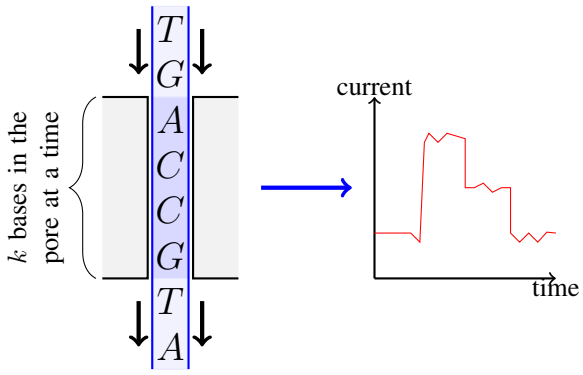

Fig. 1. High-level view of the nanopore sequencer.

As the strand is passed through the pore, the contents of the pore will shift, say from ATGC to TGCC, then to GCCA, and so on. This will result in a change in the current reading, according to some function f that maps k-mers to current levels. The process is depicted in Figure 1. Given the current readout, the goal is to recover the original DNA sequence.

This channel is difficult to analyze, for several reasons. First, the output at any given time depends on $k > 1$ bases, and so there is inter-symbol interference. Second, there may be collisions in the output: two different pore contents may lead to similar current readouts. Third, the current readout can be noisy. Fourth, the amount of time that each k-mer spends in the pore can vary, and sometimes never occur at all, leading to synchronization errors in the output.

Due to this complexity, typical basecallers (that is, methods for recovering the original sequence from the current readouts) rely on machine learning techniques [11]–[13]. This is effective in practice, but difficult to get a theoretical handle on. While there are several practical approaches to error correction for nanopore sequencing in DNA storage [5]–[9], the theoretical limits of this channel are not well understood. The work [14] proposed a probabilistic model of the nanopore sequencer and developed bounds on the capacity of the resulting channel. As discussed more below, their model is stochastic and captures the binary deletion channel.

In this work, we take a different approach to developing a theoretical understanding of the nanopore sequencer. We

2021 IEEE International Symposium on Information Theory (ISIT) | 978-1-5386-8209-8/21/$31.00 ©2021 IEEE | DOI: 10.1109/ISIT45174.2021.9518236

develop a highly abstracted, *deterministic* model, which highlights the first two sources of noise mentioned above: the inter-symbol interference, and the possible collisions when multiple k-mers lead to similar current outputs. We develop methods and theory to understand the capacity of our abstracted channel, and we propose efficient coding schemes and algorithms.

Our goal in this work is to open up a research direction towards a theoretical understanding of the nanopore sequencer. We fully acknowledge that our work is not yet practical; in particular our abstracted channel does not include noise in the current readings or synchronization errors that can arise from variable pore dwelling times. It is our hope that a solid understanding of our abstracted channel can then be combined with (much more well-studied) theories of coding for substitution and synchronization errors, in order to make progress on a more realistic channel model.

a) Contributions: First, we propose a novel abstraction of the nanopore sequencer, which highlights the inter-symbol interference and the prospect of collisions. This abstraction is simple enough that it is tractable, yet complex enough that it (i) captures some fundamental properties of the nanopore sequencer, and (ii) already gives rise to extremely interesting problems from a theoretical perspective. We hope that our abstraction will lead to future work in this area.

Second, in Section III, we develop an algorithm to determine the capacity of this channel. The algorithm is inefficient as pore size k grows, but we can use it for small k.

Third, in Section IV, we develop simple bounds on the "best" and "worst" capacity for an arbitrary pore size k (where "best" and "worst" refer to the choice of the map from k-mers to current readouts). We use our aforementioned algorithm to compare these bounds to the exact values for $k = 2$.

