Optimal Reference for DNA Synthesis

Ohad Elishco, Member, IEEE, and Wasim Huleihel, Member, IEEE

Abstract—In recent years, DNA has emerged as a potentially viable storage technology. DNA synthesis, which refers to the task of writing the data into DNA, is perhaps the most costly part of existing storage systems. Consequently, the high cost and low throughput limit the practical use of available DNA synthesis technologies. It has been found that the homopolymer run (i.e., the repetition of the same nucleotide) is a major factor affecting the synthesis and sequencing errors. Recently, Lenz et al. (2020) raised and studied the coding problem for efficient synthesis for DNA-based storage systems. Among other things, they studied the maximal code size under synthesis constraints. In Makarychev et al. (2020), the authors studied the role of batch optimization in reducing the cost of large-scale DNA synthesis, for a given pool \( S \) of random quaternary strings of fixed length. This problem is related to the problem posed in Lenz et al. (2020) which can be viewed as the opposite side of the coin. Instead of seeking the largest code in which every codeword can be synthesized in a certain amount of time, they asked what is the average synthesis time of a randomly chosen string. Following the lead of Makarychev et al. (2020), in this paper, we take a step forward towards the theoretical understanding of DNA synthesis, and study the homopolymer run of length \( k \geq 1 \). Specifically, we are given a set of DNA strands \( S \), randomly drawn from a Markovian distribution modeling a general homopolymer run length constraint, that we wish to synthesize. For this problem, we derive asymptotically tight high probability lower and upper bounds on the cost of DNA synthesis, for any \( k \geq 1 \). Our bounds imply that, perhaps surprisingly, the periodic sequence \( ACGT \) is asymptotically optimal in the sense of achieving the smallest possible cost. Our main technical contribution is the representation of the DNA synthesis process as a certain constrained system, for which string techniques can be applied.

Index Terms—Information entropy, DNA, encoding, decoding, codes, biological information theory.

I. INTRODUCTION

DNA data storage refers to the process of encoding (decoding) data to (from) synthesized strands (or, strands) of DNA. Recently, there has been a growing interest in the problem of storing data in synthetic DNA molecules. Indeed, DNA, as a storage medium, has an enormous potential because of its high storage density compared to other conventional storage media. Unfortunately, however, the practical use of DNA as a provable efficient storage technology is currently greatly restricted. This is mainly because of its high cost and long read and write duration.

The typical approach used for producing DNA molecules is array-based DNA synthesis (see, e.g., [22]). In a nutshell, in this technique, a machine synthesizes a large number of DNA strands in parallel (referred to as information strands), where each such strand is grown by one DNA character at each time step of the process. To that end, the machine generates multiple copies of a specific nucleotide, and these nucleotides are concatenated to a selected subset of the information strands. The nucleotide that the machine generates at any given time is determined according to a fixed reference sequence. Specifically, as the synthesizer passes through this reference strand, the next character it reads in the reference strand is generated and concatenated to the selected subset of information strands. This process terminates when the machine reaches the end of the reference strand. It is evident that in order for the synthesis process to work, the reference strand must be a supersequence of all the information strands. This way each synthesized DNA strand is a subsequence of the reference sequence and is synthesized. Thus, the length of the reference strand determines the synthesis time of this DNA synthesis process. In this paper, we will refer to the synthesis time as the cost.

The encoding process in DNA data storage generates a list of DNA strands that need to be synthesized, by translating binary sources into strands of nucleotides (for example, by mapping two binary source bits into a single nucleotide). It is well-known that repetitions of the same nucleotide, also known as, a homopolymer run, may significantly increase the chance of sequencing errors [3], [41]. For example, it was observed in [41] that a long homopolymer run (e.g. more than 4 nucleotides) results in a significant increase of insertion and deletion errors, and therefore long runs should be avoided. Therefore, encoding algorithms often ensure that the resulting strands do not contain long runs of the same character [9], [36].

A. Main Conceptual and Technical Contributions

In this paper, we consider the following meta-generative model: we are given a set of DNA strands \( S_k \), drawn at random from a “natural” distribution. This natural distribution aims to capture a general homopolymer run length constraint of length \( k \geq 1 \), i.e., the strands to synthesized are not allowed to contain \( k + 1 \) repeated nucleotides. Our main goal is to find the optimal reference strand, where optimality is measured in terms of the synthesis cost, denoted by \( \text{cost}(S_k) \). This cost is defined as the length of the shortest common supersequence of all strands, and in our analysis we derive high probability lower and upper bounds on this cost. To that
end, we start by representing our DNA synthesis problem as a homopolymer run-length constrained system associated with a unique, Entropy-maximizing Markov measure. This Markov measure plays the role of the previously mentioned “natural” distribution. Using a characterization of this measure we prove that for any \( k \geq 1 \), the optimal reference strand is ACGT.\(^1\) Our analysis is a generalization of the single batch analysis for the special case of \( k = 1 \) considered in [29]. This generalization to homopolymer run length constraint of length \( k \) adds another level of difficulty that requires techniques from the field of constrained systems.

### B. Related Work

The optimal synthesis problem was first raised in the context of coding theory in [26]. The authors of [26] studied the design of codes that minimize the synthesis time. One of the main objects that were studied is the maximal size of codes relative to their synthesis time. In addition, it was shown that, for array-based DNA synthesis techniques, by introducing redundancy to the synthesized strands, one can significantly decrease the number of synthesis cycles. The authors also derive the maximum amount of information per synthesis cycle assuming that the strands to be synthesized are arbitrary periodic sequences.

Another related line of works is on cost-constrained channels. First raised in Shannon’s paper [43], the cost constrained problem is considered as a noiseless channel where the transmission of each symbol incurs a cost. The goal is to find the capacity of such a channel, defined in two ways - probabilistic and combinatorial. A formula for the capacity of the channel was already sketched in Shannon’s paper, together with a proof of the equality between the probabilistic and the deterministic capacity. In [20] a formal proof for the capacity formula is given, together with a proof for the equality between the probabilistic and combinatorial definitions of capacity with some extensions (similar method was also used in [17]). In a more recent work, the authors of [25] studied the cost-capacity formula from a combinatorial point of view, and further studied the connection between the cost constrained and DNA synthesis that was drawn in [26]. Generally speaking, the synthesis problem can be formulated as a cost-constrained system in which the cost of a symbol is its synthesis time. Using their results, they managed to provide a formula for the number of strings of any length that can be synthesized in a given number of steps, and to the average cost of a symbol.

In this paper, we focus on a slightly different problem which can be thought of as the opposite side of the same coin. The most closely related paper to our work is [29], in which the authors study the problem of identifying the optimal synthesis time of a randomly chosen string. They consider the cases where the strands to be synthesized are either unconstrained or constrained, in the sense that the strands do not contain repeats of the same character (homopolymers). Our paper generalizes their results for the case of a general homopolymer run constraint of length \( k \geq 1 \) (i.e., \( k + 1 \) repeated characters are not allowed).

Similarly to [29], our work is motivated by both theoretical and experimental papers that tackle the problem of reducing the cost of DNA synthesis. Specifically, in terms of theoretical results, a few recent works have considered coding-based approaches for the analysis of the cost. For example, in [1], [16], [24], and [42], a somewhat different synthesis model which assumes that information is stored based on run length patterns in the strings was considered, for which the amount of information bits per synthesis time unit is analyzed.

Another large body of related work is on the study of the longest common subsequence (LCS) of random strings, e.g., [4], [6], [8], [13], [21], [28], and [32]. Specifically, it is well-known that for two strings of length \( n \), generated at random, the expected length of LCS is approximately \( \gamma n \), where \( \gamma > 0 \) is the Chvátal-Sankoff constant.

There is a large amount of prior works from the experimental point of view of DNA synthesis cost, e.g., [7], [10], [14], [18], [19], [23], [34], [35], [38], [39], and [44], and many references therein. The majority of these papers analyze empirically the cost when using ACGT as the reference strand. For example, in [38], it was observed that the cost of uniformly random strings is approximately Gaussian. In terms of the selection of a short reference strand, many algorithms have been proposed and tested empirically. Unfortunately, these heuristics do not provide any provable guarantees.

### C. Organization

The rest of this paper is organized as follows. In Section II we formulate our model, state our primary goals, and present our main findings. Section III is devoted to the proofs of our main results, and finally, in Section IV we conclude our paper and present a few intriguing questions for future research.

## II. Setup and Problem Statement

As mentioned above, the underlying problem in DNA synthesis is that strands of nucleotides with long repetitions of the same nucleotide are prone to errors, and thus we would like to avoid those DNA strands with more than a fixed number \( k \geq 1 \) of consecutive nucleotides of the same type. In practice, the maximum run length of each symbol in each strand is at most three. Throughout this paper, \( n \in \mathbb{N} \) denotes the strand length. Consider the following definition.

**Definition 1 (Strands Without \( k \) Homopolymers):** Fix \( n, k \in \mathbb{N} \). Let \( \mathcal{H}_{n,k} \subset \{A, C, G, T\}^n \) be the subset of all strands of length \( n \) with no \( k + 1 \) consecutively repeated characters.

Given \( \mathcal{H}_{n,k} \), let \( S_k \) be a subset of strands in \( \mathcal{H}_{n,k} \) with \( |S_k| = M \), for some \( M \in \mathbb{N} \). The set \( S_k \) is the pool of strands to be synthesized. We consider a popular synthesis process that produces many strands in parallel, step-by-step, using a fixed supersequence denoted by \( R \in \{A, C, G, T\}^* \) where \( \{A, C, G, T\}^* \) is the set of all finite strings over \( \{A, C, G, T\} \).

We will refer to \( R \) as the reference sequence. The machine

---

\(^1\)Given a string \( w \), we denote by \( \ldots w \ldots \) the infinite string generated by repeated concatenations of \( w \) with itself.
trace reconstruction, e.g., [2], [11], [12], [37], and [45].

be a part of the decoding process such as clustering, e.g., [36] and [40], and
which will not contain long runs only with high probability.

Henceforth, the cost of \( S \) is synthesized in cycles 2, 4, 5, 6, 7, \( S_2 \) is synthesized in cycles 1, 3, 4, 5, and similarly \( S_3 \) is synthesized in cycles 2, 4, 8.

Thus, in order to select \( S_k \) from \( H_{n,k} \), a different generative model is needed for selecting \( S_k \).

Following [29], we assume a certain generative model for the strands selection. In principle, it is possible to assume the same uniformity assumption as in [29], and analyze the cost of DNA synthesis. Although studying the uniform generative model is interesting on its own, in this paper, however, we opted to focus on the probabilistic model below, which we found more natural. To present our model, we start with a brief background on constrained systems. At this point, we would like to mention that although in the above we have focused our attention on quaternary alphabet (motivated by DNA genetic codes), our results hold for any alphabet \( \Sigma \) of cardinality \( |\Sigma| = r \geq 2 \); henceforth, we shall follow this generality.

A. Constrained Systems Recap

We provide here a brief background on the topic of constrained system. The notations and definitions that we use throughout the paper follow [30]. For a natural number \( n \in \mathbb{N} \), we denote by \([n]\) the set \([n] = \{0, 1, \ldots, n - 1\}\), and for a number \( t \in \mathbb{N} \) we let \( t + [n] = \{t + j : j \in [n]\} \). Fix a finite alphabet \( \Sigma \) of size \( |\Sigma| = r \). We denote by \( \Sigma^* \) the set of all finite strings over \( \Sigma \) and for \( w \in \Sigma^* \) we denote by \(|w|\) its length. For \( w, u \in \Sigma^* \) we denote by \( wu \) the string obtained by concatenating \( u \) to \( w \) and for \( n \in \mathbb{N} \), \( w^n \) denotes the concatenation of \( w \) with itself \( n \) times. For a string \( w = (w_0, w_1, \ldots, w_{n-1}) \in \Sigma^n \) and for a set \( I \subseteq [n] \) we denote by \( w_I \) the string obtained by restricting \( w \) to the coordinates in \( I \). For example, if \( w = (w_0, \ldots, w_{n-1}) \) with \( n \geq 4 \) and \( I = 1 + [3] = \{1, 2, 3\} \), then \( w_I = (w_1, w_2, w_3) \).

