Coding for Efficient DNA Synthesis

Andreas Lenz[*], **Yi Liu**[‡], **Cyrus Rashtchian**[†], **Paul H. Siegel**[‡], **Antonia Wachter-Zeh**[*], and **Eitan Yaakobi**[§]
[*]Institute for Communications Engineering, Technical University of Munich, Germany
[†]Computer Science and Engineering Department and the Qualcomm Institute, University of California, San Diego
[‡]Department of Electrical and Computer Engineering, CMRR, University of California, San Diego
[§]Computer Science Department, Technion – Israel Institute of Technology, Haifa, Israel
andreas.lenz@mytum.de, yil333@eng.ucsd.edu, crashtchian@eng.ucsd.edu, psiegel@ucsd.edu,
antonia.wachter-zeh@tum.de, yaakobi@cs.technion.ac.il

***Abstract*—For DNA data storage to become a feasible technology, all aspects of the encoding and decoding pipeline must be optimized. Writing the data into DNA, which is known as DNA synthesis, is currently the most costly part of existing storage systems. As a step toward more efficient synthesis, we study the design of codes that minimize the time and number of required materials needed to produce the DNA strands. We consider a popular synthesis process that builds many strands in parallel in a step-by-step fashion using a fixed supersequence $\boldsymbol{S}$. The machine iterates through $\boldsymbol{S}$ one nucleotide at a time, and in each cycle, it adds the next nucleotide to a subset of the strands. The synthesis time is determined by the length of $\boldsymbol{S}$. We show that by introducing redundancy to the synthesized strands, we can significantly decrease the number of synthesis cycles. We derive the maximum amount of information per synthesis cycle assuming $\boldsymbol{S}$ is an arbitrary periodic sequence. To prove our results, we exhibit new connections to cost-constrained codes.**

I. Introduction

In the past decade, DNA has emerged as a potentially viable storage technology [1], [2]. Compared to traditional storage media, DNA offers the possibility of significantly improved information density and durability [3]–[5]. While much recent work has optimized many aspects of the DNA data storage pipeline [6]–[9], we identify and address the goal of optimizing the synthesis process. Typically information is stored by first preprocessing the digital data and then encoding it in physical DNA molecules using a synthesis machine. Most experiments on DNA data storage use the same type of synthesis process [10]–[12]. The machine creates a large number of DNA strands in parallel, where each strand is grown by one nucleotide at a time. To append nucleotides to the strands, the synthesis machine follows a fixed supersequence of possible nucleotides. As the machine iterates through this supersequence, the next nucleotide is added to a select subset of the DNA strands. This process continues until the machine reaches the end of the supersequence. In particular, each synthesized DNA strand must be a subsequence of the machine's supersequence. Figure 1 depicts the synthesis process of multiple strands from a fixed supersequence. To increase the throughput of the DNA synthesis process, we consider the problem of encoding the DNA strands with the goal of minimizing the total number of cycles used by the synthesis machine. Not only will this decrease the synthesis time, but it will also decrease the monetary cost because each cycle expends chemicals and reagents [10]–[13].

A. Lenz acknowledges support from the German-American Fulbright Commission for funding the visit to UCSD. E. Yaakobi acknowledges support from the Center for Memory and Recording Research at UCSD. This work is also funded by the European Research Council under the EU's Horizon 2020 research and innovation programme (grant No. 801434), by NSF grant CCF-BSF-1619053 and by the United States-Israel BSF grant 2018048.

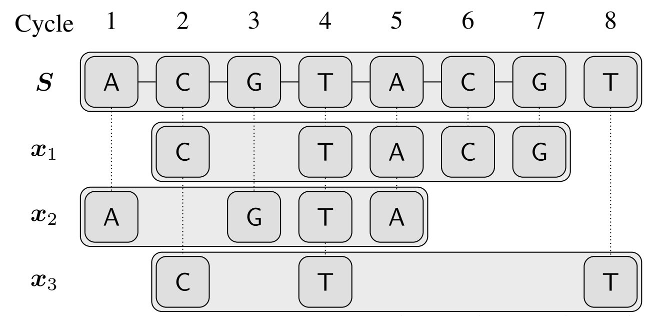

Fig. 1: Synthesis of three strands $\boldsymbol{x}_1 = (\mathsf{CTACG}), \boldsymbol{x}_2 = (\mathsf{AGTA})$, and $\boldsymbol{x}_3 = (\mathsf{CTT})$ using the synthesis sequence $\boldsymbol{S} = (\mathsf{ACGTACGT})$. The strand $\boldsymbol{x}_1$ is synthesized by attaching the nucleotides in cycles $2, 4, 5, 6, 7$, $\boldsymbol{x}_2$ is synthesized in cycles $1, 3, 4, 5$, and similarly $\boldsymbol{x}_3$ is synthesized in cycles $2, 4, 8$. Henceforth, the synthesis time of $\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3$ is given by $t_{\boldsymbol{S}}(\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3) = 8$.

A theoretical model of the process involves several variables. For simplicity, assume that the number of strands k is fixed, based on the size of the synthesis machine (in typical systems, $k \approx 10^6$). Then, the length n of each strand may be fixed or variable (usually n is between 100 and 1000). When n is fixed, a naive solution in which information is encoded into DNA strands using the rule $(00) \to \mathsf{A}$, $(01) \to \mathsf{C}$, $(10) \to \mathsf{G}$, $(11) \to \mathsf{T}$ uses a supersequence of length $4n$ that repeats the substring ACGT exactly n times. This scheme achieves an *information rate* of 0.5 bits/cycle, since it requires 4 cycles to synthesize one nucleotide and one nucleotide contains two bits of information. We show that by introducing redundancy to the synthesized strands in the form of a *synthesis code*, the synthesis time can be significantly decreased, respectively the information rate can be increased, compared to uncoded systems. As one of our results, we present a simple encoding scheme with a single redundancy symbol that achieves an information rate of 0.8 bits/cycle. Then, we show that this can be further optimized to an optimum of 0.95 bits/cycle, while increasing the number of redundancy symbols, using a more sophisticated scheme based on constrained codes. In other words, we encode the strands to achieve nearly twice the throughput and half the cost compared to the uncoded solution.

More precisely, we aim to determine the maximum number of bits that can be encoded into a set of k strands, each of length n, while given an overall budget T on the number of synthesis cycles (i.e., the supersequence has length T). In particular we derive the maximum possible amount of information per synthesis time, given the usage of an arbitrary periodic synthesis sequence. To prove our results, we exhibit connections between the number of subsequences of a sequence and cost-constrained systems.

