DNA Synthesis Using Shortmers

Maria Abu-Sini
Technion - Israel Institute of Technology
Haifa 3200009, Israel
maria.as@cs.technion.ac.il

Andreas Lenz
Technical University of Munich
DE-80333 Munich, Germany
andreas.lenz@mytum.de

Eitan Yaakobi
Technion - Israel Institute of Technology
Haifa 3200009, Israel
yaakobi@cs.technion.ac.il

***Abstract*—In conventional DNA synthesis machines many strands are usually synthesized in parallel by iterating through a supersequence s and adding in each cycle a single nucleotide to a subset of the strands. Then, the length of s determines the number of the cycles, hence the time and the cost of the synthesis process too. Recently, in order to optimize the synthesis process, researchers have suggested to append in each cycle a shortmer instead of a single nucleotide. The present work studies this optimization from a theoretical point of view. In particular, it discusses which shortmers are the best to use, and how to calculate the number of cycles required to synthesize in parallel a set of strands using a set of shormers. Lastly, and following a previously described connection between the DNA synthesis problem and costly constrained graphs, the paper investigates calculating the capacities of such non-deterministic graphs.**

I. Introduction

In DNA storage systems archival data is stored by synthesizing artificial DNA strands of specific sequences. This process is known by its high runtime complexity and monetary costs, hence its optimization has been the main goal of several recent papers [1], [2], [5], [7], [11]–[13], [15]–[17]. The present work, as well, investigates the DNA synthesis process; specifically by extending the study of [11] and [12].

To describe the operation of the synthesis machine let $S \subseteq \{A,C,G,T\}^*$ be the set of strands we aim to synthesize. Moreover, let $B \subseteq \{A,C,G,T\}^*$ be a set of b shortmers (sequences of nucleotides) and $\boldsymbol{s}$ be a length-ℓ sequence over B referred to as the *synthesis sequence*. Then the machine iterates over $\boldsymbol{s}$ and in each cycle adds to some strands the next shortmer in $\boldsymbol{s}$. Such a process synthesizes in parallel all of the strands of S in ℓ cycles, which means, the length of $\boldsymbol{s}$ determines the number of cycles, hence the time and monetary cost too [1], [3], [4], [10]. Therefore, one should aim to decrease the number of cycles; for example by choosing $\boldsymbol{s}$ to be a shortest common supersequence of the strands of S when $B = \{A,C,G,T\}$. However, usually an enormous number of strands are synthesized in parallel. Thus, choosing $\boldsymbol{s}$ accordingly might be unfeasible in terms of runtime complexity. In this case, $\boldsymbol{s}$ is chosen independently, yet wisely to enable synthesizing many strands using a small number of cycles. That is, instead of choosing $\boldsymbol{s}$ according to S, we fix $\boldsymbol{s}$ to some semi-infinite sequence and use its prefix to synthesize in parallel the strands of S.

In conventional synthesis machines $B = \{A,C,G,T\}$ and this case was extensively studied in [11] and [12]. The present work follows the lines of these papers to investigate the general case of $B \subseteq \{A,C,G,T\}^*$. In particular, Section III tackles the case of unfixed synthesis sequence and proposes an efficient algorithm to find a shortest possible $\boldsymbol{s}$ given S and B. Then, Section IV addresses the case of fixed synthesis sequence by formalizing and discussing the following questions.

- Given $b \in \mathbb{N}$, assume one may choose the set of b shortmers B and the synthesis sequence $\boldsymbol{s} \in B^*$ arbitrarily. Which shortmers and synthesis sequence should be used?
- Given $b, \ell \in \mathbb{N}$, assume $B = \{A,C,G,T\} \cup \{x\}$ and $\boldsymbol{s} = (A,C,G,T,\boldsymbol{x},A,C,G,T,\boldsymbol{x},\dots)$, where $\boldsymbol{x} \in \{A,C,G,T\}^{\ell}$. Which shortmer $\boldsymbol{x}$ should be used?