A constrained system is defined by a (possibly infinite) set \( \mathcal{F} \) of finite strings, \( \mathcal{F} \subseteq \Sigma^* \). We think of the set \( \mathcal{F} \) as a set of forbidden patterns. A constrained system \( \mathcal{X} = \mathcal{X}_\mathcal{F} \) comprises of the set of all finite strings that do not contain any string from \( \mathcal{F} \) as a substring, i.e., \( w \in \mathcal{X}_\mathcal{F} \) if there are no pairs of indices \( i, j \in \mathbb{N} \), \( i < j \) and for which \( w_i w_{i+1} \ldots w_j = \alpha \), for some \( \alpha \in \mathcal{F} \).

An equivalent way to describe a constrained system is using a graph. Specifically, let \( G = (V, E, L) \) be a finite graph with \( V \) being its vertex set, \( E \subseteq V \times V \) a set of (directed) edges, and \( L : E \rightarrow \Sigma \) a label function. A path \( \gamma \) of length \( n \) in \( G \) is a sequence of edges \( \gamma = (e_0, e_1, \ldots, e_{n-1}) \in E^n \) where \( e_i = (v_i, v_{i+1}) \) (notice that \( e_i \) ends in the vertex \( v_{i+1} \) starts from).

Fig. 1. Synthesis of three strands \( S_1 = \langle\text{CTAGC}\rangle, S_2 = \langle\text{AGTA}\rangle, \) and \( S_3 = \langle\text{CTT}\rangle \) using the reference sequence \( R = \langle\text{ACGTACG}\rangle \). The strand \( S_1 \) is synthesized by attaching the nucleotides in cycles 2, 4, 5, 6, 7, \( S_2 \) is synthesized in cycles 1, 3, 4, 5, and similarly \( S_3 \) is synthesized in cycles 2, 4, 8.

Table 1. Synthesis of three strands.

| Cycle | R | S1 | S2 | S3 |
|-------|---|----|----|----|
| 1     | A | C  | T  | T  |
| 2     | G | T  | A  | A  |
| 3     | C | G  | C  | C  |
| 4     | T | T  | T  | T  |
| 5     | A | A  | A  | A  |
| 6     | C | C  | C  | C  |
| 7     | T | T  | T  | T  |
| 8     | A | A  | A  | A  |

Figures may be missing or altered due to disintegration.
The label of \( \gamma \) is the string \( L(\gamma) \doteq L(e_0)L(e_1)\ldots L(e_{n-1}) \) and we say that \( \gamma \) starts at \( v_0 \) and ends at \( v_n \). A constrained system \( \mathcal{X} \) is the set of all finite strings obtained from reading the labels of paths in a labeled graph \( G \). We say that \( G \) is a presentation of \( \mathcal{X} \), or \( G \) presents \( \mathcal{X} \). Notice that there are many other different presentations for the same system.

A simple description of a labeled graph \( G \) can be obtained using the adjacency matrix \( A = A_G \). The adjacency matrix is a \( |V| \times |V| \) matrix where the \((i,j)\) entry is the number of edges going from state \( v_i \) to state \( v_j \) in \( G \). Fig. 2 below illustrates a graph that presents a system \( \mathcal{X} \) that comprises of all the strings in which no symbol appears next to itself. A useful property of constrained systems is irreducibility. A constrained system \( \mathcal{X} \) is called irreducible if for every \( \alpha, \beta \in \mathcal{X} \) there is a string \( \gamma \in \mathcal{X} \) such that \( \alpha \gamma \beta \in \mathcal{X} \). An equivalent requirement for \( \mathcal{X} \) to be irreducible is the existence of a presentation \( G \) of \( \mathcal{X} \) in which there is a path between every two vertices (the graph \( G \) is strongly connected).

The capacity of a constrained system \( \mathcal{X} \) is defined as \( \text{cap}(\mathcal{X}) \doteq \lim_{n \to \infty} \frac{1}{n} \log |B_n(\mathcal{X})| \), where \( B_n(\mathcal{X}) \doteq \{ s \in \Sigma^n : s \in \mathcal{X} \} \) is the set of all \( n \)-length strings in \( \mathcal{X} \) and the logarithm is to the base of \( |\Sigma| = r \). It is a well-known fact that the limit in this definition exists [30]. If \( G \) is a graph that presents an irreducible constrained system \( \mathcal{X} \), the Perron-Frobenius Theorem provides a characterization of \( \text{cap}(\mathcal{X}) \) using the eigenvalues of the adjacency matrix \( A_G \) (see, for example [30, Ch. 3]).

**Lemma 1 (Perron-Frobenius Theorem):** Let \( \mathcal{X} \) be an irreducible constrained system presented by a strongly connected graph \( G \). Then the following hold.

1. The adjacency matrix \( A_G \) has a real, simple, maximal eigenvalue \( \lambda \) called the Perron eigenvalue.
2. \( A_G \) has a positive left eigenvector \( \xi \) and a positive right eigenvector \( \phi \) associated with \( \lambda \), called Perron eigenvectors.
3. The only eigenvectors with all positive components are the Perron eigenvectors.

It can be shown that \( \text{cap}(\mathcal{X}) = \log \lambda \), where \( \lambda \) is the Perron eigenvalue of \( A_G \) [30, Ch. 3].

**B. Probabilistic Viewpoint of Constrained Systems**

To relate our DNA synthesis problem to constrained systems we need to associate \( \mathcal{H}_{n,k} \) with a Markov chain. Let \( \mathcal{X} \) be an irreducible constrained system over the alphabet \( \Sigma \). Each string is chosen independently at random from \( \mathcal{H}_{n,k} \), according to the (unique, stationary) entropy maximizing measure \( \mu \), defined in (1)–(3). The multiset \( S_k \) of the \( M \) strands to be synthesized is a batch. Given the generative model above, our goal is to analyze the synthesis cost, \( \text{cost}(S_k) \), defined in Definition 2, and characterize a reference string \( \mathbb{R}^* \) that approaches the smallest possible cost with high probability, as \( n \to \infty \).

**Example 2:** Assume \( \Sigma = \{0,1\} \) and \( k = 2 \). Thus, the system \( \mathcal{H}_{n,k}^0 \) comprises of all the binary strings in which there is a unique (stationary) Markov measure \( \mu \), such that the \( n \) marginals of \( \mu \) are supported on \( B_n(\mathcal{X}) \) for \( n \in \mathbb{N} \), and with Shannon-entropy rate equals to the capacity \( \text{cap}(\mathcal{X}) = \log \lambda \) (see [30, Ch. 3], [27, Ch. 4] or [46, Ch. 8]).

The measure \( \mu \) can be described by a stationary distribution \( \pi \) over the set of vertices \( V \) and a transition probabilities matrix \( Q \in \mathbb{R}^{|V| \times |V|} \). Given \( \pi, Q \), the probability of a path \( \gamma = (e_0, \ldots, e_{n-1}) \) in \( G \) is given by

\[
\mathbb{P}(\gamma) = \pi(v_0)Q_{v_0,v_1}Q_{v_1,v_2} \cdots Q_{v_{n-1},v_n},
\]

where \( e_i = (v_i, v_{i+1}) \). Since there are several paths with the same label, the probability of a string \( w \) under \( \mu \) is

\[
\sum_{\gamma : L(\gamma) = w} \mathbb{P}(\gamma).
\]

The constructions of \( \pi \) and \( Q \) are well-known [30]. Specifically, let \( \xi, \phi \) be the left and right Perron eigenvectors of \( A_G \) associated with the Perron eigenvalue \( \lambda \), normalized such that \( \sum_{i \in V} \xi_i \phi_i = 1 \). Then, the transition matrix \( Q \) is given by

\[
Q_{i,j} = \frac{(A_G)_{i,j} \phi_j}{\lambda \phi_i},
\]

and the stationary distribution is given by

\[
\pi_i = \phi_i \xi_i.
\]

Finally, the entropy rate of this Markov chain is exactly \( \text{cap}(\mathcal{X}) \), and \( \mu \) is the unique (stationary) measure that maximizes the entropy (\( \mu \) is a Markov measure of \( [r]^2 \) obtained using Kolmogorov’s extension theorem).

**C. Problem Statement and Main Result**

We now state our problem and main result rigorously. First, we define the relevant constrained system we study in this paper, dubbed \( k \)-run length constraint.

**Definition 3:** Let \( \Sigma \) be a finite alphabet and fix \( k \in \mathbb{N} \). Denote by \( F_k \) the set of all strings of length \( k+1 \) that comprise of a single symbol \( F_k \doteq \{ a^{k+1} : a \in \Sigma \} \). The \( k \)-run length (constrained) system is denoted by \( \mathcal{H}_{n,k}^\Sigma \) and is defined by the set \( F_k, \mathcal{H}_{n,k}^\Sigma = \mathcal{H}_{F_k} \). In words, \( \mathcal{H}_{n,k}^\Sigma \) is the set of all finite strings over \( \Sigma \) in which there are no runs of length \( k+1 \).

We write \( \mathcal{H}_{n,k}^{\Sigma_k} \) to denote the set of length \( n \) strings in \( \mathcal{H}_{n,k}^{\Sigma_k} \), i.e., \( \mathcal{H}_{n,k}^{\Sigma_k} = \mathcal{H}_{n,k}^{\Sigma_k} \cap \Sigma^n \). When \( \Sigma \) is clear from the context we will write \( \mathcal{H}_k, \mathcal{H}_{n,k} \) instead of \( \mathcal{H}_{n,k}^{\Sigma_k} \).

**Example 1:** Let \( \Sigma = [2] = \{0,1\} \) and let \( \mathcal{H}_{n,k}^{\Sigma_2} \) be the \( 2 \)-run length constrained system. The system \( \mathcal{H}_{n,k}^{\Sigma_2} \) comprises of all the finite binary strings \( w \) such that \( w \) does not contain any of the patterns 000, 111, i.e., there are no triples of similar consecutive symbols.

We next describe our generative model. Fix an alphabet \( \Sigma \) and numbers \( k, n, M \in \mathbb{N} \). We are to synthesize \( M \) strings over the alphabet \( \Sigma \). Each string is chosen independently at random from \( \mathcal{H}_{n,k}^{\Sigma_k} \), according to the (unique, stationary) entropy maximizing measure \( \mu \), defined in (1)–(3). The multiset \( S_k \) of the \( M \) strands to be synthesized is a batch. Given the generative model above, our goal is to analyze the synthesis cost, \( \text{cost}(S_k) \), defined in Definition 2, and characterize a reference string \( \mathbb{R}^* \) that approaches the smallest possible cost with high probability, as \( n \to \infty \).
are no runs of length 3 (or more) of the same symbol. Let $n = 8$, to generate a batch $S_k$ of size $M$, we are to select (or to generate) $M$ strands from $H_{n, k}$ according to $\mu$. Thus, at first, we find the graph $G$ that presents the system $H_k$ and its corresponding adjacency matrix $A_G$. In this case, The graph $G$ contains 4 states (vertices), and its adjacency matrix is

\[
A_G = \begin{bmatrix}
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 1 \\
1 & 1 & 0 & 0 \\
0 & 0 & 1 & 0
\end{bmatrix}.
\]

The Perron eigenvalue of $A_G$ is $\lambda = 1 + \sqrt{5}$ and the left and right Perron eigenvectors are $\zeta = \phi = (1, \frac{1 + \sqrt{5}}{2}, \frac{1 + \sqrt{5}}{2}, 1)$. Normalizing $\zeta$ and $\phi$ such that $\zeta \cdot \phi = 1$ we obtain $\zeta = \phi \approx (0.37175, 0.601504, 0.601504, 0.37175)$. Plugging everything to (2) and (3), we obtain

\[
Q = \begin{bmatrix}
0 & 1 & 0 & 0 \\
0.381966 & 0.618034 & 0 & 0 \\
0 & 0 & 0.618034 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix},
\]

and

\[
\pi \approx (0.138198, 0.361807, 0.361807, 0.138198).
\]

The $M$ strings in $S_k$ are chosen according to $\pi$, $Q$ (or similarly, according to $\mu$).