II. PRELIMINARIES

Throughout the paper, we use the (shifted) modulo operator $(a \bmod p) \in \{1, \dots, p\}$. Let $\boldsymbol{x} \in \Sigma^n$ be a strand, which is a string of length n over the alphabet Σ of size $|\Sigma| = q$. The length of a string $\boldsymbol{x} \in \Sigma^n$ is denoted by $|\boldsymbol{x}| = n$ and the length-m prefix of $\boldsymbol{x}$ is denoted by $\boldsymbol{x}_{1:m}$. The *radius-t deletion ball* obtained after exactly t deletions in $\boldsymbol{x}$ is denoted by $\mathsf{D}(\boldsymbol{x}, t)$. The *global deletion ball of* $\boldsymbol{x}$ is defined by $\mathsf{D}^*(\boldsymbol{x}) = \bigcup_{t=1}^{|\boldsymbol{x}|} \mathsf{D}(\boldsymbol{x}, t)$. A strand $\boldsymbol{S} \in \Sigma^*$ is called a *supersequence* of $\boldsymbol{x} \in \Sigma^*$, if $\boldsymbol{x}$ can be obtained by deleting symbols from $\boldsymbol{S}$, i.e., $\boldsymbol{x} \in \mathsf{D}^*(\boldsymbol{S})$. In reverse, $\boldsymbol{s} \in \Sigma^*$ is called a *subsequence* of $\boldsymbol{x} \in \Sigma^*$, if it can be obtained from $\boldsymbol{x}$ via deletions, i.e., $\boldsymbol{s} \in \mathsf{D}^*(\boldsymbol{x})$. Further, for any k strands $\boldsymbol{x}_1, \dots, \boldsymbol{x}_k \in \Sigma^*$ we define $\boldsymbol{S} \triangleq \mathsf{SCS}(\boldsymbol{x}_1, \dots, \boldsymbol{x}_k)$ to be any *shortest common supersequence* (*SCS*) of $\boldsymbol{x}_1, \dots, \boldsymbol{x}_k$. Note that while there may be more than one SCS, the length is unique and in the following, the particular choice of the SCS will not be of importance.

A. Problem Formulation

Consider a system, where digital data shall be encoded and synthesized into k DNA strands $\boldsymbol{x}_1, \dots, \boldsymbol{x}_k \in \Sigma^*$ *in parallel*. These strands can be of equal or different lengths. The synthesis is performed by choosing a synthesis sequence of nucleotides $\boldsymbol{S} = (S_1, S_2, \dots) \in \Sigma^*$ and in each cycle $i = 1, \dots, |\boldsymbol{S}|$, for each DNA strand $\boldsymbol{x}_j$, it is possible to either attach the symbol S_i to the strand $\boldsymbol{x}_j$ or to perform no action. The sequence $\boldsymbol{S}$ must be chosen such that it is possible to synthesize each strand with this procedure. If a strand $\boldsymbol{x}$ can be synthesized by the synthesis sequence $\boldsymbol{S}$, then this implies that $\boldsymbol{x}$ is a subsequence of $\boldsymbol{S}$ and the other direction holds as well, since by definition every subsequence of $\boldsymbol{S}$ can be synthesized by $\boldsymbol{S}$. Hence, the following lemma follows directly.

Lemma 1. *The sequences $\boldsymbol{x}_1, \dots, \boldsymbol{x}_k$ can be synthesized using the synthesis sequence $\boldsymbol{S}$ if and only if $\boldsymbol{S}$ is a common supersequence of $\boldsymbol{x}_1, \dots, \boldsymbol{x}_k$.*

The key figure of merit of our analysis is the *synthesis time* of a set of sequences $\boldsymbol{x}_1, \dots, \boldsymbol{x}_k$, which is defined as follows.

Definition 1. *The **synthesis time** $t_{\boldsymbol{S}}(\boldsymbol{x}_1, \dots, \boldsymbol{x}_k)$ of a set of strands $\boldsymbol{x}_1, \dots, \boldsymbol{x}_k$ with the synthesis sequence $\boldsymbol{S}$ is defined to be the smallest number of synthesis cycles that are required to synthesize the strands $\boldsymbol{x}_1, \dots, \boldsymbol{x}_k$ with $\boldsymbol{S}$, i.e.,*

$$t_{\boldsymbol{S}}(\boldsymbol{x}_1, \dots, \boldsymbol{x}_k) = \min_{t \in \mathbb{N}} t \quad \text{s.t. } \{\boldsymbol{x}_1, \dots, \boldsymbol{x}_k\} \subseteq \mathsf{D}^*(\boldsymbol{S}_{1:t}).$$

The aim of this analysis is to design codes $\mathcal{C}_k^n \subseteq (\Sigma^n)^k \triangleq \{(\boldsymbol{x}_1, \dots, \boldsymbol{x}_k) : \boldsymbol{x}_i \in \Sigma^n\}$ over DNA strands, such that the tuples of strands inside this code have a small synthesis time. That is, we allow only to synthesize tuples of sequences $\boldsymbol{x}_1, \dots, \boldsymbol{x}_k$ that are contained in $\mathcal{C}_k^n$. Designing $\mathcal{C}_k^n$ thus allows to control the synthesis time. Hereby, it is possible to distinguish between two different setups. First, there is the case, where $\boldsymbol{S}$ is fixed and does not depend on the strands $\boldsymbol{x}_1, \dots, \boldsymbol{x}_k$ and second, the case, where the synthesis sequence is variable and may be a function of the strands $\boldsymbol{x}_1, \dots, \boldsymbol{x}_k$. In the latter case, it is natural to use $\boldsymbol{S}(\boldsymbol{x}_1, \dots, \boldsymbol{x}_k) = \mathsf{SCS}(\boldsymbol{x}_1, \dots, \boldsymbol{x}_k)$ as this sequence minimizes the synthesis time for these strands. In this paper we focus however on the former case, where the synthesis sequence is fixed. Note that in this case, by Lemma 1, each strand $\boldsymbol{x}_i$ individually must be a subsequence of the fixed synthesis sequence $\boldsymbol{S}$ and thus the number of sequences k is irrelevant for the code design. We therefore restrict $k = 1$ in the following. With the above definition of the synthesis time, we can directly identify the main trade-off in the code design. On the one hand, it is desirable to have a small synthesis time, i.e., strongly restrict the strands $\boldsymbol{x}$ to be a subsequence of a possibly short prefix of $\boldsymbol{S}$. On the other hand, we are striving for a large information content $\log |\mathcal{C}_k^n|$. These contradictory goals naturally motivate the statement of the following optimization.