Fourth, in Section V, we develop efficient coding schemes for our abstracted channel. In particular, we develop a scheme that achieves rate $C_f - \epsilon$, where C_f is the capacity, that requires preprocessing time $O\left(\frac{1}{\epsilon} \cdot 4^{1/\epsilon}\right)$, and has encoding/decoding time $O(n)$, where the $O(\cdot)$ hides a constant that depends on f. We also present an efficient coding scheme that improves over the "naive" one with high probability, where the probability is over a random map from k-mers to current readouts.

b) Related Work: DNA storage has been around as an idea since the 1960's, and there has been renewed interest in the past decade, starting with the works [2], [15]. We refer the reader to [1] for an excellent survey. Most works on DNA storage have focused on other sequencing technologies like Illumina, but recently there have been several which developed practical DNA storage systems for nanopore sequencers, including [5]–[9]. All of these works developed practical coding schemes for DNA storage with a nanopore sequencer, but did not explicitly model the sequencer or analyze the theoretical limits. The capacity of DNA storage systems *has* been studied theoretically [16]–[20], but usually *without* regard to the unique features of the nanopore sequencer.

Notably, the work [21] does perform a theoretical analysis of the nanopore sequencer. However, they focus on a different type of error than the present work: the variable amount of time each k-mer spends in the pore. Specifically, they analyze the effect of errors in the lengths of runs of identical bases (without explicitly modeling the current readout), and analyze the sample complexity rather than the capacity.

Perhaps the work most related to ours is that of [14], who also developed a model of the nanopore sequencer and studied the capacity of their model. In particular, they give a multi-letter capacity formula for their channel, and derive computable bounds for the capacity, in terms of a Markov transition matrix P that captures the probability of transitioning from one k-mer to the next in the pore. Our work complements that work by focusing on different aspects of the problem. First, the model in [14] is stochastic, while ours is deterministic. As a result, they take an information-theoretic approach, while our approach is more combinatorial. Second, their model includes the possibility that k-mers might get dropped; in particular, the binary deletion channel appears as one part of their model. Since understanding the capacity of the binary deletion channel is a difficult open problem, this makes their problem extremely difficult. In contrast, we ignore this aspect to more cleanly focus on the effects of the inter-symbol interaction and collisions between k-mers. Third, that work focuses on the nanopore sequencer for general applications (not necessarily for DNA storage), and in particular does not consider efficient coding schemes. Finally, that work derives bounds on the channel capacity for a particular choice of (a stochastic analog of) the map f from k-mers to current levels, derived from experimental data. In contrast, we are interested in results for any f, and in particular for the best and worst such functions f. While the former direction is obviously of immediate interest for existing technology, it is our hope that understanding how the capacity of the channel changes with f could perhaps guide how nanopore technology is developed in the future. We note that this is still an emerging area and the technology is evolving; see [22] for an overview.

Finally, we note that our problem is related to coding for constrained systems [23]. In particular, for a fixed mapping f, the set of possible current readouts forms a constrained system (ignoring the question of which DNA strands give rise to which current readouts). Thus, Algorithm 1 for computing the capacity could instead have been written using the approach of [23]: using the spectral radius of an appropriate irreducible deterministic matrix, rather than the transfer matrix. Additionally, the coding scheme of Theorem 5 is similar to a *block encoder* as described in [23], with the added complication of choosing the appropriate DNA sequences to create the current readout blocks.

II. Abstract Model of Nanopore Sequencer

In this section we formalize our model. As mentioned above, our goal is to focus on (i) inter-symbol interference, and (ii) the possibility of different k-mers producing similar current readouts. With that in mind, we propose a very simple model for the nanopore sequencer. As input, we take an encoded string $s_0 s_1 \cdots s_{n-1} \in \{A, C, G, T\}^n$. This is transformed into a sequence of k-mers according to a sliding window, to obtain

$(s_0 \cdots s_{k-1}), (s_1 \cdots s_k), \ldots, (s_{n-k} \cdots s_{n-1})$. Finally, each of these k-mers is mapped to one of b distinct current levels, according to a mapping $f : \{A, C, G, T\}^k \to \{0, 1, \ldots, b-1\}$. This mapping f defines the channel.

Definition 1 (Abstract Nanopore Channel): Given a mapping $f : \{A, C, G, T\}^k \to \{0, 1, \ldots, b-1\}$ from k-mers to current levels, let $f^* : \{A, C, G, T\}^* \to \{0, 1, \ldots, b-1\}^*$ represent the mapping from DNA strands to their current readout:

$$f^*(s_0 \cdots s_{n-1}) = f(s_0 \cdots s_{k-1}) \circ \cdots \circ f(s_{n-k} \cdots s_{n-1}).$$

We call f^* the *abstract nanopore channel* given by f.