Our main result below provides asymptotically tight high probability lower and upper bounds on the synthesis cost. These bounds imply that with high probability the periodic sequence $012\ldots(r-1)$ approaches the smallest possible cost for synthesizing all the strands in a batch $S_k$.

**Theorem 1** (Optimal reference): For an alphabet $\Sigma = [r]$ and for any $k \geq 0$, let $S_k \subseteq H_{n,k}$ be a batch of $M$ i.i.d. strings chosen according to $\mu$. Then, there exists a constant $C_0 \leq \frac{e}{2}$, such that with probability at least $1 - 1/n$,

\[
\text{cost}(S_k) \leq C_0 \cdot n + O\left(\sqrt{n \cdot \log \max(M, n)}\right).
\]

On the other hand, fix $\epsilon > 0$. There exists a constant $C_{\epsilon, k, r}$, such that for $M \geq C_{\epsilon, k, r}$, with probability at least $1 - e^{-\epsilon n}$,

\[
[C_0 - \epsilon] \cdot n \leq \text{cost}_{\epsilon, r}(S_k) \leq \text{cost}(S_k),
\]

where $R^* = 012\ldots(r-1)$.

**Remark 1**: The optimal reference string is not unique. For example, every periodic string with period $r$ that includes all the alphabet symbols is optimal.

**Remark 2**: A specific characterization of the various constants (such as, $C_0$ and $M_{\epsilon, k, r}$) as well as the $O(\cdot)$ term in Theorem 1 will be provided in the proof (see, the statement of Theorem 3). While in the practice of DNA synthesis the parameters $n, M,$ and $k$ are concrete numbers, to facilitate the asymptotic study of the problem we focus on the following relevant scenario: $n$ is growing, $M$ is significantly larger than but polynomial in $n$, and $k$ is either a constant or a growing function of $n$.

As a by-product of the proof of Theorem 1, we obtain the following result on the Markov measure $\mu$.

**Theorem 2**: Let $\Sigma$ be a finite alphabet and let $k \in \mathbb{N}$. Let $\mu$ be the Markov measure that is associated with the system $H_k$ with $\pi$ its stationary distribution and $Q$ its transition probabilities matrix. Then for every $a, c \neq b \in \Sigma$ and every $i \leq k$, $\mu(ab^i) > \mu(ab^{i+1})$ and for any $w \in \Sigma^{k+1}$, $\mu(w)$ depends only on the runs in $w$ and not on the symbols. In other words, for any string $w = w_1\ldots w_{m-1} \in H_{m,k}$ and any symbols $a, b \neq a_{m-1}$, we have $\mu(w_0\ldots w_{m-1}a) < \mu(w_0\ldots w_{m-1}b)$. Moreover, if $w$ can be written in terms of its maximal runs, i.e., $w = w^{a_1}_{n_1}w^{a_2}_{n_2}\ldots w^{a_k}_{n_k}$ where the $n_i \in \mathbb{N}$ are maximal, and if $w$ is a string that is also written in terms of its maximal runs, $u = (u^{a_1}_{n_1}u^{a_2}_{n_2}\ldots u^{a_k}_{n_k})$, then $\mu(w) = \mu(u)$.

Before diving into the proofs, we are now able to better explain our generative model for selecting the set $S_k$. In the next section, we show that for $k = 0$, i.e., when no two consecutive appearances of the same symbol are allowed, our generative model with respect to $\mu$ coincides with the uniform generative model that appears in [29]. Thus, it can be viewed as an extension of the generative model presented in [29]. In fact, for all $k \geq 0$, the measure $\mu$ is closely related to the uniform distribution. If $G$ is the graph that presents an irreducible constrained system, the measure $\mu$ gives a uniform probability to all $n$-length strings with a fixed starting and ending vertex in $G$ (see (1) and the formula for $Q_{i,j}$).

Second, the Markov measure $\mu$ naturally extends to strands of any length $n$, while the uniform measure does not (this follows since the uniform distribution on $H_{n,k}$ is not shift invariant, or, does not satisfy the conditions for Kolmogorov’s extension theorem). Shift invariance in one-dimensional strings amounts to Kirchhoff’s law, i.e., for every $m \in \mathbb{N}$,

\[
\sum_{b \in \Sigma} \text{Pr}(b a_0 a_1 \ldots a_m) = \sum_{b \in \Sigma} \text{Pr}(a_0 a_1 \ldots a_m b).
\]

This is not satisfied in the uniform generative model. For example, $H^{2,4}_{4,2}$ (the binary strings of length 4 with no runs of length 3) contains 10 strings. Two of them start with 001 (the strings 0010 and 0011) and only one ends with 001 (the string 0011). Thus, there is a different probability of seeing 001 in different positions and the distribution is not shift invariant. This implies that if one tries to extend this distribution to longer strings (in our example, to strings of length 5), the result does not obey the same generative model. In this example, since 00100 and 00101 obtained from 0010 both should have the same probability $\text{Pr}(00100) = \text{Pr}(00101) < \text{Pr}(0010)$. But 00110 is the only string obtained from 0011 and therefore $\text{Pr}(00110) = \text{Pr}(00111) = \text{Pr}(00110)$. Thus, the analysis depends on the length of the strings and one cannot deduce properties of long strings by analyzing shorter strings in a straightforward manner. Since in this work we study the asymptotic cost of synthesizing constrained strands, the shift invariant property is needed.

Thirdly, assuming the information string is obtained with no constraints, it must first be encoded to obey the $k$-homopolymer constraint. If this encoding process is performed using a finite state encoder (see [27], [30]), the obtained constrained strings obey a Markovian distribution supported on $H_{n,k}$. Among those, the maxentropic distribution is $\mu$. 

Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.
Finally, we would like to mention that other interesting

generative models can be pursued. In this paper, we study

a generative model which is based on shift invariant distri-

butions; among those, we chose to start with the one that

maximizes the entropy.

III. PROOF OF THEOREM 1

In this section we prove Theorem 1. The proof is divided to

three parts. In the first part, we provide a construction of the

digraph $G$ that presents $\mathcal{H}_k$ from the well known de Bruijn
digraph of span $k$. We also show that the adjacency matrix $A_G$ of $G$
can be obtained from the adjacency matrix of the de Bruijn
digraph of span $k$. By doing so, a specific structure of $G$ and $A_G$
reveals. Some properties are immediately obtained from this

structure. In the second part, we prove Theorem 1 by using

Theorem 2, along with Hoeffding’s inequality and a certain

stochastic domination argument. The last part is devoted for

the proof of Theorem 2. This is done by a characterization of

some properties of $\mu$, obtained using the structure of $G$

and the relation between constrained systems and Markov chains.

A. Constructing $G$ and $A_G$ From De Bruijn Graph

We begin by constructing the digraph $G$ that presents $\mathcal{H}_k$

and its adjacency matrix. The graph $G$ is constructed from the

de Bruijn digraph $G'$ of span $k$ which is described next. Let

$G' = G'([r], k) = (V', E', L')$ where $V' = [r]^k$ is the set of all

$k$-length strings over $[r]$. The set $V'$ inherits the natural

order obtained by interpreting the vertices as numbers written

in their base-$r$ representation. To construct the set of edges $E'$,
draw a directed edge from $u = (a_0, \ldots, a_{k-1}) \in V'$ to $v = (b_0, \ldots, b_{k-1}) \in V'$ if $b_i = a_{i+1}$ for every $i \in [k-1]$.

In words, we draw an edge from $u$ to $v$ if the $k-1$ suffix of $u$
is equal to the $k-1$ prefix of $v$. The label of the edge $(u, v)$ is
$L'((u, v)) = b_{k-1}$. The graph $G'$ is called the de Bruijn
digraph of span $k$.

Notice that every path of length $k$ in $G'$ that ends at a vertex
$v = (b_0, \ldots, b_{k-1})$, yields the string $L(\gamma) = b_0 \ldots b_{k-1}$.

Thus, with a slight abuse of notation we will sometimes

use the vertex symbol $v$ instead of its corresponding $k$-tuple
$b_0 \ldots b_{k-1}$. Notice that an appearance of an edge $(u, v)$ in a
path in $G'$ can be related to the appearance of the $k+1$-length
substring $uL((u, v)) = ub_{k-1} = a_0v$.

The graph that presents $\mathcal{H}_k$ can be easily obtained from the
de Bruijn digraph $G'$ by removing some edges. Since runs of

length $k+1$ are forbidden in $\mathcal{H}_k$, and since a $k+1$ run

corresponds to a self-edge in $G'$, to obtain a presentation

$G = G(\Sigma, k) = (V, E, L)$ of $\mathcal{H}_k$ we let $V = V'$, and $E = E'$
after eliminating self loops from $E'$, i.e., we take $E = E' \setminus \{(v, v) : v \in V\}$. Notice that the only self loops in $E'$ are edges $(v, v) \in E'$ of the form $v = a^k$ where $a \in \Sigma$, i.e., $v$ corresponds to a $k$-length string comprises of a single letter. Since self loops in $G'$ correspond to $k+1$-tuples of the form $a^{k+1}$ for some $a \in \Sigma$, and since $G$ is obtained after all the self-loops are removed from $G'$, the graph $G$ presents the system $\mathcal{H}_k$. From the structure of $G$ we immediately obtain that there is a path from any vertex $u$ to any vertex $v = (b_0, \ldots, b_{k-1})$ in $G$, by walking over the

edges labeled with $b_0, b_1, \ldots, b_{k-1}$. The only case in which

this is not possible is when $uv$ contains a forbidden pattern.

In this case, the path $cb_0, \ldots, b_{k-1}$, for some $c \neq b_0$, is a path

from $u$ to $v$ in $G$. This immediately implies the following
corollary.

Corollary 1: The $k$-run length system $\mathcal{H}_k^r$ is irreducible.

Remark 3: Notice that the presentation of $\mathcal{H}_k$ described

above is not the minimal (in terms of number of vertices)

representation. We will, however, consider this presentation

throughout the paper since it is more accessible for analysis. This

accessibility follows from two facts. The first, is that the

adjacency matrix of $G$ has some structure which is preserved

when $k$ is increased. The second, is that this presentation is

lossless, which means that fixing a starting vertex and an

ending vertex, different paths generate different strings. The

latter property will be used later to bound the cost.

Example 3: Figure 3 shows the formation of $G$ from $G'$.
The LHS graph is a de Bruijn graph of order $k = 2$ over the

binary alphabet. The RHS graph is the graph presenting $\mathcal{H}_2$, 

obtained by removing the 2 self loops in $G'$.

Similarly to the construction of $G$, the adjacency matrix $A_G$
of the graph $G$ can be obtained from the adjacency matrix of

the de Bruijn graph of span $k$. Obviously, to get $A_G$ from $A_G'$

we only need to set zero the entries $A_G(v, v)$. The following

lemma provides a formal construction.

Lemma 2: Let $G = G(\Sigma, k)$ be the graph presentation of

$\mathcal{H}_k$. Then, the adjacency matrix of $G$, denoted by $A_G^k \in \mathbb{N}^{r^k \times r^k}$, is given by

$$
(A_G^k)_{i,j} = \begin{cases} 
0, & \text{if } (i,j) \in A \\
1, & \text{otherwise},
\end{cases}
$$

(4)

where $A = \{(i,j) : j \notin \{(i \mod r) \cdot r + [r] \} \text{ or } i = j = 0 \text{ (mod } \frac{r^{k+1} - 1}{r-1})\}$. We will write $A_k$ instead of $A_G^k$ when the alphabet is clear from the context or if it is irrelevant.

The structure of the adjacency matrices $A_k$ is best seen by

examples.