$$N^n(\boldsymbol{S}, T) \triangleq \max_{\mathcal{C}^n \subseteq \Sigma^n} |\mathcal{C}^n| \quad \text{s.t.} \quad t_{\boldsymbol{S}}(\boldsymbol{x}) \leq T \ \forall\ \boldsymbol{x} \in \mathcal{C}^n.$$

In other words, given a maximum synthesis time T, we would like to characterize the maximum amount of information that we can synthesize in this time. Similarly, for codes that contain strands of any length, we replace n by $*$ in the above definition. Given the above maximization problem, we are now in the position to present the figure of merit discussed in this paper, i.e., the (asymptotic) maximum *information rate* measured by number of bits per synthesis cycle. Given a semi-infinite synthesis sequence[1] $\boldsymbol{S}$ and $0 \leq \alpha \leq 1$, we define

$$\mathcal{R}(\boldsymbol{S}, \alpha) = \limsup_{T \to \infty} \frac{\log(N^{\lfloor \alpha T \rfloor}(\boldsymbol{S}_{1:T}, T))}{T},$$

and similarly

$$\mathcal{R}^*(\boldsymbol{S}) = \limsup_{T \to \infty} \frac{\log(N^*(\boldsymbol{S}_{1:T}, T))}{T}.$$

Interestingly, the value of $N^n(\boldsymbol{S}, T)$ and $N^*(\boldsymbol{S}, T)$ (and thus also of $\mathcal{R}(\boldsymbol{S}, \alpha)$ and $\mathcal{R}^*(\boldsymbol{S})$) can directly be related to the number of subsequences, i.e. the deletion ball, of the synthesis sequence $\boldsymbol{S}$ as stated in the next lemma.

Lemma 2. *For all $\boldsymbol{S}$ and T such that $|\boldsymbol{S}| = T$ it holds that*

$$N^n(\boldsymbol{S}, T) = |\mathsf{D}(\boldsymbol{S}, T - n)|, N^*(\boldsymbol{S}, T) = |\mathsf{D}^*(\boldsymbol{S})|.$$

Proof. If $\boldsymbol{x}$ can be synthesized using the synthesis sequence $\boldsymbol{S}$, then $\boldsymbol{x}$ is a subsequence of $\boldsymbol{S}$ and thus $\boldsymbol{x} \in \mathsf{D}(\boldsymbol{S}, T - n)$ since the length of $\boldsymbol{x}$ is n. On the other hand, every $\boldsymbol{x} \in \mathsf{D}(\boldsymbol{S}, T - n)$ can be synthesized using $\boldsymbol{S}$ since it is its subsequence and thus $N^n(\boldsymbol{S}, T) = |\mathsf{D}(\boldsymbol{S}, T - n)|$. The proof for $N^*(\boldsymbol{S}, T)$ follows by repeating the last argument for all lengths. □

III. INFORMATION-RATE OPTIMAL SYNTHESIS SEQUENCE

Before we present how to find the information rates $\mathcal{R}(\boldsymbol{S}, \alpha)$ and $\mathcal{R}^*(\boldsymbol{S})$ for a general synthesis sequence $\boldsymbol{S}$, we find the sequence that maximizes the information rates $\mathcal{R}(\boldsymbol{S}, \alpha)$ and $\mathcal{R}^*(\boldsymbol{S})$, and compute the resulting information rates using combinatorial tools. That is, we are seeking to solve the problems $\max_{\boldsymbol{S} \in \Sigma^*} \mathcal{R}(\boldsymbol{S}, \alpha)$, and $\max_{\boldsymbol{S} \in \Sigma^*} \mathcal{R}^*(\boldsymbol{S})$. Clearly, the maximizers of $\max_{\boldsymbol{S} \in \Sigma^T} \{N^n(\boldsymbol{S}, T - \lfloor \alpha n \rfloor)\}$, and $\max_{\boldsymbol{S} \in \Sigma^*} \{N^*(\boldsymbol{S}, T)\}$ provide solutions to the earlier optimization problems. Together with Lemma 2, the maximizer of both problems is the sequence that maximizes the number of subsequences and thus we obtain

$$\operatorname*{arg\,max}_{\boldsymbol{S} \in \Sigma^*} \mathcal{R}(\boldsymbol{S}, \alpha) = \operatorname*{arg\,max}_{\boldsymbol{S} \in \Sigma^*} \mathcal{R}^*(\boldsymbol{S}) = \mathbf{A}_q,$$

where $\mathbf{A}_q$ is the *alternating sequence* [14] that cyclically repeats all symbols in Σ in ascending order. For example, for

[1]A semi-infinite sequence is a sequence $\boldsymbol{S} = (S_i : i \in \mathbb{N})$ that starts at symbol S_1, but does not have an end. This semi-infiniteness is necessary to formally define the limit value in the definitions of $\mathcal{R}(\boldsymbol{S}, \alpha)$ and $\mathcal{R}^*(\boldsymbol{S})$.

$\Sigma = \{0,1\}$, the alternating sequence is $\mathbf{A}_2 = (0101\ldots)$. The number of subsequences of the length-n alternating sequence $\mathbf{A}_q^n \stackrel{\text{def}}{=} [\mathbf{A}_q]_{1:n}$ is given by the recursive formula [15]

$$\mathsf{D}_q(n,t) \triangleq |\mathsf{D}(\mathbf{A}_q^n, t)| = \sum_{i=0}^{t} \binom{n-t}{i} \mathsf{D}_{q-1}(t, t-i).$$

In particular, $\mathsf{D}_2(n,t) = \sum_{i=0}^{t} \binom{n-t}{i}$ and $\mathsf{D}_3(n,t) = \sum_{i=0}^{t} \binom{n-t}{i} \sum_{j=0}^{t-i} \binom{i}{j}$. Explicit values for the resulting information rates are given in the next theorem.

Theorem 3. *For all $0 \le \alpha \le 1$, and $q = 2$, it holds that*

$$\max_{\boldsymbol{S}\in\Sigma^*} \mathcal{R}(\boldsymbol{S},\alpha) = \mathcal{R}(\mathbf{A}_2,\alpha) = \begin{cases} \alpha h(\alpha^{-1}-1), & \text{if } \alpha \ge \frac{2}{3} \\ \alpha, & \text{otherwise} \end{cases},$$

where $h(x)$ is the binary entropy function. Further, for any q,

$$\max_{\boldsymbol{S}\in\Sigma^*} \mathcal{R}^*(\boldsymbol{S}) = \mathcal{R}^*(\mathbf{A}_q) = -\log z_q,$$

where z_q is the largest root of the polynomial $\sum_{i=1}^{q} z^i - 1$.