Given a mapping f, we are interested in the capacity C_f (in bits-per-base), which we define as follows.

Definition 2: Let $f : \{A, C, G, T\}^k \to \{0, 1, \ldots, b-1\}$. The capacity C_f of channel f^* is defined as

$$C_f = \lim_{n \to \infty} \frac{\log |\{c \mid \exists s \in \{A, C, G, T\}^n \text{ s.t. } f^*(s) = c\}|}{n}.$$

Observe that if $\mathcal{S} \subseteq \{A, C, G, T\}^n$ is a collection of strings so that $f^*(s)$ are all distinct for $s \in \mathcal{S}$, then by assigning a different message to each $s \in \mathcal{S}$, we can communicate perfectly (if not necessarily efficiently) across the channel.

III. Computing the Capacity

Our first contribution is an algorithm (Algorithm 1 below) that computes C_f, given f. The basic idea is to consider a finite automaton on the alphabet $\{0, 1, \ldots, b-1\}$ that accepts exactly those current readouts that can be generated by some DNA strand; then we use the transfer matrix method [24], [25] for counting accepting paths in that finite automaton.

Formally, an Nondeterministic Finite Automaton (NFA) is a tuple $(Q, \Sigma, \Delta, Q_0, F)$ where Q is the set of states, Σ is the alphabet, $\Delta : Q \times \Sigma \to 2^Q$ is the transition function, $Q_0 \subseteq Q$ is the set of initial states, and $F \subseteq Q$ is the set of accepting states. Likewise, a Deterministic Finite Automaton (DFA) is a tuple $(Q, \Sigma, \delta, q_0, F)$, defined analogously except that $\delta : Q \times \Sigma \to Q$ is the transition function and there is only a single initial state $q_0 \in Q$.

We consider the NFA M and the DFA M' described in Algorithm 1. The NFA M has states indexed by strings in $\{A, C, G, T\}^{k-1}$ and alphabet $\Sigma = \{0, 1, \ldots, b-1\}$, the current levels. Given a state $(s_0 \cdots s_{k-2})$ and an input current level $i \in \Sigma$, the NFA M can transition to any other state of the form $(s_1 \cdots s_{k-1})$ so that $f(s_0 s_1 \cdots s_{k-1}) = i$. All states are accepting states. By construction, a current readout $c \in \Sigma^{n-k+1}$ can be an output $f^*(s_0 s_1 \cdots s_{n-1})$ of the channel if and only if c is accepted by M. The DFA M' accepts exactly the same strings as M, and is obtained using the classic subset construction [26], such that each state of the DFA corresponds to a subset of the states of the NFA.

The *transfer matrix method* is a method for obtaining a *generating function* $g_f(z)$ for the number of strings accepted by a given DFA. In more detail, given the *transfer matrix* T for the automaton (so that $T_{i,j}$ is the number of transitions from state i to state j), the transfer matrix method gives an expression for a function $g_f(z)$ (in terms of T), so that $g_f(z) = \sum_{m=0}^{\infty} N_m z^m$, so that N_m is the number of strings of length m accepted by the finite automaton. We will use the notation $[z^m] g_f(z)$ to denote the coefficient N_m on z^m.

Lemma 1 (Transfer Matrix Method [24]): Given a DFA $D = (Q, \Sigma, \delta, q_0, F)$, let $D' = (Q \cup q_F, \Sigma \cup \lambda, \delta', q_0, \{q_F\})$ be obtained from the DFA D by adding a new state q_F and new symbol λ, along with λ-transitions from each of the accepting states of D to q_F. Let T be the transfer matrix of D', where $T_{i,j}$ is the number of transitions from state i to state j, with q_0 being state 0 and q_F being state $|Q|$. Then the generating function for D' is

$$g_f(z) = (-1)^{|Q|} \times \frac{\det(I - zT : |Q|, 0)}{z \det(I - zT)}$$

where $(I - zT : |Q|, 0)$ is the minor of index $|Q|, 0$, i.e., the matrix $I - zT$ with the $|Q|^{th}$ row and 0^{th} column deleted.

In our case, the number of accepted strings $[z^m] g_f(z)$ is the number of current readouts of length $m = n - k + 1$ that can be generated by some DNA strand of length n.