Example 4: For $\Sigma = [2]$, the adjacency matrices $A_k^2$ and

$A_k^3$ corresponding to $\mathcal{H}_k^2$ and $\mathcal{H}_k^3$, respectively, are:

$$
A_k^2 = \begin{bmatrix}
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 1 \\
1 & 1 & 0 & 0 \\
0 & 0 & 1 & 0
\end{bmatrix}
$$

and

$$
A_k^3 = \begin{bmatrix}
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\
0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{bmatrix}
$$

Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.
For $\Sigma = [3]$, the adjacency matrix $A_2^{[3]}$ corresponding to $H_2^{[3]}$ is

$$A_2^{[3]} = \begin{bmatrix}
0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\
1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\
1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 
\end{bmatrix}.$$  

The numbers in bold are those numbers that were changed from the standard de Bruijn matrix due to the removal of self-loops. Notice that $A_2^{[2]}$ comprises of $|\Sigma| = 2$ “main blocks” where each block comprises of $2^{2-1}$ “stairs” of length $|\Sigma| = 2$ each, and $A_3^{[2]}$ comprises of $|\Sigma| = 2$ main blocks where each block comprises of $2^{2-1}$ “stairs” of length 2 each. The matrix $A_2^{[3]}$ comprises of 3 main blocks where each block comprises of $3^{2-1}$ “stairs” of length each. In general, if $H_k^{[r]}$ is a $k$-run length constrained system over $\Sigma = [r]$, then the corresponding adjacency matrix, $A_1^{[r]}$, comprises of $r$ main blocks, each block has $r^{k-1}$ “stairs” where each stairs contains $r$ ones or $r-1$ ones and a “bold” zero. Moreover, the zeros in bold correspond to vertices that have self loops in the de Bruijn graph of span $k$. Those vertices are exactly the vertices that are labeled as $a^k$ for some $a \in \Sigma$. Interpreting the labels of the vertices as base $r$ numbers, we obtain that for every $i \in [r]$, the $i$th number in the $i r^{k-1} - 1$-th “stair” of the $i$th block is a (bold) 0.

We notice that for the special case of $k = 1$, the adjacency matrix $A_1$ has the following form. With $\Sigma = [r]$, observe that every row contains $r-1$ ones and a single zero. The zeros are located on the diagonal of $A_1$. It is immediate to see that the all-one vector $1$ serves as a left and as a right eigenvector with eigenvalue $r-1$ (considered as a row or a column vector).

Thus, normalizing the left and right eigenvectors, $\xi, \phi$ such that $\sum_{i \in [r]} \xi_i \phi_i = 1$ we obtain that $\phi_i = \xi_i = \frac{1}{r^k}$. As a result, the characterization of $\mu$ is simple.

**Lemma 3:** For the special case of $k = 1$, the Markov chain $\mu$ is the uniform distribution.

Therefore, in the special case of $k = 1$, considered in [29], we see that our generative assumption on the set of strands $H_k$ coincides with the probabilistic model assumed in [29].

### B. The Structure of $\mu$ Implies Theorem 1

In this section we show that Theorem 1 follows from the structure of $\mu$. Specifically, Theorem 1 follows almost immediately from Theorem 2, stated here for convenience.

**Theorem 2:** Let $\Sigma$ be a finite alphabet and let $k \in \mathbb{N}$. Let $\mu$ be the Markov measure that is associated with the system $H_k$ with $\pi$ its stationary distribution and $Q$ its transition probabilities matrix. Then for every $a, c \neq b \in \Sigma$ and every $i \leq k$, $\mu(ab^i c) > \mu(ab^{i+1})$ and for any $w \in \Sigma^{k+1}$, $\mu(w)$ depends only on the runs in $w$ and not on the symbols. In other words, for any string $w = w_0 \ldots w_{m-1} \in H_{m,k}$ and any symbols $a, b \neq w_{m-1}$, we have $\mu(w_0 \ldots w_{m-1} w_{m-1} b) > \mu(w_0 \ldots w_{m-1} a) = \mu(w_0 \ldots w_{m-1} b)$. Moreover, if $w$ can be written in terms of its maximal runs, i.e., $w = w_0^{n_0} w_1^{n_1} \ldots w_k^{n_k}$ where the $n_i \in \mathbb{N}$ are maximal, and if $u$ is a string that is also written in terms of its maximal runs, $u = (u_0^{n_0} u_1^{n_1} \ldots u_k^{n_k})$, then $\mu(w) = \mu(u)$.

The proof contains two parts. At first, we show that $R^*$ is optimal for a single string $S$, chosen according to $\mu$. This is done using a stochastic domination argument. We then apply Hoeffding’s inequality for Markov chains to prove the optimality of $R^*$ for a batch $S_k$ of $M$ strings. The following result is needed (see [31, Corollary 1] or [31]).

**Lemma 4 (Hoeffding’s Inequality for Markov Chains [31]):** Let $\{X_i\}$ be an irreducible Markov chain on a finite state space $V$, with initial distribution $\pi$ (the stationary distribution).
Let $f : V^2 \rightarrow [a, b]$ be a real-valued function evaluated on the edges of the Markov chain. Then for any $t > 0$,  
\[
P_\pi \left( \left| \frac{1}{n} \sum_{i=0}^{n-1} f(X_i, X_{i+1}) - E_\pi [f(X_0, X_1)] \right| \geq t \right) 
\leq 2e^{-\frac{2nt^2}{(b-a)^2 \text{HitT}^2}},
\]
(5)  
where $P_\pi(\cdot)$ and $E_\pi[\cdot]$ are the probability and expected value when $\pi$ is the initial distribution, and  
\[
\text{HitT} \triangleq \max_{(x_0, x_1) \in V^2} E \left[ T_{(y_0, y_1)} \mid (X_0, X_1) = (x_0, x_1) \right],
\]
with $T_{(y_0, y_1)} \triangleq \inf \left\{ n \geq 0 \mid (X_{n+1}, X_{n+2}) = (y_0, y_1) \right\}$ is the first hitting time of edge $(y_0, y_1)$.  

We are now in a position to prove Theorem 1 using Theorem 2. As mentioned right after the statement of Theorem 1, we in fact prove the following result, where the various constants in Theorem 1 are characterized.  

**Theorem 3 (Optimal Reference):** For an alphabet $\Sigma = [r]$ and for any $k \geq 0$, let $S_k \subseteq \mathcal{H}_{n,k}^r$ be a batch of $M$ i.i.d. strings chosen according to $\mu$. Then, there exists a constant $\frac{t}{2} \leq C_0 \leq \frac{t^2}{4}$, such that with probability at least $1 - 1/n$,  
\[
\text{cost}(S_k) \leq \left[ C_0 + \sqrt{\frac{2\text{HitT}^2 \log \max(M, n)}{n}} \right] \cdot n.
\]
On the other hand, fix $\epsilon > 0$ and let $M \geq \frac{(C_0 - \epsilon) \log r}{\text{HitT}^2 \log 2}$. Then, with probability at least $1 - e^{-n}$,  
\[
[C_0 - \epsilon] \cdot n \leq \text{cost}_R(S_k) \leq \text{cost}(S_k),
\]
where $R^* = 012 \ldots (r - 1)$.  

Proof of Theorem 3: Let $S \in S_k$ be a length-$n$ string chosen according to $\mu$ and let $R \in \Sigma^*$ be any reference sequence. Extend $R$ by concatenating $01 \ldots (r - 1)$, so that $R$ will surely be a supersequence of $S$. Let $\tau_i(R)$ denote the index of the symbol of $R$ that is used to print the $i$th symbol of $S$. Define $X_0 \triangleq \tau_0(R)$ and $X_i \triangleq \tau_i(R) - \tau_{i-1}(R)$, for $i \geq 1$. Notice that the probability that cost($S$) $\leq \rho$, for some $\rho \in \mathbb{N}$, is given by  
\[
P(\text{cost}(S) \leq \rho) = P(\tau_n(R) \leq \rho) = P\left( \sum_{i=0}^{n-1} X_i \leq \rho \right).
\]
Now let $R^* = 01 \ldots (r - 1)$ and similarly, let $\tau_i(R^*)$ denote the index of the symbol of $R^*$ that is used to print the $i$th symbol of $S$. Define $Y_0 \triangleq \tau_0(R^*)$ and $Y_i \triangleq \tau_i(R^*) - \tau_{i-1}(R^*)$, for $i \geq 1$. For an arbitrary (predetermined and known) reference $R$, the random variables $\{X_i\}$ are not i.i.d. However, the support of these random variables contains at most $r$ integers, which are the distances to the next occurrences of the $r$ symbols. In some cases, the support contains at most $r - 1$ integers due to the run length constraint. The support of the random variables $\{Y_i\}$ is $\{1, 2, \ldots, r\}$ (or, due to the run length constraint, $\{1, 2, \ldots, r - 1\}$), i.e., the support comprises of the minimal integers possible. We write  
\[
P(\tau_i(R) \geq t) = P(\tau_i(R) + X_i \geq t) 
\geq E[P(\tau_i(R) + X_i \geq t \mid \tau_i(R))].
\]
Notice that given $R$, both $\tau_{i-1}(R)$ and $X_i$ are functions of $S_{[i-1]}$. Indeed, $X_i$ is the number of steps it takes to synthesize $S_i$ after synthesizing $S_{i-1}$ and can be written as a function of $S_{[i-1]}$ and $R$ (in fact, only the last $k$ symbols in $S_{[i-1]}$ suffice). Similarly, $\tau_{i-1}(R)$ can be written as a function of $S_{[i]}$ and thus, also of $S_{[i-1]}$. Conditioning on the RV $\tau_{i-1}(R)$ is, by definition, conditioning on a sub-$\sigma$-algebra generated by the inverse image of $\tau_{i-1}(R)$. This sub-$\sigma$-algebra is obtained by cylinder sets on the first $l$ symbols of $S$, or, in other words, it provides some information on the first $l$ symbols of $S$. In addition, it gives a complete information about the $l$th symbol $S_{l-1}$ since knowing $R$ and the exact index of the symbol in $R$ that is used to synthesize $S_{l-1}$, identifies $S_{l-1}$.  

Let $G$ be the graph that presents $\mathcal{H}_{n,k}^r$ as constructed in Section III-A in which every vertex corresponds to a $k$-tuples over the alphabet $[r]$. Considering $S$ as a random walk on $G$, and using the Markov property of $S$, when estimating $P(X_i + \tau_{i-1}(R) \geq r \mid \tau_{i-1}(R))$, conditioning on $\tau_{i-1}(R)$ amounts to a certain distribution on the vertices in $G$ in which the random walk $S_{[i]}$ ends. Since $S_{l-1}$ is known, this distribution is supported only on vertices in $G$ that correspond to $k$-tuples that end with $S_{l-1}$.  

Let $w = (w_0 \ldots w_{k-1})$ be a vertex in $G$ that ends with $S_{l-1}$, i.e., $w_{k-1} = S_{l-1}$. Let $a, b \neq S_{l-1}$ be two symbols and denote $u = (w_0 \ldots w_{k-1} u_{a,b}), v = (w_0 \ldots w_{k-1} v_{a,b})$. Theorem 2 states the transition probability from $u$ to $v$ is equal to the transition probability from $w$ to $v$. In other words, $Q_{w,u} = Q_{w,v}$. In addition, if $w^* = (w_1 \ldots w_{k-1} S_{l-1})$, the transition probability from $w$ to $w^*$, $Q_{w,w'}$, is strictly smaller than $Q_{w,u}$. Thus, if the walk $S_{[i]}$ ends with $w$, $S_i$ could be any symbol from $[r] \setminus \{S_{l-1}\}$ with equal probability, and $S_{l-1}$ with strictly smaller probability.  