Proof. The first part of the theorem directly follows from an asymptotic analysis of the quantity $\mathsf{D}_2(T, T - \lfloor \alpha T \rfloor)$. The second part is proven as follows. For a fixed number t of deletions, the number of subsequences of $\mathbf{A}_q^T$ is given by [15]

$$|\mathsf{D}(\mathbf{A}_q^T, t)| = [z^T] \left(\sum_{j=1}^{q} z^j \right)^{T-t} \frac{1}{1-z},$$

where $[z^T]$ denotes the operation of extracting the coefficient of z^T. Therefore, we obtain

$$|\mathsf{D}^*(\mathbf{A}_q^T)| = \sum_{t=0}^{T} |\mathsf{D}(\mathbf{A}_q^T, t)| = [z^T] \frac{1}{1-z} \sum_{t=0}^{T} \left(\sum_{j=1}^{q} z^j \right)^t$$

$$= [z^T] \frac{1}{1-z} \sum_{t=0}^{\infty} \left(\sum_{j=1}^{q} z^j \right)^t = [z^T] \frac{1}{(1-z)} \left(1 - \sum_{j=1}^{q} z^j \right)^{-1}.$$

Denote now by z_q the smallest singularity of the generating function, i.e., the smallest solution to $z + \cdots + z^q = 1$ for z. Given this singularity of the generating function, we can deduce by standard combinatorial arguments that the asymptotic behavior of $|\mathsf{D}^*(\mathbf{A}_q^T)|$ is $\mathcal{R}^*(\mathbf{A}_q) = -\log z_q$. $\square$

Note that the appearance of the threshold $\alpha = \frac{2}{3}$ in Theorem 3 can be explained using probabilistic arguments as follows. Consider a random binary string $\boldsymbol{x}$ of length n that takes values 0 and 1 with probability $\frac{1}{2}$. Two consecutive symbols x_i and x_{i+1} of $\boldsymbol{x}$ are equivalent with probability $\frac{1}{2}$ and differ with the same probability. Let now T be the shortest length such that $\boldsymbol{S} = \mathbf{A}_2^T$ is a supersequence of $\boldsymbol{x}$. Assembling $\boldsymbol{S}$ symbol-by-symbol, we see that we need to append 2 symbols to $\boldsymbol{S}$ if $x_i = x_{i+1}$ and 1 symbol otherwise. The expected length of $\boldsymbol{S}$ is therefore $T = \frac{2+1}{2}n = \frac{3}{2}n$. It then follows that if $T > \frac{3}{2}n$ (or equivalently $\alpha = \frac{n}{T} < \frac{2}{3}$), $\boldsymbol{x}$ is a subsequence of the alternating sequence $\mathbf{A}_2$ of length T with high probability. Derandomizing the above discussion, it follows that, given $\alpha < \frac{2}{3}$, most $\boldsymbol{x}$ are a subsequence of $\mathbf{A}_2^T$. However, on the other hand when $\alpha \ge \frac{2}{3}$ this is not the case anymore, which explains the threshold. For more details and an interpretation of the results of Theorem 3 using Markov chains we refer the reader to Section IV-C. Figure 4 displays $\mathcal{R}(\mathbf{A}_2,\alpha)$ versus α.

Theorem 3 provides a solution to the problem $\max_{\boldsymbol{S}\in\Sigma^*} \mathcal{R}(\boldsymbol{S},\alpha)$ for binary sequences and to the problem $\max_{\boldsymbol{S}\in\Sigma^*} \mathcal{R}^*(\boldsymbol{S})$ for any q. We will show in Sections IV-A a connection between $\mathcal{R}^*(\boldsymbol{S})$ and the capacity of a cost-constrained channel, along with a technique for computing it for an arbitrary periodic sequence. We will also present evidence of a conjectured relationship between $\mathcal{R}(\boldsymbol{S},\alpha)$ and a capacity associated with a cost-constrained channel with fixed average symbol cost.

IV. Synthesis Codes via Constrained Codes

A. Code Construction for the Alternating Sequence

In this section we present a construction of a family of synthesis codes for the synthesis sequence $\mathbf{A}_q$. Without loss of generality we let the alphabet be $\Sigma = \{0,1,2,\ldots,q-1\}$ while the results apply for arbitrary alphabets of size q.

For a given string $\boldsymbol{x}$ of length n, we start by determining its synthesis time using the alternating sequence $\mathbf{A}_q$, i.e., the value of $t_{\mathbf{A}_q}(\boldsymbol{x})$. A useful tool will hereby be the derivative of the strand $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$, which is defined by $\boldsymbol{x}' = (x_1', x_2', \ldots, x_n') \in \{1,2,\ldots,q\}^n$, where for $1 \le i \le n$, $x_i' = (x_i - x_{i-1}) \pmod q \in \{1,2,\ldots,q\}$, while $x_0 = 0$. Note that this mapping is invertible, i.e., given the derivative $\boldsymbol{x}'$, it is possible to revert back the string $\boldsymbol{x}$. The L_1 weight of a string $\boldsymbol{y}$ is the sum of its entries, that is, $L_1(\boldsymbol{y}) = \sum_{i=1}^{|\boldsymbol{y}|} y_i$.

Lemma 4. *It holds that $t_{\mathbf{A}_q}(\boldsymbol{x}) = L_1(\boldsymbol{x}')$.*

Proof. Assume that $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$. For $1 \le i \le n$, it holds that if the symbol x_i is synthesized on the t_i-th cycle, then the symbol x_{i+1} will be synthesized on the $(t_i + (x_{i+1} - x_i) \pmod q)$-th cycle. Thus, the number of cycles is $\sum_{i=1}^{n} (x_{i+1} - x_i) \pmod q$, which is equal to $L_1(\boldsymbol{x}')$. $\square$

Following Lemma 4, a general code construction of a synthesis code with strands of length n and synthesis time T, with the alternating sequence, is simply given by

$$\mathcal{C}_T^n = \{\boldsymbol{x} \in \Sigma^n \mid L_1(\boldsymbol{x}') \le T\}.$$

However, this does not provide an explicit code construction with efficient encoding and decoding maps. For the special case of $T = \lfloor \frac{q+1}{2} n \rfloor$, we have the following encoder which encodes any strand $\boldsymbol{u} = (u_1, u_2, \ldots, u_{n-1})$ of length $n-1$ into a strand $\boldsymbol{c}$ of length n such that its synthesis time is at most T. Define $\boldsymbol{x}' = (u_1', \ldots, u_{n-1}', 1)$ and the *complement* of some $\boldsymbol{y}$ by $\boldsymbol{y}^{\mathsf{c}}$, where $y_i^{\mathsf{c}} = q + 1 - y_i$. We encode

$$\boldsymbol{c}' = \begin{cases} \boldsymbol{x}', & \text{if } L_1(\boldsymbol{x}') \le T \\ (\boldsymbol{x}')^{\mathsf{c}}, & \text{otherwise} \end{cases}.$$

Since $L_1(\boldsymbol{x}') + L_1((\boldsymbol{x}')^{\mathsf{c}}) = (q+1)n$, it follows that $t_{\mathbf{A}_q}(\boldsymbol{c}) \le T$. In order to decode the strand $\boldsymbol{x}$ from $\boldsymbol{c}$, one simply verifies the value of c_n' which can be 1 or q. In the former $\boldsymbol{u} = (c_1, \ldots, c_{n-1})$, while in the latter $\boldsymbol{u}$ is decoded from $(\boldsymbol{c}')^{\mathsf{c}}$.