From $g_f(z)$, we can derive the asymptotic behavior of the number of possible current readouts required to determine C_f. As shown in the proof below, it is related to the smallest positive singularity of $g_f(z)$.

Algorithm 1 Calculate C_f

input: window size k, # of current levels b, mapping f

1: $Q \leftarrow \{A, C, G, T\}^{k-1}$
2: $\Sigma \leftarrow \{0, 1, \ldots, b-1\}$
3: NFA $M \leftarrow (Q, \Sigma, \Delta, Q, Q)$ where
$\Delta(s_0 \cdots s_{k-2}, i) = \{s_1 \cdots s_{k-1} \mid f(s_0 \cdots s_{k-1}) = i\}$
4: DFA $M' \leftarrow (2^Q, \Sigma, \delta, q_0, F)$ via subset construction
5: $M' \leftarrow (2^Q \cup \{q_F\}, \Sigma \cup \{\lambda\}, \delta', q_0, \{q_F\})$ where

$$\delta'(q, \sigma) = \begin{cases} \delta(q, \sigma) & q \in 2^Q \text{ and } \sigma \neq \lambda \\ q_F & q \in F \text{ and } \sigma = \lambda \\ \emptyset & \text{otherwise} \end{cases}$$

6: $T \leftarrow$ transfer matrix of M', i.e.,
$T_{i,j} =$ number of transitions from state i to state j
7: $g_f(z) \leftarrow (-1)^{2^{|Q|}} \times \frac{\det(I - zT : 2^{|Q|}, 0)}{z \det(I - zT)}$
8: $r \leftarrow$ smallest positive root of the denominator of $g_f(z)$
9: **return** $\log \frac{1}{r}$

Theorem 2: Given a mapping $f : \{A, C, G, T\}^k \to \{0, 1, \ldots, b-1\}$, Algorithm 1 computes C_f.

Proof: First, observe that the NFA M accepts exactly those current readouts that can be generated by some DNA strand under the mapping f. For any current readout accepted by M, consider an arbitrary accepting path $P = s_0 \cdots s_{k-2}, s_1 \cdots s_{k-1}, \ldots, s_m \cdots s_{m+k-2}$. By construction, the DNA strand $s_0 \cdots s_{m+k-2}$ generates that current readout. In the other direction, for any DNA strand $s_0 \cdots s_{m+k-2}$, the path P is an accepting path for $f^*(s_0 \cdots s_{m+k-2})$.

The DFA obtained from the subset construction accepts the same current readouts as M [26]. Then we apply Lemma 1 to obtain the generating function $g_f(z)$, which counts the number of current readouts accepted by M.

Finally, we need to extract the asymptotic behavior of $[z^m]g_f(z)$. Since $g_f(z)$ is a generating function with non-negative coefficients, the Exponential Growth Formula [25] tells us that $\limsup_{m\to\infty}\left([z^m]g_f(z)\right)^{1/m} = 1/r$ where r is the smallest positive singularity of $g_f(z)$. Note that the number of possible current readouts is monotonically non-decreasing in m, so the lim sup is equal to the limit. Therefore, based on Definition 2, and the fact that the length of the current readouts for DNA strands of length n is $m = n-k+1$,

$$\begin{aligned} C_f &= \lim_{n\to\infty} \frac{\log\left([z^{n-k+1}]g_f(z)\right)}{n} \\ &= \lim_{n\to\infty} \frac{n-k+1}{n} \log\left([z^{n-k+1}]g_f(z)\right)^{1/(n-k+1)} \\ &= \log\frac{1}{r} \end{aligned}$$

■

Unfortunately, the DFA obtained from the subset construction has $2^{4^{k-1}}$ states, so the runtime of Algorithm 1 is exponential in the problem size. It is possible that calculating C_f may be hard, because computing such a statistic for NFAs *in general* is PSPACE-complete [27]. Specifically, even determining whether $C_f = \log b$ for $b = 2$ or 4 (in which case $C_f = \log b$ is the best possible capacity over all mapping functions f; see Lemma 4) is equivalent to determining whether the corresponding NFA is universal (i.e., accepts every string in the alphabet). We make this precise in the following lemma, the proof of which can be found in the full version.

Lemma 3: For $b = 2$ or 4 and any f, C_f is equal to $\log b$ if and only if every current readout in $\{0, 1, \ldots, b-1\}^*$ can be generated by some DNA strand.