Since this is true for every such vertex $w$ that ends with $S_{l-1}$, it is also true for a mixture of vertices that end with $S_{l-1}$. Which in turn, implies that conditioned on $\tau_{i-1}(R)$, $X_i$ has equal probability to be any one of $r - 1$ integers, and there is at most one integer with strictly smaller probability (the integer that corresponds to the next appearance of $S_{l-1}$ in $R$). Therefore, conditioned on $\tau_{i-1}(R)$, $X_i$ is minimized if those $r - 1$ integers with equal (positive) probabilities are $1, 2, \ldots, r - 1$ and the integer $r$ with the smallest probability. But this is exactly $Y_i$ since $Y_i$ is the number of steps it takes to synthesize $S_i$ after synthesizing $S_{l-1}$ with the reference $R^*$. Thus, we obtain  
\[
P(\tau_{i-1}(R) + X_i \geq t \mid \tau_{i-1}(R)) 
\geq P(\tau_{i-1}(R) + Y_i \geq t \mid \tau_{i-1}(R)).
\]
Since this is true for every $\tau_{i-1}(R)$ it is also true in expectation. Putting everything together, we obtain  
\[
P(\tau_i(R) \geq t) = E[P(\tau_{i-1}(R) + X_i \geq t \mid \tau_{i-1}(R))] 
\geq E[P(\tau_{i-1}(R) + Y_i \geq t \mid \tau_{i-1}(R))] 
\geq P(\tau_{i-1}(R) + Y_i \geq t),
\]
(6)  

Next, for $j \leq l \leq [n]$, define $\bar{X}_j \triangleq \sum_{i=j}^l X_i$ and $\bar{Y}_j \triangleq \sum_{i=j}^l Y_i$. Let $j < l < n$ and consider $P(\bar{X}_j + \bar{Y}_{j+1} \geq t)$.  

Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.
We have

\[
P \left( X_0^j + Y_{j+1}^i \geq t \right) = P \left( X_0^{j-1} + X_j + Y_{j+1}^i \geq t \mid X_0^{j-1}, Y_{j+1}^i \right)
\]

\[
= E \left[ P \left( X_0^{j-1} + X_j + Y_{j+1}^i \geq t \mid X_0^{j-1}, Y_{j+1}^i \right) \right]
\]

\[
(b) \geq E \left[ P \left( X_0^{j-1} + Y_j + Y_{j+1}^i \geq t \mid X_0^{j-1}, Y_{j+1}^i \right) \right]
\]

\[
= P \left( X_0^{j-1} + Y_j \geq t \right).
\]

To account for (b) notice that conditioning on \((X_0^{j-1}, Y_{j+1}^i)\) is equivalent to using the reference sequence \(R\) up to the printing of \(S_j\) and then switching to the reference sequence \(R^*\), while knowing the first \(j\) symbols \(S_0^{j-1}\), and knowing the rest of the symbols \(S_{j+1}^i\) as a function of \(S_j\). We stress that although conditioning on \(X_0^{j-1}\) is the same as conditioning on \(S_{j+1}^i\) (since the reference sequence is known and \(X_0\) implies \(S_0\)), conditioning on \(Y_{j+1}^i\) is not the same as conditioning on \(S_{j+1}^i\) (since \(S_j\) is unknown). This is because \(Y_j\) is the number of steps it takes to print the \(i\)th symbol in \(S\) after the \((i-1)\)th symbol was printed, using the reference \(01 \ldots (r-1)\). Thus, (b) follows from Theorem 2 together with the symmetry of the run length constraint. Indeed, given the prefix \(S_0^{j-1}\), the probability that \(S_j = a \neq S_{j-1}\) is the same for every \(a \in \Sigma \setminus S_{j-1}\) and is larger than the probability that \(S_j = S_{j-1}\). This implies that if \(S_{j-1}\) was printed using the symbol in index \(\tau_{j-1}(R)\), putting the next appearance of the symbol \(S_{j-1}\) in index \(\tau_{j-1}(R) + r\) minimizes the cost.

From symmetry, it is clear that the first symbol is distributed uniformly over \([r]\); hence \(P(X_0 \geq t) \geq P(Y_0 \geq t)\). This, together with (6) and (7), imply that for every \(R\),

\[
P \left( \tau_n(R) \geq t \right) \geq P \left( \tau_n(R^*) \geq t \right),
\]

or, equivalently,

\[
P \left( X_0^n \leq t \right) \leq P \left( Y_0^n \leq t \right).
\]

To show that \(R^*\) is optimal for a batch \(S_k\) of \(M\) i.i.d. strings, we use Lemma 4. As mentioned in Remark 3, the graph \(G\) that presents \(H_k\) is lossless, i.e., two different paths that start at the same vertex and end at the same vertex generate different strings. Moreover, after \(k\) steps, every path with the same \(k\)-length prefix arrive at the same vertex. Thus, the Markov process \(\{Y_i\}\) can be obtained using the graph \(G = (V, E, L)\) that presents the system \(H_k\) by replacing the label function \(L\) with \(f : V^2 \to [0, r]\), denoting the cost of synthesis for each edge. For example, with \(r = k = 3\), the value of \(f(000, 002) = 2\) and \(f(010, 100) = 3\). Under this “new” setting, the strong law of large numbers for Markov chains implies that the Cesàro mean of \(Y_i\)s converges to \(E[f]\). We can now use Hoeffding’s inequality as follows. Since the graph \(G\) is irreducible, denoting \(\rho \triangleq (E[f] - \epsilon)n\) and using (5) yields

\[
P \left( Y_0^{n-1} \leq \rho \right) \leq 2e^{-cn},
\]

where \(c = \frac{2e^2}{r^2 \text{Hit}^2} > 0\) is a (finite) constant due to irreducibility of \(G\).

Let us start by bounding the cost from strands. To this end, we define a few notations. For two strands \(R, S \in \Sigma^*\), let \(E_R(S)\) denote the event that \(R\) is a superstring of \(S\). Let \(E_R(S_k)\) denote the event that \(R\) is a superstring of all strands in \(S_k\), that is, \(E_R(S_k) \triangleq \bigcap_{S \in S_k} E_R(S)\). For an integer \(\rho\), let \(E_R(S_k)\) denote the event that there exists a strand \(R \in \Sigma^{\leq \rho}\) such that the event \(E_R(S_k)\) holds. Thus, by the union bound,

\[
P \left( \text{cost}(S_k) \leq \rho \right) = P \left( E_R(S_k) \right)
\]

\[
\leq \sum_{R \in \Sigma^{\leq \rho}} P \left( E_R(S_k) \right)
\]

\[
\leq r^\rho \cdot 2^M \cdot e^{-cM_0} \exp \left[ n \left( (E[f]) - \epsilon \right) \log r + M \log 2/n - cM \right].
\]

Thus, if \(M \geq (E[f] - \epsilon) \log r/(c - \log 2/n)\), then the probability that the cost of synthesis is less than \(n \cdot (E[f]) + \epsilon n\) goes to zero exponentially fast. On the other hand, let \(\rho' \triangleq (E[f] + \epsilon)n\) and obtain

\[
P \left( \text{cost}(S_k) \geq \rho' \right) \leq M \cdot P \left( \text{cost}(S) \geq \rho' \right)
\]

\[
\leq M \cdot P \left( Y_0^{n-1} \geq \rho' \right)
\]

\[
\leq M \cdot 2e^{-cn}
\]

\[
\leq n^{-1},
\]

where (a) follows by choosing a specific reference sequence \(R^*\), (b) follows from Lemma 4, and (c) follows by taking \(\epsilon = \sqrt{\frac{2 \text{Hit}^2 \log \max(M, n)}{n}}\). This concludes the proof. □

Remark 4: The constant \(C_0\) that appears in the statement of Theorem 1 is the expected cost per symbol, which is a function of \(\mu\) and of the reference sequence \(R^*\). While estimating the exact value of \(C_0\) requires estimating the distribution \(\mu\), it is beyond the scope of this paper, from Theorem 2 we can deduce that \(\frac{\tau}{2} \leq C_0 \leq \frac{\tau + 1}{2}\). Moreover, for \(k = 1\) (in which runs of length 2 are forbidden) the constant \(C_0 = r/2\) while for \(k = \infty\) (in which there is no restriction on the runs) the constant is \(C_0 = r^{k+1}/2\).
Remark 5: The result in this paper can be generalized immediately to any shift invariant measure that satisfies the properties given in Theorem 2.

Since we showed how Theorem 1 is obtained from Theorem 2, it is left to prove Theorem 2. The rest of the paper is devoted for that cause.

C. Proof of Theorem 2

In this section we prove Theorem 2. We first give an overview of the proof idea. Throughout the section, when considering a graph presentation of $H^r_k$, we will always use the graph $G$ obtained from the de Bruijn graph as explained in Section III-A. Therefore, the adjacency matrix of the graph presenting $H_k$, is the adjacency matrix $A_k$ given in Lemma 2.

Recall that the measure $\mu$ is determined by the stationary distribution $\pi$ and the transition probabilities matrix $Q$. Both $\pi$ and $Q$ are determined by a right Perron eigenvector $y$ of $A_k$ and by the corresponding Perron eigenvalue $\lambda_k$. Since Theorem 2 is implies by $\mu$, it suffices to find $y$ and $\lambda_k$ to prove the theorem.

Finding explicit expressions for $y$ and $\lambda_k$ is not an easy task. However, it is not necessary to explicitly find $y$ and $\lambda_k$ to prove Theorem 2, but suffices to only find the structure of $y$ as a function of $\lambda_k$. Using this structure, the proof of Theorem 2 will follow. Thus, the main work in this section is to describe the structure of $y$ using $\lambda_k$.

Since $y$ will be presented as a function of $\lambda_k$, we start with presenting the polynomial that is used to determine $\lambda_k$. We then present an auxiliary polynomial denoted $g_k,r$, together with some properties of $g_k,r$. Using this auxiliary polynomial and its properties, we provide a recursive construction for the right Perron eigenvector $y$ of $A_k$ from the right Perron eigenvector $x$ of $A_{k-1}$. Together with the right Perron eigenvector for $A_1$, this recursive construction allows us to present $y$ as a function of $\lambda_k$ and to study its structure. We then use this structure to prove Theorem 2.

For an alphabet $[r]$, every vertex in $G$ can be described by its corresponding $k$-tuple, or alternatively, as a number in a base-$r$ representation. Thinking of vertices as numbers in their base-$r$ representation will make the proofs easier to follow.

Example 5: The zeros in bold in $A_k^r$ in Example 4 correspond to vertices of the form $a^k$ for some $a \in \Sigma$. Thinking of vertices as base-$r$ numbers, vertices of the form $a^k$ correspond to $i r^{k-1}$ for a number $i \in [r]$.

Before we dive in, we provide a flowchart in Fig. 4 to draw the connections between the different results. This should serve as a road-map to see which lemmas and theorems are used in the proofs of other results.

We start with an explicit formula for the capacity of $H^r_k$ (see, e.g., [15], [33]).

Theorem 4 (Capacity Formula): Let $H^r_k$ be a $k$-run length system over the finite alphabet $\Sigma = [r]$. The capacity of $H^r_k$ is given by $\text{cap}(H^r_k) = \log \lambda_k$ where $\lambda_k$ is the largest real root of the polynomial

$$h_{k,r}(z) := z^k - \sum_{i=0}^{k-1} (r-1) z^i = \frac{z^k (z-r) + 1}{z-1}.$$  \hfill (14)

When the alphabet is clear from the context we will write $h_k(z)$ instead of $h_{k,r}(z)$.

Remark 6: Notice that in the binary case, there is a natural correspondence between the $k$-run length system $H_k$ and the $(0,k-1)$-run length limited (RLL) constrained system denoted $X_{k-1}$. Under the $(0,k-1)$-RLL constraint, a string $x \in X_{k-1}$ satisfies the constraint if between two ones, there are at most $k-1$ consecutive zeros. The correspondence is as follows. Let $w \in H_{n,k}$ be a string of length $n$. Now generate $x$ from $w$ by writing $x_i = 0$ if $w_i = w_{i+1}$ and $x_i = 1$ if $w_i \neq w_{i+1}$. Notice that $x$ is a string of length $n-1$. Moreover, since $w$ contains a run of maximal length $k$, then there is a maximal string of $k-1$ zeros between two ones, so $x \in X_{k-1} \cap \Sigma^n$. Similarly, we can generate a string $w \in H_k$ from a string $x \in X_{k-1}$ by inverting the process (and deciding that the strings start with 0). Thus, the capacity is the same for both systems.