Next, Section V reviews the connection between the synthesis problem and calculating capacities of costly constrained graphs as established in [11] and [12]. It also shows that for the study of the case $B \subseteq \{A,C,G,T\}^*$, one may need to calculate the capacity of lossy costly constrained graphs. Though deterministic costly constrained graphs have been extensively studied in [9], [12] and [14], to the best of our knowledge, non-deterministic ones have never been addressed. Therefore, Section V takes the first steps towards studying these graphs by deriving bounds on their capacities. Lastly, before discussing the topics mentioned earlier in Sections III, IV, and V, Section II provides the notations used throughout the paper. Due to the lack of space, proofs are omitted from the paper.

II. Definitions and Preliminaries

This section introduces the notations and the problems tackled in the present work. First, for $n \in \mathbb{N}$, $[n] = \{1,2,\dots,n\}$. Second, given an alphabet $\Sigma = \{\sigma_1,\dots,\sigma_q\}$, a sequence $\boldsymbol{y} = y_1 \cdots y_n \in \Sigma^*$ is sometimes written as a vector $(y_1,\dots,y_n)$. In addition, $\boldsymbol{y}_{[i,j]}$ denotes the substring $y_i \cdots y_j = (y_i,\dots,y_j)$ and $|\boldsymbol{y}|$ stands for the length of the sequence. A sequence $\boldsymbol{z} \in \Sigma^{\ell}$ is called a supersequence of $\boldsymbol{y}$ if there exist indices $1 \leqslant i_1 < i_2 < \cdots < i_n \leqslant \ell$ such that $\boldsymbol{y} = (z_{i_1}, z_{i_2}, \dots, z_{i_n})$. Given a set S, $|S|$ refers to the size of the set while $SCS(S)$ denotes the set of all shortest common supersequences of S.

Though for the study of DNA storage systems it suffices to consider the alphabet $\{A,C,G,T\}$, we derive all of the results for an arbitrary alphabet of size q, $\Sigma = \{\sigma_1,\sigma_2,\dots,\sigma_q\}$. Next, let $B = \{\boldsymbol{x}^1,\boldsymbol{x}^2,\dots,\boldsymbol{x}^b\} \subseteq \Sigma^*$. Then a sequence $\boldsymbol{y} \in \Sigma^*$ may be obtained by concatenating B's sequences, just as it is obtained by concatenating Σ's symbols. Therefore, B may be seen as an alphabet of size b, and we define a mapping $G : B^* \to \Sigma^*$ as follows. Given $\boldsymbol{z} \in B^*$, $G(\boldsymbol{z})$ is the sequence over Σ obtained by concatenating $\boldsymbol{z}$'s elements. Example 1 clarifies this definition and shows that sometimes a sequence in Σ^* may be obtained by several concatenations of B's sequences, thereby G is not necessarily an injective mapping.

Example 1 Let $\Sigma = \{A,C,G,T\}$,

$$B = \{\boldsymbol{x}^1 = A, \boldsymbol{x}^2 = C, \boldsymbol{x}^3 = AC, \boldsymbol{x}^4 = G, \boldsymbol{x}^5 = T\},$$

$$B' = \left\{\boldsymbol{u}^1 = AC, \boldsymbol{u}^2 = AA, \boldsymbol{u}^3 = C, \boldsymbol{u}^4 = G, \boldsymbol{u}^5 = T\right\},$$