Since the universality problem for NFAs is PSPACE-complete, an efficient algorithm would have to in some way leverage the highly structured nature of the NFAs corresponding to abstract nanopore channels.

IV. Bounding the Capacity

The above approach for computing C_f exactly given a mapping f is only practical for small window sizes k. However, we can derive some general bounds that apply to any mapping f. In particular, we are interested in the *worst-case* capacity (i.e., $\min_f C_f$), as well as the *best-case* capacity (i.e., $\max_f C_f$).

However, note that if the mapping f is unrestricted, then $\min_f C_f = 0$ — consider $f(\cdot) = 0$. Therefore, in this section, we will focus on *balanced* mappings f. These are mappings so that $|f^{-1}(i)|$ is the same for all $i \in \{0, 1, \ldots, b-1\}$.

Lemma 4: For a given window size k and with b distinct current levels, we have the following bounds on C_f:

1) $\max_f C_f = \min(\log(b), 2)$
2) $\min_f C_f \geq \frac{\log(b)}{k}$
3) $\min_f C_f \leq 1$ when $b \leq 2^k$

Proof: We defer the proofs of 1) and 3) to the full version. For 2), regardless of the mapping f, we can always generate at least $b^{\lfloor n/k \rfloor}$ distinct current readouts: any choice of desired 0^{th}, k^{th}, $2k^{th}$, etc. current readings can be obtained because they correspond to non-overlapping length-k windows. Thus $\min_f C_f \geq \lim_{n\to\infty} \frac{\log b^{\lfloor n/k \rfloor}}{n} = \frac{\log(b)}{k}$. ■

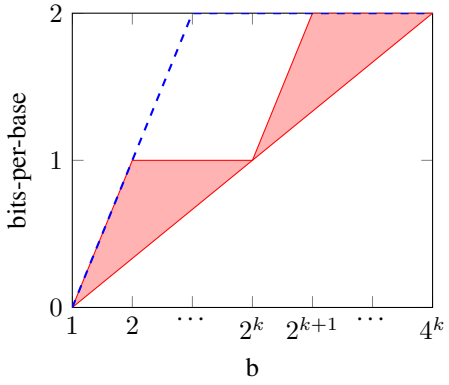

Fig. 2. Bounds on the value of C_f. The dashed blue line is equal to $\max_f C_f$. The value of $\min_f C_f$ must lie somewhere in the red shaded area.

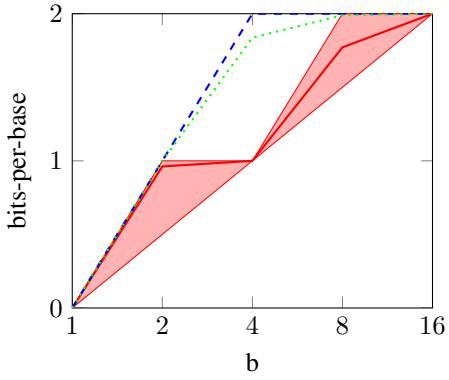

Fig. 3. For $k = 2$, the true $\max_f C_f$ (dashed blue), $\min_f C_f$ (solid red), and $\mathbb{E}_f C_f$ (dotted green), superimposed over our bounds from Figure 2.

We plot these bounds in Figure 2. One might wonder about the exact $\min_f C_f$, and also where C_f lies for a "typical" mapping f. Using Algorithm 1, we can exactly calculate C_f for every balanced mapping f with $k = 2$; the empirical maximum, minimum, and average C_f are shown in Figure 3.

The bounds on $\min_f C_f$ are curious. Our theoretical bounds on $\min_f C_f$ are not tight, but the $k = 2$ results suggest there may be some "bumpiness" in the true bound. Also based on the $k = 2$ results, it appears that random (balanced) mappings are closer to the best-case scenario than the worst-case. Section V-B illustrates one coding scheme that takes advantage of a property shared by most random mappings for $b = 2$, and we may hope there exist more general coding schemes that perform well for random mappings.

V. Coding Schemes

In the proof of Lemma 4, we observed that given a mapping f with window size k and b distinct current levels, we can achieve a rate of $\frac{\log(b)}{k}$ by coding only on the $0^{th}, k^{th}, 2k^{th}$, etc. current readings. Here, we propose two generalizations of this trivial scheme to improve the rate with additional preprocessing based on the mapping f.