B. From Subsequences to Cost-constrained Systems

As we have illustrated in Section IV-A for the alternating sequence, the synthesis time constraint can be translated to a cost constraint for the L_1 weight of a string. We now turn to establishing this connection between the synthesis problem $\mathcal{R}^*(\boldsymbol{S})$ and cost-constrained systems for arbitrary sequences. Note that, although the alternating sequence maximizes the information rates $\mathcal{R}(\boldsymbol{S},\alpha)$ and $\mathcal{R}^*(\boldsymbol{S})$, building the theory

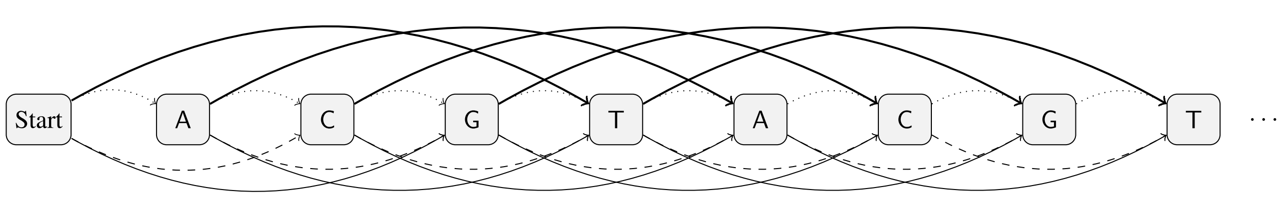

Fig. 2: Subsequence graph $G(\boldsymbol{S})$ for the semi-infinite sequence $\boldsymbol{S} = (\mathsf{ACGTACGT}...)$. Edge decorations highlight the edge weights (Dotted $\triangleq 1$, dashed $\triangleq 2$, solid $\triangleq 3$, thick $\triangleq 4$).

for general sequences can be of importance when, e.g., incorporating other synthesis constraints, such as limited run-lengths. Let $\boldsymbol{S} = (S_1 S_2 \dots)$, $S_i \in \Sigma$ be a semi-infinite sequence. We define the *subsequence graph* $G(\boldsymbol{S})$ of $\boldsymbol{S}$ by the directed graph which has vertices $\mathcal{V} = \{v_0, v_1, v_2, \dots\}$, where a vertex v_i, $i \geq 1$ corresponds to the i-th symbol in $\boldsymbol{S}$ and v_0 is an auxiliary starting vertex. Vertex v_i and v_j, are connected by an edge e of weight $\tau(e) = j - i$, if $j > i$ and $S_k \neq S_j$ for all $i < k < j$. Thus, each vertex v_i has $|\Sigma|$ outgoing edges, denoted by $\mathcal{E}_i$, to the next appearance of each symbol in Σ, succeeding v_i. Fig. 2 shows $G(\boldsymbol{S})$ for $\boldsymbol{S} = (\mathsf{ACGTACGT}...)$. A *path* $\boldsymbol{r}$ of length ℓ through $G(\boldsymbol{S})$ is a sequence $\boldsymbol{r} = (v_i, v_{i_1}, \dots, v_{i_\ell})$ of consecutive vertices, starting from some vertex v_i. Its generated sequence is $g(\boldsymbol{r}) = (S_{i_1} S_{i_2} \dots S_{i_\ell})$ and has length ℓ. We define $M_i(n, n')$, $c \in \{0, 1, \dots, n\}$ to be the number of paths of length exactly n' that have a total edge weight of at most n and start from vertex v_i in $G(\boldsymbol{S})$. $G(\boldsymbol{S})$ compactly characterizes all subsequences of $\boldsymbol{S}$ as paths through $G(\boldsymbol{S})$. The following lemma establishes the exact relationship between the number of subsequences of a sequence $\boldsymbol{S}$ and the graph $G(\boldsymbol{S})$.

Lemma 5. *For any semi-infinite sequence $\boldsymbol{S}$, the number of subsequences of $\boldsymbol{S}_{1:n}$ of length $n - t$ is given by*

$$|\mathsf{D}(\boldsymbol{S}_{1:n}, t)| = M_0(n, n - t).$$

Proof. Denote by $\mathcal{Q}$ the set of all paths of length $n-t$ through $G(\boldsymbol{S})$ that start from v_0 and have weight at most n. We will show that $|\mathcal{Q}| = |\mathsf{D}(\boldsymbol{S}_{1:n}, t)|$. First, notice that for each $\boldsymbol{r} \in \mathcal{Q}$, we have $g(\boldsymbol{r}) \in \mathsf{D}(\boldsymbol{S}_{1:n}, t)$ by construction of the graph $G(\boldsymbol{S})$. On the other hand, let $\boldsymbol{x} \in \mathsf{D}(\boldsymbol{S}_{1:n}, t)$ and let $1 \leq i_1 < i_2 < \dots < i_{n-t} \leq n$ with $\boldsymbol{x} = (S_{i_1} S_{i_2} \dots S_{i_\ell})$ be the left-most alignment of $\boldsymbol{x}$ in $\boldsymbol{S}$. Then, $\boldsymbol{r} = (v_0, v_{i_1}, \dots, v_{i_{n-t}})$ is a path through $G(\boldsymbol{S})$ with $g(\boldsymbol{r}) = \boldsymbol{x}$. Finally, using that two non-identical paths $\boldsymbol{r}_1, \boldsymbol{r}_2 \in \mathcal{Q}$, $\boldsymbol{r}_1 \neq \boldsymbol{r}_2$, generate two different sequences $g(\boldsymbol{r}_1) \neq g(\boldsymbol{r}_2)$, we obtain $|\mathcal{Q}| = |\{\boldsymbol{x} : \boldsymbol{x} = g(\boldsymbol{r}), \boldsymbol{r} \in \mathcal{Q}\}| = |\mathsf{D}(\boldsymbol{S}_{1:n}, t)|$. ☐