It is possible to generalize this correspondence to larger alphabets $\Sigma = [r]$ with $r > 2$. In this case, the generalized $(0,k-1)$-RLL system $X_k$ is a system in which between every two symbols from $[r]\{0\}$ there are most $k-1$ consecutive zeros. Given a string $w \in H_{n,k}$, we can generate $x \in X_{k-1}$ by a similar rule: write $x_i = d$ for $w_{i+1} - w_i = d$ with addition (and subtraction) done modulo $r$. 
Example 6: Consider the system \( \mathcal{H}_2 \) and \( \mathcal{H}_3 \) over the binary alphabet \( \Sigma = \{2\}. \) The capacities of these systems are
\[
\text{cap}(\mathcal{H}_2) = \text{cap}(\mathcal{X}_1) = 0.694, \\
\text{cap}(\mathcal{H}_3) = \text{cap}(\mathcal{X}_2) = 0.879.
\]

Since finding an explicit expression for \( \lambda_k \) is impossible for general \( k, \) our next aim is to find some properties of \( h_{k,r}(z) \) and of \( \lambda_k. \) Those properties, listed in the next lemma, will be used later in the description of the right eigenvector \( y. \)

Lemma 5: Let \( \Sigma = [r] \) with \( r \geq 2, \) let \( h_{k,r}(z) \) be defined as in (14), and let \( \lambda_k \) denote the maximal root of \( h_{k,r}(z). \) Then

1) We have \( \lambda_1 = r - 1 \) and for \( k \geq 2, \) \( \lambda_k \in (r - \frac{1}{k}, r). \)

Specifically, \( \lambda_k \geq r - 1. \)

2) For every \( k, \lambda_k \leq \lambda_{k+1}. \)

3) For \( k \geq 2, h_{k,r}(z) \) is increasing for \( z \in [\lambda_k, \infty). \)

Proof: The proof of the properties is straightforward.

1) For \( \Sigma = [r] \) with \( r \geq 2, \) the polynomial \( h_{1,r}(z) = z - (r - 1) \) is clear that \( r - 1 \) is its root. Assume \( k \geq 2 \) and plug in \( z = \frac{r^{k+1}}{k} \) to obtain
\[
h_{k,r} \left( \frac{r^{k+1}}{k} - 1 \right) = 1 - \frac{(kr - 1)^{k+1}}{kr - k - 1}.
\]

Using the binomial formula and the inclusion-exclusion principle, we have
\[
(kr - 1)^{k+1} \geq (kr)^k - k(kr)^{k-1}.
\]

Plugging this to the above equation we obtain
\[
h_{k,r} \left( \frac{r^{k+1}}{k} - 1 \right) \leq \frac{1 - r^{2k-1}(r - 1)}{kr - k - 1} \leq 0,
\]
due to the assumption \( k \geq 2 \) and because \( r \geq 2. \)

Since \( \sum_{i=0}^{k-1} (r - 1)^{z_i} = (r - 1) \frac{z^{k-1} - 1}{z - 1}, \)
\[
h_{k,r}(r) = r^k - (r - 1) \frac{r^{k-1} - 1}{r - 1} = 1 > 0.
\]

Hence, there is a root in \( (r - \frac{1}{k}, r). \) Moreover, for \( k \geq 1 \) we have \( r - (1/k) \geq r - 1. \)

The fact that the maximal root of \( h_{k,r} \) is not greater than \( r \) follows from noticing that for \( z \geq r, h_{k,r}(z) > 0 \) (see (14)).

2) Notice that
\[
h_{k+1,r}(z) = z^{k+1} - (r - 1) (1 + z + \cdots + z^k) = (z - r) z^k + h_{k,r}(z).
\]

For \( 0 < z < r, \) we have \( h_{k+1,r}(z) < h_{k,r}(z). \)

Together with the fact that \( \lambda_{k+1} \) is the maximal root of \( h_{k+1,r}(z), \) we obtain that for all \( z \in [\lambda_{k+1}, r), \)
\( 0 \leq h_{k+1,r}(z) < h_{k,r}(z). \)

Specifically, for \( \lambda_{k+1} \), we have that \( 0 < h_{k,r}(\lambda_{k+1}) \) and that \( h_{k,r}(z) > 0 \) for all \( z \in (\lambda_{k+1}, r). \)

Since \( h_{k,r}(\lambda_k) = 0, \) we obtain that \( \lambda_k < \lambda_{k+1}. \)

3) First notice that for \( k = 1 \) the polynomial \( h_{1,r}(z) \)

is a constant which is non-decreasing. For \( k = 2, \) the statement is clear, so we may assume \( k \geq 3. \)

We first notice that \( h_{k,r}(\lambda_k) = 0, \) the polynomial \( h_{k,r}(z) \)

is continuous for \( z > r - 1 \) and that \( h_{k,r}(r) = 1 > 0. \)

Together with the maximality of \( \lambda_k \) we obtain that
\[
\frac{d}{dz} h_{k,r}(\lambda_k) \geq 0. \text{ In fact, consider the numerator of } h_{k,r}(z) \text{ and notice that}
\]
\[
\frac{d}{dz} z^k (z - r) + 1 = z^k + k z^{k-1} (z - r),
\]

which, in turn, implies that the numerator of \( h_{k,r}(z) \) has a single real minimum point at \( z = \frac{r}{k+1}. \)
From Part 1 together with \( k \geq 2, \) we have \( \lambda_k > r - \frac{1}{k}, \) which implies that the minimum point \( \frac{r_k}{k+1} < \lambda_k. \)
Thus, for \( z > \lambda_k, \) it suffices to show that
\[
\frac{d}{dz} z^k (z - r) + 1 \geq 1,
\]
to conclude that \( h_{k,r}(z) \) is increasing for \( z \geq \lambda_k. \)
To that end, since \( \lambda_k \geq r - \frac{1}{k}, \) we have that for \( z \geq \lambda_k, \)
\[
\frac{d}{dz} z^k (z - r) + 1 = z^k + k z^{k-1} (z - r) \geq z^k - z^{k-1}.
\]

Notice that \( \frac{d}{dz} (z^k - z^{k-1}) = (z - 1) z^{k-1} \log z, \)
which is positive for \( z > 1. \) Therefore, it suffices to show that
\[
z^k - z^{k-1} > 1 \text{ for } z \geq \lambda_k. \text{ Since } z^3 - z^2 = 1 \text{ for } z \approx 1.47 < 1.5 \text{ we obtain the desired result.}\]

Remark 7: The first part of the previous lemma provides a lower bound on the capacity of the \( k \) run length system \( \text{cap}(S_k^n) \geq \log (r - \frac{1}{k+1}). \)

Next, we would like to present a construction for the right Perron eigenvector of \( A_{k+1}^r \) using the right Perron eigenvector of \( A_k^r \) and \( \lambda_{k+1}. \)
To this end, we define an auxiliary polynomial \( g_k(z) \) which is used in the construction.

For \( k = 1 \) we define \( g_k(z) = \frac{1}{z} \) and for \( 2 \leq k \in \mathbb{N}, \) let us denote by \( g_{k,r}(z) \) the polynomial
\[
g_{k,r}(z) = \frac{1}{r - 1} z^{k-1} - \sum_{i=1}^{k-2} z^i = \frac{1}{r - 1} z (h_{k-2}(z)). \quad (15)
\]

When that alphabet \( [r] \) is clear from the context, we will write \( g_k(z) \) instead of \( g_{k,r}(z). \)

We now present the construction of the right Perron eigenvector \( y \) of \( A_{k+1} \) assuming we have the right Perron eigenvector \( x \) of \( A_k \) and \( \lambda_{k+1}. \)

The construction uses the following notation. For a vector \( x \in \mathbb{R}^n, x = (x_0, \ldots, x_{n-1}), \) a set \( B \subseteq [n], \) and a number \( t, \) we denote \( x|_{t-B} \) the vector \( x \)
in which every coordinate that appears in \( B \) is replaced with \( t, \) i.e.,
\[
(x|_{t-B})_i = \begin{cases} 
  t, & \text{if } i \in B, \\
  x_i, & \text{otherwise}. 
\end{cases}
\]

For example, if \( x = (x_0, x_1, x_2, x_3, x_4, x_5, x_6, x_7) \in \mathbb{R}^8, B = \{3, 5\}, \) and \( t = 99, \) then \( x|_{t-B} = (x_0, x_1, x_2, 99, x_4, 99, x_6, x_7). \)

Construction 1: Fix \( 1 \leq k \in \mathbb{N}, \) let \( A_k^n \) denote the adjacency \((r^k \times r^k)\) matrix of the graph \( G[r^n, k] \) that presents \( H_k^n, \) and let \( \lambda_k \) denote its Perron eigenvalue.

Let \( x \in \mathbb{R}^{r^k} \) be a right Perron eigenvector of \( A_k. \) Assume that there are \( f_0, \ldots, f_{r^k-1} : \mathbb{R} \rightarrow \mathbb{R} \) such that
\[
x = (f_0(\lambda_k), \ldots, f_{r^k-1}(\lambda_k)).
\]
Let \( \tilde{x} = (f_0(\lambda_{k+1}), \ldots, f_{r+1}(\lambda_{k+1})) \)
where \( \lambda_{k+1} \) is the Perron eigenvalue of \( A_{k+1}^r \) (or alternatively, the largest root of \( h_{k+1}(\lambda(z)) \)). For every \( i \in [r] \), let \( B_i \) be the set of coordinates \( B_i = \{ j \in [r] \setminus \{ i \} \} \). Let \( y \in \mathbb{R}^{r+1} \) be the vector obtained by
\[
y = (\tilde{x}|_{g_{k+1}(\lambda_{k+1})-B_0}, \tilde{x}|_{g_{k+1}(\lambda_{k+1})-B_1}, \ldots, \tilde{x}|_{g_{k+1}(\lambda_{k+1})-B_{r+1}}).
\]
(16)
We will also use the notation \( T(x) \) to denote the application of the construction on \( x \), so applying the construction \( k \) times will be denoted \( T^k(x) \).

Notice that \( k \) and calculate \( B \) of the construction on \( T \)
We will also use the notation \( A \) for \( A \).

The next example shows the use of Construction 1 and Lemma 6 to find the right Perron eigenvectors of \( A \).

**Example 7:** Consider the systems \( \mathcal{H}_2^2, \mathcal{H}_3^2 \) given in Example 4. It is straightforward to show that a right Perron eigenvector of \( A_2^2 \) is \( x = (1, \lambda_2, \lambda_2, 1) \) \( \lambda_2 \) is the Perron eigenvalue of \( A_2^2 \). Moreover, \( \lambda_2 \) is the smallest root of \( h_{2,2}(z) \), and hence, \( \text{cap}(S_2) = \log \lambda_2 \). A right Perron eigenvector of \( A_2^3 \) is \( y = (1, \lambda_3, \lambda_3, \lambda_3 - \lambda_3, \lambda_3 - \lambda_3, \lambda_3, \lambda_3, 1) \) where \( \lambda_3 \) is the Perron eigenvalue of \( A_2^3 \).

It is possible to obtain \( x \) from the eigenvector of \( A_2^2 \) using Lemma 6. Indeed, from Lemma 6, a right Perron eigenvector for \( A_2^3 \) is \( (1, 1)^T \). Notice that \( x = (f_0(\lambda_1), f_1(\lambda_1)) \) where \( f_0 = f_1 = 1 \). Denote by \( \lambda_2 \) the Perron eigenvalue of \( A_2^2 \) and calculate \( B_0 = \{ 1 \} \), \( B_1 = \{ 0 \} \). Since \( g_{2,2}(\lambda_2) = \lambda_2 \) we obtain that \( x = T((1, 1)) = (1, \lambda_2, \lambda_2, 1) \).