A. "Block" Encoding

Instead of coding on non-overlapping windows of length k, each of which maps to one of b current readings, we may instead choose a block length $\ell \geq k$ and code on non-overlapping blocks of length ℓ. This requires precomputing an alphabet $\Sigma(\ell) \subseteq \{A, C, G, T\}^\ell$ such that each DNA strand in $\Sigma(\ell)$ generates a unique current readout, and together they generate every possible current readout of length $\ell - k + 1$ permitted by the mapping f. Provided our desired strand length n is divisible by ℓ, this coding scheme obtains the rate

$$C_f(\ell) = \frac{\log |\{c \mid \exists s \in \{A, C, G, T\}^\ell \text{ s.t. } f^*(s) = c\}|}{\ell},$$

which equals $\frac{\log |\Sigma(\ell)|}{\ell}$. Since $\lim_{\ell \to \infty} C_f(\ell) = C_f$, we can get arbitrarily close to capacity by picking ℓ sufficiently large.

Theorem 5: Given a mapping f, for all $\epsilon > 0$, there is a coding scheme achieving rate $C_f - \epsilon$ with linear time encoding and decoding that requires preprocessing time $O(\frac{1}{\epsilon} \cdot 4^{1/\epsilon})$, where the $O(\cdot)$ hides constants that may depend on f.

Proof: We will exhibit such a scheme by choosing an appropriate block length ℓ. Let $|\Sigma(\ell)|$ be the number of distinct current readouts that can be generated from DNA strands of length ℓ. Equivalently, $|\Sigma(\ell)| = [z^{\ell-k+1}] g_f(z)$ is the number of current readouts of length $\ell - k + 1$ accepted by the NFA M constructed by Algorithm 1. Because $g_f(z)$ is a counting function for a regular language, and because $[z^{\ell-k+1}] g_f(z)$ is non-decreasing in ℓ, the asymptotic behavior of $[z^{\ell-k+1}] g_f(z)$ has a simple form ([25], Theorem V.3):

$$|\Sigma(\ell)| = [z^{\ell-k+1}] g_f(z) = \Theta(\Pi(\ell - k + 1)(2^{C_f})^{\ell-k+1})$$

where $\Pi(x)$ is a polynomial.

Therefore, there must exist some constants ℓ_0 and C, depending only on the mapping f, such that for all $\ell \geq \ell_0$, we have $|\Sigma(\ell)| \geq C(2^{C_f})^{\ell-k+1}$. Thus, provided that we choose $\ell \geq \max\left(\ell_0, \frac{C_f \cdot (k-1) - \log C}{\epsilon}\right)$, we see that

$$\begin{aligned} C_f(\ell) = \frac{\log |\Sigma(\ell)|}{\ell} &\geq \frac{\log\left(C(2^{C_f})^{\ell-k+1}\right)}{\ell} \\ &= C_f - \frac{C_f \cdot (k-1) - \log C}{\ell} \geq C_f - \epsilon. \end{aligned}$$

Therefore, to achieve rate $C_f - \epsilon$, we should choose ℓ proportional to $1/\epsilon$, with the constants depending only on f.

Given the block length ℓ, we now describe how to compute the alphabet $\Sigma(\ell)$. We will construct an array E containing the alphabet $\Sigma(\ell)$ and a hash table D mapping current readouts of length $\ell - k + 1$ to the index in E of the DNA strand that generates that readout.

For each DNA strand s of length ℓ, compute $f^*(s)$. If $f^*(s)$ has not yet been added to the hash table D, append s to the end of array E and map $f^*(s)$ to the appropriate index. This preprocessing takes $O(\ell)$ time for each of 4^ℓ DNA strands, for a total of $O(\ell \cdot 4^\ell) = O(\frac{1}{\epsilon} \cdot 4^{1/\epsilon})$.