Having available this correspondence, we directly obtain an efficient way to compute the number of subsequences.

$$M_i(c, n') = \begin{cases} \sum\limits_{e \in \mathcal{E}_i : \tau(e) \leq c} M_{i+\tau(e)}(c - \tau(e), n' - 1), & \text{if } n' > 0, \\ 1, & \text{if } n' = 0. \end{cases} \quad (1)$$

We now turn to compute $|\mathsf{D}(\boldsymbol{S}_{1:n}), t|$ for a periodic sequence $\boldsymbol{S} = (\boldsymbol{s}\boldsymbol{s}\dots)$, where $\boldsymbol{s} \in \Sigma^L$ is the period of $\boldsymbol{S}$ and $L \in \mathbb{N}$ is the period length of the sequence $\boldsymbol{S}$. Clearly, also for such a sequence $\boldsymbol{S}$, the graph $G(\boldsymbol{S})$ allows to explore the subsequence spectrum. However, it is possible to simplify the graph by taking into account the periodicity. In particular, we observe that $M_i(c, n') = M_{i+zL}(c, n')$ for any $i \in \{1, \dots, L\}$ and any integer $z \geq 0$. This motivates to introduce the

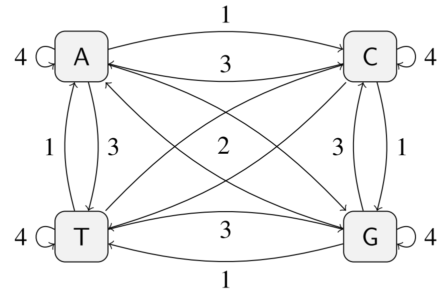

Fig. 3: Simplified graph $\widehat{G}(\boldsymbol{s})$ for $\boldsymbol{s} = (\mathsf{ACGT})$. Hereby, the last symbol of the period, T, can take the role of the starting vertex.

variables $\widehat{M}_i(c, n') \overset{\text{def}}{=} M_i(c, n')$, $i \in \{1, \dots, L\}$, which obey the recursive relationship derived in (1). Note that for the special case of alternating and balanced sequences, alternative recursive expressions for the number of subsequences have been observed in [15] and [16]. It is hence natural to define a simplified graph $\widehat{G}(\boldsymbol{s})$ for periodic sequences $\boldsymbol{S} = (\boldsymbol{s}\boldsymbol{s}...)$ in the following manner. Construct a graph with L vertices $\widehat{\mathcal{V}}$, one for each symbol in $\boldsymbol{s} \in \Sigma^L$. Construct the edges $\widehat{\mathcal{E}}$ according to the rule for $G(\boldsymbol{S})$, with the only difference that a vertex v_i is cyclically connected to q vertices $v_{i+\tau(e) (\bmod L)}$, for each $e \in \mathcal{E}_i$. The simplified periodic graph $\widehat{G}(\boldsymbol{s})$ of $\boldsymbol{S} = (\boldsymbol{s}\boldsymbol{s}...)$ with $\boldsymbol{s} = (\mathsf{ACGT})$ is depicted in Fig. 3. We are now in the position to state the final result of this section.

Theorem 6. *Let $\boldsymbol{S} = (\boldsymbol{s}\boldsymbol{s}...)$ be any semi-infinite periodic sequence with period $\boldsymbol{s} \in \Sigma^L$. Then,*

$$\mathcal{R}^*(\boldsymbol{S}) = C(\widehat{G}(\boldsymbol{s})),$$

where $C(\widehat{G}(\boldsymbol{s}))$ is the combinatorial capacity [17] of the cost-constrained channel defined by the graph $\widehat{G}(\boldsymbol{s})$.

Proof. Lemmas 2 and 5 imply that $N^n(\boldsymbol{S}_{1:T}, T) = \widehat{M}_0(T, n)$. So $N^*(\boldsymbol{S}_{1:T}, T) = \sum_{n=1}^{T} \widehat{M}_0(T, n)$. Let $\widetilde{M}_0(T, n) = \widehat{M}_0(T, n) - \widehat{M}_0(T-1, n)$ denote the number of length-n paths that start from v_0 and have weight *exactly* T, and define $\widetilde{M}_0^*(T) = \sum_{n=1}^{T} \widetilde{M}_0(T, n)$, the number of paths that start from v_0 and have weight exactly T. Then $\widetilde{M}_0^*(T) = N^*(\boldsymbol{S}_{1:T}, T) - N^*(\boldsymbol{S}_{1:T-1}, T-1)$. The graph $\widehat{G}(\boldsymbol{s})$ is strongly connected, and the edges emanating from any vertex correspond to distinct symbols in $\boldsymbol{s}$. The theorem follows from the definition of combinatorial capacity [17] and arguing as in [18, Theorem 3.4] and [17]. ☐

C. Information Rates for DNA Synthesis

Theorem 6 shows that the optimum information rate for DNA synthesis from an arbitrary periodic sequence $\boldsymbol{S}$ can be determined using tools from the theory of cost-constrained channels [17]. In this section, we review the approach and provide several illustrative examples.

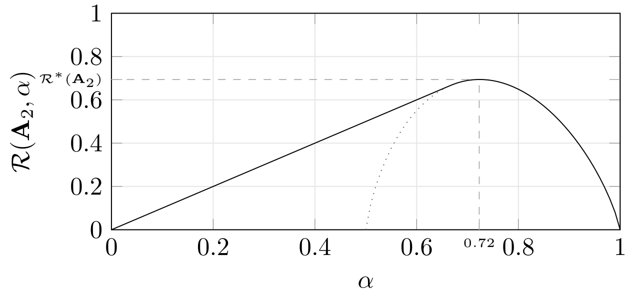

Fig. 4: Number of bits per unit synthesis time $\mathcal{R}(\mathbf{A}_2, \alpha)$

Given the graph $\widehat{G}(\boldsymbol{s})$, we define the $L \times L$ edge cost partition matrix $P(s)$. For a pair (i,j), the entry $P(z)_{i,j}$ is

$$P(z)_{i,j} = \begin{cases} 0, & \text{if there is no edge } e: i \to j \\ 2^{-z\tau(e)}, & \text{if edge } e: i \to j \text{ has cost } \tau(e) \end{cases}.$$

Let $\rho(z)$ be the largest eigenvalue of $P(z)$. The (combinatorial) capacity of the cost-constrained channel is given by $C = z_0$, where z_0 is the unique solution of $\rho(z) = 1$, or, equivalently, the largest real solution of the determinental equation

$$\det(I - P(z)) = 0.$$

This represents the maximum information rate of a synthesis code using the synthesis sequence $\boldsymbol{S} = (\boldsymbol{s}\boldsymbol{s}\ldots)$, measured in units of bits per synthesis cycle. Its inverse $T^*_{\text{bit}} = C^{-1}$ is the minimum possible number of synthesis cycles per bit.