We now use \( x \) to find \( T(x) = y \), the right Perron eigenvector for \( k + 1 = 3 \). First, we find \( f_i, i \in [r^k] = [2^k] \). It is evident that \( f_0 = f_3 = 1 \) and \( f_1 = f_2 = id. \). Thus,
\[
\tilde{x} = (f_0(\lambda_3), f_1(\lambda_3), f_2(\lambda_3), f_3(\lambda_3)) = (1, \lambda_3, \lambda_3, 1),
\]
where \( \lambda_3 \) is the Perron eigenvalue of \( A_2^3 \) or alternatively, the maximal root of
\[
h_{3,2}(z) = z^3 - z^2 - z - 1.
\]
For \( i \in [2] \), we calculate \( B_i \) and obtain that \( B_0 = \{ 1 \} = \{ 1 \} \) and \( B_1 = \{ 0 \} \). Next, we find \( g_3(\lambda_3) = \lambda_3^2 - \lambda_3 \). Overall, we have
\[
\tilde{x}|_{g_3(\lambda_3)-B_0} = \tilde{x}|_{\{ 3 \}} = (1, \lambda_3, \lambda_3, \lambda_3^2 - \lambda_3, 1).
\]
and
\[
\tilde{x}|_{g_3(\lambda_3)-B_0} = \tilde{x}|_{\{ 3 \}} = (1, \lambda_3, \lambda_3, \lambda_3^2 - \lambda_3).
\]
Combining the above we obtain that
\[
y = T(x) = (1, \lambda_3, \lambda_3, \lambda_3^2 - \lambda_3, \lambda_3, 1).
\]

**Remark 8:** Let \( x \) be the vector obtained by \( x = T^{k-1}(1) \).

Construction 1 implies that in the process of constructing \( T(x) \) from \( x \), the entries that are replaced by \( g_{k+1}(\lambda_{k+1}) \) contain 1 before the replacement takes place.

**Remark 9:** Thinking of vertices in \( G([r], k) \) as base-\( r \) numbers, the \( \frac{j-1}{r} \) entry in \( x \) corresponds to the vertex of the form \( j \).

In addition, if \( x \in \mathbb{R}^r \) then \( T(x) \in \mathbb{R}^{r+k+1} \). Thus, when generating \( T(x) \) from \( x \), we first generate \( \tilde{x} \), then we place \( r \) copies of \( \tilde{x} \) one next to the other, and then we change the values in positions that correspond to vertices (in \( G([r], k+1) \)) of the form \( ab^k \) for \( a \neq b \in [r] \). The value in positions that correspond to vertices of the form \( a^k \) remains the same as in the value in \( \tilde{x} \) that appears in position that corresponds to \( a^k \). This, together with Lemma 6, imply that in the process of constructing \( T(x) \) from \( x \), the entries that are replaced by \( g_{k+1}(\lambda_{k+1}) \) contain 1 before the replacement takes place.

Our next goal is to prove that Construction 1 indeed construct the right Perron eigenvector \( y \) of \( A_{r,k+1} \). To this end, we need several lemmas. The first lemma provides some useful properties of the polynomial \( g_r(x) \).

**Lemma 7 (Properties of \( g_r(x) \)):** Fix an alphabet \( \Sigma = [r] \) with \( r \geq 2 \), fix \( k \), let \( \lambda_k \) be the Perron eigenvalue of \( A_r^k \), and let \( g_k(z) = g_r(z) \) be the polynomial defined in (15). Then
1. For \( z \leq r \) we have \( g_k(z) \leq g_{k-1}(z) \leq \ldots \leq g_2(z) \).
2. \( g_k(\lambda_k) \geq 1 \).
3. For \( k \geq 2 \), \( g_k(\lambda_k - 2) = 0 \).
4. For every \( k \geq 1 \) and \( m \geq k \), \( g_k(\lambda_m) \leq g_k(\lambda_{m+1}) \).

**Proof:** The proof of Lemma 7 is straightforward.

1. Notice that for \( k \geq 2 \),
\[
g_{k+1}(z) = \frac{1}{r-1} z^k - \sum_{i=1}^{k-1} z^i = \frac{z-r+1}{r-1} z^{k-1} - \sum_{i=1}^{k-2} z^i = \frac{z-r}{r-1} z^{k-1} + g_k(z).
\]
Therefore, for \( z \leq r \) we have \( g_{k+1}(z) \leq g_k(z) \) which finishes the proof.

2. Notice that
\[
g_k(z) = \frac{1}{r-1} z^{k-1} - \sum_{i=1}^{k-2} z^i - 1 + 1 = \frac{1}{r-1} h_{k-1}(z) + 1 \geq \frac{1}{r-1} h_k(1),
\]
where the last inequality follows from the proof of Lemma 5.2. This, in turn, implies that \( g_k(\lambda_k) \geq 1 \).
3) The fact that \( g_k(\lambda_{k-2}) = 0 \) follows readily from (15) and since \( h_{k-2}(\lambda_{k-2}) = 0 \).
4) The last part is clearly true for \( k = 1 \). Moreover, since 
\[
g_2(z) = \frac{1}{1-z}
\]
is an increasing function of \( z \) and since \( \lambda_m \leq \lambda_{m+1} \), the statement is true for \( k = 2 \) as well. Thus, we may assume \( k \geq 3 \). Differentiate \( g_k(z) \) with respect to \( z \) to obtain
\[
\frac{d}{dz} g_k(z) = \frac{1}{r} \left( h_{k-2}(z) + z \frac{d}{dz} h_{k-2}(z) \right). \quad (17)
\]

Since \( \lambda_{k-2} \) is the maximal root of \( h_{k-2}(z) \), noticing that 
\( h_{k-2}(r) = 1 \) implies that \( h_{k-2}(z) > 0 \) for \( z > \lambda_{k-2} \). Use Lemma 5.2 to obtain 
\( h_{k-2}(\lambda_m) > 0 \). From Lemma 5.3 we obtain that 
\[ z \frac{d}{dz} h_{k-2}(z) \geq 0 \] for \( z \geq \lambda_{k-2} \) which finishes the proof.

The properties of \( g_k,r(z) \) provide some insights on the structure of the Perron eigenvector \( y \) of \( A_1^k \).

**Lemma 8:** Let \( \Sigma = [r] \) and let \( 1 \in \mathbb{R}^r \) be the vector of all ones (the eigenvector of \( A_1^1 \)). For every \( k \in \mathbb{N} \), let 
\( y = T^k(1) \in \mathbb{R}^r \) be the vector obtained after applying the Construction 1 on the vector \( 1 \) of all ones for \( k \) times. Then all the entries of \( y \) are strictly positive.

**Proof:** In order to prove the statement for \( k \geq 1 \), we notice that it is sufficient to show that \( g_m(\lambda_k) > 0 \) for all \( m \leq k \). Indeed, from the construction, all the arguments that appear in \( y = T(x) \) have the form \( g_m(\lambda_k) \) for \( m \leq k \). Now the lemma follows immediately from parts 2 and 4 of Lemma 7.

**Lemma 9:** Fix \( \Sigma = [r] \). For every \( k \in \mathbb{N} \), let 
\( y = T^k(1) \in \mathbb{R}^r \) be the vector obtained by applying Construction 1 (in particular, (16)) \( k \) times on \( 1 \). Then \( y \) contains ones in positions 
\[ \left\{ j \frac{k+1}{r} : j \in [r] \right\}. \]

**Remark 10:** Lemma 9 suggests that if we enumerate the positions of the vector \( y \) as base-\( r \) numbers then \( y \) contains ones in positions of the form 
\[ \frac{j}{r} \ldots \frac{j}{r} \] for \( j \in [r] \).

**Proof:** We prove the lemma using induction. For \( k = 1 \) Lemma 6 states that the all ones vector contain ones in all the positions, i.e., in positions \([r]\). Assume that the lemma is correct for \( k - 1 \) and we will show it is correct for \( k \). Let \( x \in \mathbb{R}^{r^{k-1}} \) be the vector obtained after applying \( T \) for \( (k - 1) \) times. From the induction hypothesis, \( x \) contains ones only in positions 
\[ \left\{ j \frac{k}{r} : j \in [r] \right\}. \]
Thus, \( x \) contain ones at the exact same positions. From the definition of \( B_i \) we obtain that 
\( B_i x \) contains ones only in position 
\[ i \frac{k-1}{r} := \left\{ i \frac{k-1}{r} : i \in [r] \right\}. \]
This implies that \( y \) contains ones in positions 
\[ \left\{ i \frac{k-1}{r} + i \frac{k-1}{r} : i \in [r] \right\} = \left\{ i \frac{k-1}{r} = i : i \in [r] \right\}, \]
and the proof follows.

The next insight, stated in the following lemma, provides the explicit values of \( y \) in terms of the polynomials \( g_i \) and \( \lambda_i \). This explicit formulation is essential for the proof of Theorem 2.

**Lemma 10:** Fix alphabet \( \Sigma = [r] \) and let \( 1 \) be the all ones vector (the right Perron eigenvector of \( A_1^1 \)). For \( k \geq 1 \), the vector \( y = T^{k-1}(1) \in \mathbb{R}^r \) has the following form. Let 
\( i = (i_{k-1} \ldots i_0) \in [r^k] \) be a number considered in its base-
\( r \) representation and let \( i_l \in [k] \) be the largest number for which 
\( i_0 = i_1 = \ldots = i_{l-1} = i_l \), i.e., \( l \) denotes the number of repeated least significant symbols in the base-\( r \) representation of \( i \). Then 
\[ y_i = \begin{cases} 1 & \text{if } l_i = k \\ g_{l_i+1}(\lambda_k) & \text{if } l_i < k, \end{cases} \]
where \( \lambda_k \) is the Perron eigenvalue of \( A_1^k \).

**Example 8:** As seen in Example 7, for \( k = 3 \), a right Perron eigenvector of \( A_3^2 \) is 
\[ y = T^2(1) = (1, \lambda_3, \lambda_3, \lambda_3^2 - \lambda_3, \lambda_3^2 - \lambda_3, \lambda_3, \lambda_3, 1) \] where \( \lambda_3 \) is the Perron eigenvalue of \( A_3^2 \). Indeed, in positions \( i = 0, 7 \) (in binary representation 000, 111), there is \( 1 \). In positions \( i = 1, 2, 4, 6 \) (positions 001, 010, 101, 110), the value of \( y_i = 1 \) and so \( y_i = g_2(\lambda_k) = \lambda_3 \), and in positions \( i = 3, 5 \) (positions 011, 100), we have \( l_i = 2 \) and so \( y_i = g_3(\lambda_k) = \lambda_3^2 - \lambda_3 \). See Fig. 5 for the process of constructing the eigenvector of \( A_1^2 \) from the eigenvector of \( A_3^3 \).

**Proof:** [Proof of Lemma 10] The proof follows by induction. For the base of induction we note that \( T(1) \) contains \( 1 \) in positions \( 00, 11, \ldots, (r-1)(r-1) \) and contains \( \frac{r^k}{r} \) in the rest of the positions, where \( \lambda_3 \) is the Perron eigenvalue of \( A_3^2 \). Now assume this is true for \( k - 1 \) and we show it holds for \( k \). Let 
\[ x = T^{k-1}(1) \] and 
\[ y = T(x) \] By Construction 1, the positions in \( y \) that contain \( g_{k+1}(\lambda_k+1) \) are positions such that written in their base-\( r \) representation have the form \( ab^k \) for \( a \neq b \in [r] \). According to Lemma 9, positions of the form \( ab^{k+1} \) contain \( 1 \). Position \( i = (i_{k-1} \ldots i_0) \) in \( x \) with \( l_i < k \) correspond to positions \( ai = (ai_{k-1} \ldots i_0) \) with \( a \in [r] \) in \( y \), and will remain with the same value \( l_{ai} = l_i \).