Encoding and decoding are straightforward: Convert the message to base $|\Sigma(\ell)|$ and use array E to map each digit to a block of length ℓ. Similarly, decode each block of $\ell - k + 1$ current readings using the hash table D (skipping the $k - 1$ readings that straddle each pair of adjacent blocks). ∎

As an example, consider the case when $b = 2$ or 4 and f is any mapping with best-case capacity, $C_f = \log b$. Per Lemma 3, this implies that every current readout can be generated by some DNA strand, so $|\Sigma(\ell)| = b^{\ell-k+1}$ and $C_f(\ell) = \frac{(\ell-k+1)\log(b)}{\ell}$. In this case, we can calculate the dependence of ℓ on ϵ exactly. For $C_f(\ell) \geq C_f - \epsilon$, we need

$$\ell \geq \frac{(k-1)\log(b)}{\epsilon}.$$

For instance, when $k = 2, b = 2$, and $\epsilon = 0.1$, we would require $\ell = 10$.

B. "Greedy" Encoding

Instead of changing the lengths of the blocks used in the trivial scheme, we may relax the "non-overlapping" requirement. This may not always be possible, depending on the mapping f. In the worst case, it is possible that once you have fixed the first k bases, the next $k - 1$ current readings may also be fixed—for instance, if the first k bases are all A, and all windows starting with A map to the same current level. However, this bad case shouldn't happen for most mappings.

Consider the case of $b = 2$ current levels, and suppose that for some length $1 \leq \ell < k$, for every "prefix" $p \in \{A, C, G, T\}^\ell$, at least one window in $f^{-1}(0)$ *and* at least one window in $f^{-1}(1)$ starts with that prefix. Then it is possible to have every $(k - \ell)^{th}$ current reading code for one binary symbol independently. This would give us a rate of $\frac{1}{k-\ell}$ rather than $\frac{1}{k}$. Such an event is not too unlikely with a random mapping f. This is formalized in the following lemma, the proof of which may be found in the full version.

Lemma 6: Given a random mapping f with $b = 2$ distinct current levels and a length $1 \leq \ell < k$, f admits the coding scheme described above with rate $\frac{1}{k-\ell}$ with probability at least $1 - 4^\ell \cdot 2 \cdot \left(\frac{1}{2}\right)^{4^{k-\ell}}$. Furthermore,

1) we can determine whether such a scheme exists for a given f and ℓ in $O(4^k)$.
2) we can find the maximum ℓ for which such a scheme exists for a given f in $O(k4^k)$.
3) we can implement such a scheme with $O(k4^k)$ preprocessing and linear encoding and decoding.

For example, with $k = 6, \ell = 4$, we see that we can obtain a rate of $1/2$ (compared to the trivial $1/6$) with probability at least $1 - \frac{1}{128}$.

VI. Conclusion

We have initiated a theoretical study of coding for a highly abstracted version of the nanopore sequencer for DNA storage. We have provided algorithms and bounds for understanding the capacity, and we have given efficient coding schemes. However, we view our work as the tip of an iceberg. First, even for this abstracted model, much remains open. Can one derive better bounds on the capacity, or compute it efficiently for, say, $k = 6$? Second, we hope that our insights will generalize to more practical models, including with substitution and synchronization errors.

REFERENCES

[1] L. Ceze, J. Nivala, and K. Strauss, "Molecular digital data storage using DNA," *Nature Reviews Genetics*, vol. 20, no. 8, pp. 456–466, 2019.

[2] 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.

[3] M. Blawat, K. Gaedke, I. Huetter, X.-M. Chen, B. Turczyk, S. Inverso, B. W. Pruitt, and G. M. Church, "Forward error correction for DNA data storage," *Procedia Computer Science*, vol. 80, pp. 1011–1022, 2016.

[4] Y. Erlich and D. Zielinski, "DNA Fountain enables a robust and efficient storage architecture," *Science*, vol. 355, no. 6328, pp. 950–954, 2017.

[5] S. H. T. Yazdi, R. Gabrys, and O. Milenkovic, "Portable and error-free DNA-based data storage," *Scientific reports*, vol. 7, no. 1, pp. 1–6, 2017.

[6] L. Organick, S. D. Ang, Y.-J. Chen, R. Lopez, S. Yekhanin, K. Makarychev, M. Z. Racz, G. Kamath, P. Gopalan, B. Nguyen *et al.*, "Random access in large-scale DNA data storage," *Nature biotechnology*, vol. 36, no. 3, p. 242, 2018.