The capacity can also be described as the maximum normalized entropy of a stationary Markov chain on the costly channel graph $\widehat{G}(\boldsymbol{s})$, as follows. Let π_i, $i \in \{1,\ldots,L\}$ be the stationary state probabilities, and let p_e denote the transition probability associated with the edge $e \in \widehat{\mathcal{E}}$. The entropy of the Markov chain $\mathcal{P}$ is defined by $H(\mathcal{P}) = \sum_{i=1}^{L} \pi_i \sum_{e \in \widehat{\mathcal{E}}_i} p_e \log p_e$. The average cost (per symbol) associated with the graph $\widehat{G}(\boldsymbol{s})$ is $T_{\text{s}}(\mathcal{P}) = \sum_{i=1}^{L} \pi_i \sum_{e \in \widehat{\mathcal{E}}_i} p_e \tau(e)$. The (probabilistic) capacity of the cost-constrained channel is then given by

$$C = \max_{\mathcal{P}} \frac{H(\mathcal{P})}{T_{\text{s}}(\mathcal{P})}.$$

The equivalence between combinatorial capacity and probabilistic capacity is proved in [17]. If $\mathcal{P}^*$ is the unique capacity-achieving Markov chain, then the quantity $R^* \stackrel{\text{def}}{=} H(\mathcal{P}^*) = CT_{\text{s}}(\mathcal{P}^*)$ can be interpreted as the coding rate of an optimal synthesis code in units of bits per symbol.

For the graph $\widehat{G}(\boldsymbol{s})$ in Figure 3 corresponding to the sequence $\boldsymbol{s} = (\mathsf{ACGT})$, we find that $\rho(z) = 2^{-z} + 2^{-2z} + 2^{-3z} + 2^{-4z}$, implying capacity $C = s_0 \approx 0.9468$ bits/synthesis cycle, with $T^*_{\text{bit}} = 1.0562$ cycles per bit[2]. The corresponding Markov chain $\mathcal{P}^*$ is defined by $q^*_e = 2^{-C\tau(e)}$ and $\pi^*_i = 1/4$ for all i. The average synthesis time per code symbol is $T_{\text{s}}(\mathcal{P}^*) \approx 1.7657$ cycles per symbol and the coding rate is $R^* \approx 1.6717$ bits per code symbol. The inverse rate $f^* = (R^*)^{-1} \approx 0.5981$ can be thought of as an *expansion factor*. It is interesting to compare this optimal coding scheme with the construction from Section IV-A. If we synthesize codewords from that construction using the alternating synthesis sequence with period $\boldsymbol{s} = (\mathsf{ACGT})$, the resulting scheme will have a higher coding rate $R = 2$ bits/code symbol and lower expansion factor $f = R^{-1} = 0.5$. However, the synthesis times per symbol and bit are $T_{\text{s}} = 2.5$ and $T_{\text{bit}} = 1.25$, respectively. We see that the increased expansion factor of the optimal code leads to substantial reductions in synthesis time per code symbol and per bit.

[2]A similar analysis for a general q-ary alternating synthesis sequence shows that $\rho(z) = \sum_{i=1}^{q} 2^{-iz}$, thus recovering the result of Theorem 3.

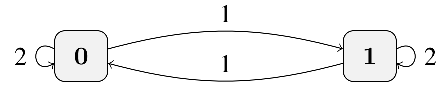

Fig. 5: Graph $\widehat{G}(\boldsymbol{s})$ for $\boldsymbol{s} = (01)$.

More generally, if we fix the average synthesis time per symbol T_{s}, we can find the corresponding maximum coding rate

$$R(T_{\text{s}}) = \max_{\mathcal{P}} \ H(\mathcal{P}) \ \text{ s.t. } \ T_{\text{s}}(\mathcal{P}) = T_{\text{s}},$$

the corresponding synthesis time per bit $T_{\text{bit}} = T_s/R(T_s)$, and the number of bits per unit synthesis cycle $C(T_s) = T_{\text{bit}}^{-1}$. For the q-ary alternating sequence, the range of possible values of T_{s} is $[T_{\min}, T_{\max}] = [\min_e \tau(e), \max_e \tau(e)] = [1, q]$. The parametric expressions for $[R(T_{\text{s}}), T_{\text{s}}]$ in terms of the parameter z are given by

$$T_{\text{s}}(z) = \frac{\rho'(z)}{(\ln 2)\rho(z)}, R(z) = \log_2 \rho(z) - z\frac{\rho'(z)}{(\ln 2)\rho(z)},$$

where $z \in (-\infty, \infty)$. Note that there is a critical value $T_{\text{crit}} \in [T_{\min}, T_{\max}]$ such that for $T_{\text{s}} > T_{\text{crit}}$ an entropy larger than $R(T_{\text{s}})$ is achieved by maximizing over Markov chains with average symbol cost *less than* T_{s}. Returning to the example where $q = 4$, we find that $\rho(z) = \sum_{i=1}^{4} 2^{-iz}$, from which we can readily derive $T_{\text{s}}(z)$ and $R(z)$.

The analysis of the $[R(T_{\text{s}}), T_{\text{s}}]$ trade-off for the binary alternating sequence $\mathbf{A}_2$ with period $\boldsymbol{s} = (01)$ leads to an interesting connection with the quantity $\mathcal{R}(\boldsymbol{S}, \alpha)$ defined in Section II-A. The graph $\widehat{G}(\boldsymbol{s})$ is shown in Figure 5. The maximum-entropy Markov chain with $T_{\text{s}} \in [1,2]$ has edge transition probabilities p and $1-p$, corresponding to edge costs 1 and 2, respectively, with average cost $(1-p) + 2p = 1 + p = T_{\text{s}}$. The entropy $R = h(p)$, and $C(T_{\text{s}}) = h(p)/(1+p)$. Set $\alpha = T_{\text{s}}^{-1} = (1+p)^{-1}$. For $T_{\text{s}} \leq T_{\text{crit}} = 3/2$, or $\alpha \geq \alpha_{\text{crit}} = 2/3$, we have $C(T_{\text{s}}) = \alpha h(\frac{1-\alpha}{\alpha}) = \mathcal{R}(\mathbf{A}_2, \alpha)$, where the second equality follows from Theorem 3. We conjecture that this is a special case of a more general result relating $\mathcal{R}(\boldsymbol{S}, \alpha)$ to a combinatorial capacity for fixed average symbol cost, analogous to Theorem 6, and an equivalence between this combinatorial capacity and a corresponding probabilistic capacity. Figure 4 shows a plot of $\mathcal{R}(\mathbf{A}_2, \alpha)$ versus α.