We are now ready to prove that \( T^{k-1}(1) \) is a right Perron eigenvector of \( A_k(z) \). This is stated formally in the next claim.

**Claim 1:** For an alphabet \([r]\) and \( k \geq 1 \). Let \( 1 \) be the all one vector comprises of \( r \) ones (the right Perron eigenvector of \( A_1^1 \)), then for \( k \geq 1 \), \( y = T^{k-1}(1) \) is a right Perron eigenvector of \( A_k \).

**Proof:** We first notice that we only need to show that \( y \) is an eigenvector. Since Lemma 8 together with Perron-Frobenius theorem and Corollary 1 imply that if \( T^{k-1}(1) \) is an eigenvector, it is a Perron eigenvector. Thus, the proof follows from a straightforward calculation of \( A_k y \) where \( y = T^{k-1}(1) \) and \( 1 \) is the vector comprises of \( r \) ones. This can be done using Lemma 10 and Lemma 2.

Let 
\[ i = (i_{k-1} \ldots i_0) \in [r^k] \] and consider \( (A_k y)_i \). Let us denote by \( l_i \) the number of repeated least significant symbol in \( i \). Lemma 10 implies that \( y_i = g_{l_i+1}(\lambda_k) \).

1) Case 1: \( l_i \leq k - 2 \). Using Lemma 2 we obtain that 
\[ (A_k y)_i = \sum_{j=0}^{r-1} g_{i_l-2 \ldots i_0 j}. \]
Since \( l_i \leq k - 2 \) the sum \( (A_k y)_i \) comprises of \( r - 1 \) values \( \lambda_k^l \) and a single value \( g_{l_i+2}(\lambda_k) \).

\[ g_{l_i+2}(\lambda_k) + \lambda_k = \frac{1}{r} \lambda_k^{l_i+1} - \sum_{i=1}^{l_i} \lambda_k^i + \lambda_k = \lambda_k(g_{l_i+1}(\lambda_k)). \]
we have
\[(A_k y)_i = \lambda_k g_{l_i+1}(\lambda_k).\]

2) Case 2: \(l_i = k - 1\). Using Lemma 2 we obtain that
\[(A_k y)_i = \sum_{j=0}^{r-1} y_{i-k-1} \ldots y_{i-j}.\]
In this case we have that the sum \((A_k y)_i\) comprises of \(r - 1\) values \(\lambda_k \ldots \lambda_1\) and a single value \(y_1\). Notice that
\[\lambda_k g_k(\lambda_k) = \frac{1}{r - 1} \lambda_k^r - \sum_{i=2}^{k-1} \lambda_k^i = h_k(\lambda_k) + \lambda_k + 1 = \lambda_k + 1.\]
Therefore, in this case as well,
\[(A_k y)_i = \lambda_k + 1 = \lambda_k g_k(\lambda_k).\]

3) Case 3: \(l_i = k\). Let \(i = (a \ldots a)\) for \(a \in [r]\). Using Lemma 2 we obtain that
\[(A_k y)_i = \sum_{j \in [r] \setminus \{a\}} y_{a \ldots a j}.\]
In this case we have that the sum \((A_k y)_i\) comprises of \(r - 1\) values \(\lambda_k \ldots \lambda_1\) and \(y_i = 1\). Thus, we obtain
\[(A_k y)_i = \lambda_k.\]
In all three cases above we obtain that \((A_k y)_i = \lambda_k y_i\), thus \(y_i\) is an eigenvector which finishes the proof.

With the right Perron eigenvector in hand, we may estimate the transition probabilities \(Q_{i,j}\) of the Markov measure \(\mu\), or in other words, to prove Theorem 2. this will follow readily from the following claim that uses the structure of \(y_i\).

\[\text{Claim 2: Let } H_k \text{ be the } k \text{-run length constrained system over the alphabet } \Sigma = [r] \text{ and let } \mu \text{ be the Markov measure that corresponds to } H_k \text{ with } \pi \text{ its stationary distribution and } Q \text{ its transition probabilities matrix. For } i \in [r^k] \text{ we denote by } l_i \text{ the number of repeated least significant symbol in the base-} r \text{ representation of } i. \text{ Then for every } i, j, t \in [r^k], \text{ if } l_j < l_t \text{ and there are edges } (i, j), (t, t), \text{ then } Q_{l, j} > Q_{l, t}. \text{ Moreover, if } l_j = l_t \text{ then } Q_{l, j} = Q_{l, t}.\]
\[\text{Proof: The proof follows immediately since } Q_{i,j} = \frac{y_i y_j}{y_i + y_j} \text{ where } y \text{ is a right Perron eigenvector of } A_k, \text{ together with Lemma 10 and Lemma 7.1.}\]
Finally, we prove Theorem 2. \[\text{Proof: [Proof of Theorem 2]}\]
The proof follows from Claim 2 since the claim suggests that the probability of repeating the last symbol is the smallest. In addition, since \(Q_{i,j} = \frac{y_i y_j}{y_i + y_j}\), where \(y\) is the right Perron eigenvector, and since by Lemma 10, \(y_i, y_j\) depend only on the numbers \(l_i, l_j\) of repeated least significant symbols in the base-

IV. CONCLUSION AND OUTLOOK

In this paper, we studied the single batch settings, in which information strings appear in the same set and are synthesized with respect to a reference \(R\). We showed that the optimal reference sequence is \(R^* = 012 \ldots (r - 1)\). Throughout the analysis, we provided an explicit formula for the right Perron eigenvector of the adjacency matrix corresponds to the constrained system \(S_k\). A complete analysis of the system \(S_k\) will be achieved if an explicit formula for the left Perron eigenvector will be found. This may also provide some concrete bounds on the cost of synthesis \(\text{cost}(S)\). Another interesting direction for future research is the inclusion of additional constraints such as the balance GC content in DNA strings. Notice that while limiting the lengths of runs is a local constraint, balancing the number of GC symbols with the number of TA symbols is a global constraints. As such,
we may require the use a new set of tools. We leave these
endevours for a future research.

REFERENCES

[1] L. Anavy, I. Vaknin, O. Atar, R. Amit, and Z. Yakhini, “Data storage in DNA with fewer synthesis cycles using composite DNA letters,” Nature Biotechnol., vol. 37, no. 10, pp. 1299–1306, 2019.
[2] T. Batu, S. Kannan, S. Khanna, and A. McGregor, “Reconstructing strings from random traces,” in Proc. 15th Annu. ACM-SIAM Symp. Discrete Algorithms, 2004, pp. 910–918.
[3] J. Bornholt, R. Lopez, D. M. Carman, L. Ceze, G. Seelig, and K. Strauss, “A DNA-based archival storage system,” ACM SIGPLAN Notices, vol. 51, no. 4, pp. 637–649, May 2016.
[4] B. Bukh and C. Cox, “Periodic words, common subsequences and frogs,” 2019, arXiv:1912.03510.
[5] S. Chandak et al., “Overcoming high nanopore basecaller error rates for DNA storage via basecaller-decoder integration and convolutional codes,” in Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP), May 2020, pp. 8822–8826.
[6] V. Chepoi and D. Sankoff, “Longest common subsequences of two random sequences,” Adv. Appl. Probab., vol. 7, no. 2, p. 253, Jun. 1975.
[7] C. Colbourn, A. Ling, and M. Tompa, “Construction of optimal quality control for oligo arrays,” Bioinformatics, vol. 18, pp. 529–535, Apr. 2002.
[8] V. Cancedda and M. Paterson, “Upper bounds for the expected length of a longest common subsequence of two binary sequences,” Random Struct. Algorithms, vol. 1, no. 4, pp. 449–458, Jul. 1995.
[9] N. Goldman et al., “Towards practical, high-capacity, low-maintenance information storage in synthesized DNA,” Nature, vol. 494, no. 7435, pp. 77–80, Feb. 2013.
[10] S. Hannenhalli, E. Hubell, R. Lipshutz, and P. A. Pevzner, “Combinatorial algorithms for design of DNA arrays,” in Proc. Adv. Biochem. Eng./Biotechnol., vol. 77, 2002, pp. 1–19.
[11] N. Holden, R. Pemantle, and Y. Peres, “Subpolynomial trace reconstruction for random strings and arbitrary deletion probabilities,” in Proc. Conf. Learn. Theory (COLT), Stockholm, Sweden, Jul. 2018, pp. 1799–1840.
[12] T. Holenstein, M. Mitzenmacher, R. Panigrahy, and U. Wieder, “Trace reconstruction with constant deletion probability and related results,” in Proc. 9th Annu. ACM-SIAM Symp. Discrete Algorithms, 2008, pp. 389–398.
[13] C. Houdré and H. Matzinger, “Closeness to the diagonal for longest common subsequences in random words,” Electron. Commun. Probab., vol. 21, pp. 1–19, Jan. 2016, doi: 10.1214/16-ECP4029.
[14] E. Hubbell and P. Pevzner, “Fidelity probes for DNA arrays,” in Proc. Int. Conf. Intell. Syst. Mol. Biol., 1999, pp. 113–117.
[15] K. A. S. Immink and K. Cai, “Properties and constructions of constrained codes for DNA-based data storage,” IEEE Access, vol. 8, 2020, pp. 49013–49031, 2020.
[16] S. Jain, F. Farnoud, M. Schwartz, and J. Bruck, “Coding for optimized writing rate in DNA storage,” in Proc. IEEE Int. Symp. Inf. Theory (ISIT), Jun. 2020, pp. 2777–2782.
[17] G. Navarro, “A guided tour to approximate string matching,” ACM Comput. Surv., vol. 33, no. 1, pp. 31–88, Mar. 2001.
[18] T. T. Nguyen, K. Cai, K. A. S. Immink, and H. M. Kiah, “Capacity-approaching constrained codes with error correction for DNA-based data storage,” IEEE Trans. Inf. Theory, vol. 67, no. 8, pp. 5602–5613, Aug. 2021.
[19] K. Ning and H. W. Leong, “The distribution and deposition algorithm for multiple oligo nucleotide arrays,” Genome Informat., vol. 17, no. 2, pp. 89–99, 2006.
[20] K. Ning and H. W. Leong, “The multiple sequence sets problem: Problem and heuristic algorithms,” J. Combinat. Optim., vol. 22, no. 4, pp. 778–796, Nov. 2011.
[21] L. Organick et al., “Random access in large-scale DNA data storage,” Nature Biotechnol., vol. 36, no. 3, pp. 242–248, Mar. 2018.
[22] Y. Peres and A. Zhai, “Average-case reconstruction for the deletion channel: Subpolynomially many traces suffice,” in Proc. IEEE 58th Annu. Symp. Found. Comput. Sci. (FOCS), Oct. 2017, pp. 228–239.
[23] S. Rahmann, “The shortest common supersequence problem in a microarray production setting,” Bioinformatics, vol. 19, pp. 156–161, Sep. 2003.
[24] S. Rahmann, “Subsequence combinatorics and applications to microarray production, DNA sequencing and chaining algorithms,” in Proc. CPM, 2006, pp. 153–164.
[25] C. Rasmishch et al., “Clustering billions of reads for DNA data storage,” in Proc. Adv. Neural Inf. Process. Syst., vol. 30, 2017, pp. 1–12.
[26] M. G. Ross et al., “Characterizing and measuring bias in sequence data,” Nature Meth., vol. 5, no. 4, pp. 379–385, Apr. 2008.
[27] S. Vinylas, V. Kambakota, and A. Bhattacharya, “A review of algorithms for border length minimization problem,” IETE Tech. Rev., vol. 31, pp. 369–382, Sep. 2014.
[28] K. Viswanathan and R. Swaminathan, “Improved string reconstruction over insertion-deletion channels,” in Proc. 19th Annu. ACM-SIAM Symp. Discrete Algorithms, 2008, pp. 399–408.
[29] P. Walters, An Introduction to Ergodic Theory. Berlin, Germany: Springer, 1982.