[7] H. H. Lee, R. Kalhor, N. Goela, J. Bolot, and G. M. Church, "Terminator-free template-independent enzymatic DNA synthesis for digital information storage," *Nature communications*, vol. 10, no. 1, pp. 1–12, 2019.

[8] R. Lopez, Y.-J. Chen, S. D. Ang, S. Yekhanin, K. Makarychev, M. Z. Racz, G. Seelig, K. Strauss, and L. Ceze, "DNA assembly for nanopore data storage readout," *Nature communications*, vol. 10, no. 1, pp. 1–9, 2019.

[9] S. Chandak, J. Neu, K. Tatwawadi, J. Mardia, B. Lau, M. Kubit, R. Hulett, P. Griffin, M. Wootters, T. Weissman *et al.*, "Overcoming high nanopore basecaller error rates for DNA storage via basecaller-decoder integration and convolutional codes," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 8822–8826.

[10] M. Jain, H. E. Olsen, B. Paten, and M. Akeson, "The Oxford Nanopore MinION: delivery of nanopore sequencing to the genomics community," *Genome biology*, vol. 17, no. 1, pp. 1–11, 2016.

[11] R. R. Wick, L. M. Judd, and K. E. Holt, "Performance of neural network basecalling tools for Oxford Nanopore sequencing," *Genome biology*, vol. 20, no. 1, pp. 1–10, 2019.

[12] "Flappie: Flip-flop basecaller for Oxford Nanopore reads," https://github.com/nanoporetech/flappie, last accessed: January 2021.

[13] "Scrappie: A technology demonstrator for the Oxford Nanopore Research Algorithms group," https://github.com/nanoporetech/scrappie, last accessed: January 2021.

[14] W. Mao, S. N. Diggavi, and S. Kannan, "Models and information-theoretic bounds for nanopore sequencing," *IEEE Transactions on Information Theory*, vol. 64, no. 4, pp. 3216–3236, 2018.

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

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

[17] M. Kovačević and V. Y. F. Tan, "Codes in the space of multisets—coding for permutation channels with impairments," *IEEE Transactions on Information Theory*, vol. 64, no. 7, pp. 5156–5169, 2018.

[18] J. Sima, N. Raviv, and J. Bruck, "On coding over sliced information," in *2019 IEEE International Symposium on Information Theory (ISIT)*, 2019, pp. 767–771.

[19] I. Shomorony and R. Heckel, "Capacity results for the noisy shuffling channel," in *2019 IEEE International Symposium on Information Theory (ISIT)*. Institute of Electrical and Electronics Engineers Inc., 7 2019.

[20] A. Lenz, P. H. Siegel, A. Wachter-Zeh, and E. Yaakobi, "Coding over sets for dna storage," *IEEE Transactions on Information Theory*, vol. 66, no. 4, pp. 2331–2351, 2020.

[21] A. Magner, J. Duda, W. Szpankowski, and A. Grama, "Fundamental bounds for sequence reconstruction from nanopore sequencers," *IEEE Transactions on Molecular, Biological and Multi-Scale Communications*, vol. 2, no. 1, pp. 92–106, 2016.

[22] Y. Goto, R. Akahori, I. Yanagi, and K.-i. Takeda, "Solid-state nanopores towards single-molecule DNA sequencing," *Journal of human genetics*, vol. 65, no. 1, pp. 69–77, 2020.

[23] B. Marcus, R. Roth, and P. Siegel, *An Introduction to Coding for Constrained Systems*, 2001.

[24] A. Aydin, L. Bang, and T. Bultan, "Automata-based model counting for string constraints," in *Computer Aided Verification*, D. Kroening and C. S. Păsăreanu, Eds. Cham: Springer International Publishing, 2015, pp. 255–272.

[25] P. Flajolet and R. Sedgewick, *Analytic Combinatorics*. Cambridge University Press, 1 2009.

[26] M. O. Rabin and D. Scott, "Finite automata and their decision problems," *IBM Journal of Research and Development*, vol. 3, no. 2, pp. 114–125, 1959.

[27] N. Rampersad, J. Shallit, and Z. Xu, "The computational complexity of universality problems for prefixes, suffixes, factors, and subwords of regular languages," *CoRR*, vol. abs/0907.0159, 2009. [Online]. Available: http://arxiv.org/abs/0907.0159