To illustrate the application of this analysis to non-alternating periodic sequences, we compare the capacities of the binary synthesis sequences with periods $\boldsymbol{s}_0 = (01)$, $\boldsymbol{s}_1 = (001)$, and $\boldsymbol{s}_2 = (0011)$. For $\boldsymbol{s}_0 = (01)$, we know $C_0 = \log_2 \frac{1+\sqrt{5}}{2} \approx 0.6942$. For $\boldsymbol{s}_1 = (001)$, we have $C_1 = \log \lambda_1$ where λ_1 is the largest positive root of $\lambda^6 - 4\lambda^3 + 1$. The largest real root $\lambda_1 \approx 1.5511$, so $\log \lambda_1 \approx 0.6333$. Finally, for $\boldsymbol{s}_2 = (0011)$, $C_2 = \log \lambda_2$ where λ_2 is the largest positive root of $\lambda^8 - 6\lambda^4 + 1$. The root is $\lambda_2 \approx 1.5538$ and $C_2 = \log \lambda_2 \approx 0.6358$. As expected, the capacity of the alternating sequence with period $\boldsymbol{s}_0$ is largest.

The design of efficient DNA synthesis codes will be discussed in a subsequent work. There is a substantial literature on constructing codes for cost-constrained channels. For pointers to the literature, see [19].

References

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

[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, Feb. 2013.

[3] L. Organick, S. D. Ang, Y.-J. Chen, R. Lopez, S. Yekhanin, K. Makarychev, M. Z. Racz, G. Kamath, P. Gopalan, B. Nguyen, C. N. Takahashi, S. Newman, H.-Y. Parker, C. Rashtchian, K. Stewart, G. Gupta, R. Carlson, J. Mulligan, D. Carmean, G. Seelig, L. Ceze, and K. Strauss, "Random access in large-scale DNA data storage," *Nat. Biotechnol.*, vol. 36, no. 3, pp. 242–248, Mar. 2018.

[4] S. M. H. T. Yazdi, R. Gabrys, and O. Milenkovic, "Portable and error-free DNA-based data storage," *Sci. Rep.*, vol. 7, no. 1, p. 5011, Dec. 2017.

[5] R. N. Grass, R. Heckel, M. Puddu, D. Paunescu, and W. J. Stark, "Robust chemical preservation of digital information on DNA in silica with error-correcting codes," *Angew. Chem. Int. Ed.*, vol. 54, no. 8, pp. 2552–2555, Feb. 2015.

[6] A. Lenz, P. H. Siegel, A. Wachter-Zeh, and E. Yaakobi, "Coding over sets for DNA storage," *IEEE Trans. Inf. Theory*, 2019.

[7] S. Newman, A. P. Stephenson, M. Willsey, B. H. Nguyen, C. N. Takahashi, K. Strauss, and L. Ceze, "High density DNA data storage library via dehydration with digital microfluidic retrieval," *Nat. Commun.*, vol. 10, no. 1, p. 1706, Dec. 2019.

[8] C. Rashtchian, K. Makarychev, M. Racz, S. Ang, D. Jevdjic, S. Yekhanin, L. Ceze, and K. Strauss, "Clustering billions of reads for DNA data storage," in *Proc. Conf. Neural Inf. Process. Sys.*, Long Beach, CA, Dec. 2017, p. 12.

[9] M. Willsey, K. Strauss, L. Ceze, A. P. Stephenson, C. Takahashi, P. Vaid, B. H. Nguyen, M. Piszczek, C. Betts, S. Newman, and S. Joshi, "Puddle: A dynamic, error-correcting, full-stack microfluidics platform," in *Proc. Int. Conf. Architectural Support for Programming Languages and Operating Systems*. Providence, RI, USA: ACM Press, 2019, pp. 183–197.

[10] M. H. Caruthers, "The chemical synthesis of DNA/RNA: Our gift to science," *J. Biol. Chem.*, vol. 288, no. 2, pp. 1420–1427, Jan. 2013.

[11] L. Ceze, J. Nivala, and K. Strauss, "Molecular digital data storage using DNA," *Nat. Rev. Genet.*, vol. 20, no. 8, pp. 456–466, Aug. 2019.

[12] S. Kosuri and G. M. Church, "Large-scale de novo DNA synthesis: technologies and applications," *Nature Methods*, vol. 11, no. 5, pp. 499–507, May 2014.

[13] L. Anavy, I. Vaknin, O. Atar, R. Amit, and Z. Yakhini, "Data storage in DNA with fewer synthesis cycles using composite DNA letters," *Nat. Biotechnol.*, vol. 37, no. 10, pp. 1229–1236, Oct. 2019.

[14] L. Calabi, "On the computation of Levenshtein's distances," Parke Mathematical Laboratories, Inc., Carlisle, MA, Tech. Rep., 1967.

[15] D. S. Hirschberg and M. Regnier, "Tight bounds on the number of string subsequences," *Journal of Discrete Algorithms*, vol. 1, no. 1, pp. 123–132, 2000.

[16] Y. Liron and M. Langberg, "A characterization of the number of subsequences obtained via the deletion channel," *IEEE Trans. Inf. Theory*, vol. 61, no. 5, pp. 2300–2312, May 2015.

[17] A. Khandekar, R. J. McEliece, and E. R. Rodemich, "The discrete noiseless channel revisited," in *Coding, Communications, and Broadcasting*. Badlock: Research Studies Series Ltd., UK, 2000, pp. 115–137. [Online]. Available: http://www.mceliece.caltech.edu/publications/PostFinal.ps

[18] B. H. Marcus, R. M. Roth, and P. H. Siegel, *An introduction to Coding for Constrained Systems*, 2001. [Online]. Available: http://ronny.cswp.cs.technion.ac.il/wp-content/uploads/sites/54/2016/05/chapters1-9.pdf

[19] Y. Liu, P. Huang, A. W. Bergman, and P. H. Siegel, "Rate-constrained shaping codes for structured sources," *IEEE Trans. Inf. Theory*, pp. 1–23, Apr. 2020, (early access).