and $\boldsymbol{y} = ACAC \in \Sigma^*$. Furthermore, let $\boldsymbol{z}^1 = (\boldsymbol{x}^1,\boldsymbol{x}^2,\boldsymbol{x}^1,\boldsymbol{x}^2) = (A,C,A,C), \boldsymbol{z}^2 = (\boldsymbol{x}^1,\boldsymbol{x}^2,\boldsymbol{x}^3) = (A,C,AC), \boldsymbol{z}^3 = (\boldsymbol{x}^3,\boldsymbol{x}^1,\boldsymbol{x}^2) = (AC,A,C)$, and $\boldsymbol{z}^4 = (\boldsymbol{x}^3,\boldsymbol{x}^3) = (AC,AC)$. Then, $\boldsymbol{z}^1,\boldsymbol{z}^2,\boldsymbol{z}^3,\boldsymbol{z}^4 \in B^*$ where $|\boldsymbol{z}^1| = 4, |\boldsymbol{z}^2| = |\boldsymbol{z}^3| = 3, |\boldsymbol{z}^4| = 2$, and $G(\boldsymbol{z}^1) = G(\boldsymbol{z}^2) = G(\boldsymbol{z}^3) = G(\boldsymbol{z}^4) = \boldsymbol{y}$. Moreover, the length-2 sequence $\boldsymbol{z}^5 = (\boldsymbol{u}^1,\boldsymbol{u}^1) = (AC,AC) \in B'^*$ is the only representation of $ACAC$ over B', i.e., is the only sequence in B'^* satisfying $G(\boldsymbol{z}^5) = \boldsymbol{y}$.

Example 1 motivates introducing the following definition, which states that B given in Example 1 provides a non-uniquely decodable presenation, while B' provides a uniquely decodable one.

Definition 1. *Let $B \subseteq \Sigma^*$. Then B provides a uniquely decodable presentation if for every sequence $\boldsymbol{y} \in \Sigma^*$, $|\{\boldsymbol{z} \in B^* : G(\boldsymbol{z}) = \boldsymbol{y}\}| \leqslant 1$. Otherwise, B provides a non-uniquely decodable presentation.*

In the present work we borrow the term *shortmer* (meaning a sequence of nucleotides in biology) to refer to specific short sequences in Σ^* (which might be of length 1). Moreover, Subsection II-A describes the synthesis process while Subsections II-B and II-C pose the problems studied in the paper.

A. Description of the DNA Synthesis Process

First, we dedicate the following definition to explain the operation of the synthesis machine.

Definition 2. *Let $\Sigma = \{\sigma_1, \sigma_2, \ldots, \sigma_q\}$ be an alphabet of size q. A synthesis machine uses the set of b shortmers $B = \{\boldsymbol{x}^1, \ldots, \boldsymbol{x}^b\}$ where $|\boldsymbol{x}^i| = \ell_i$ for every $1 \leqslant i \leqslant b$ and operates as follows. Let $S = \{\boldsymbol{y}^1, \ldots, \boldsymbol{y}^m\} \subseteq \Sigma^*$ be a set of m sequences to synthesize where $|\boldsymbol{y}^i| = n_i$ for every $1 \leqslant i \leqslant m$. Then the synthesis machine iterates over a sequence $\boldsymbol{s} \in B^*$, referred to as the synthesis sequence, and in each cycle adds the next shortmer to some strands until they are all synthesized.*

The following observation plays an integral role in the rest of the paper and is further emphasized in Example 2.

Observation 1 Following the notations of Definition 2, the length of the synthesis sequence $\boldsymbol{s}$ determines the number of the synthesis cycles. Furthermore, a sequence $\boldsymbol{y}^i \in S$ may be synthesized only using a sequence of cycles $\boldsymbol{z} \in B^*$ that fulfills $G(\boldsymbol{z}) = \boldsymbol{y}^i$. Since $\boldsymbol{s}$ enables synthesizing all of the m sequences of S, then for every $\boldsymbol{y}^i \in S$, $\boldsymbol{s}$ is a supersequence of some $\boldsymbol{z} \in B^*$ that satisfies $G(\boldsymbol{z}) = \boldsymbol{y}^i$.

Example 2 Let $\Sigma = \{A, C, G, T\}$ and $B = \Sigma \cup \{AT, AC, CCG\}$. In addition, let $\boldsymbol{y}^1 = ACCG$. Then, $\boldsymbol{y}^1$ may be synthesized using the synthesis sequence $\boldsymbol{s}^1 = (A, C, C, G), \boldsymbol{s}^2 = (AC, C, G), \boldsymbol{s}^3 = (A, CCG)$ in $4, 3, 2$, cycles, respectively, for $G(\boldsymbol{s}^1) = G(\boldsymbol{s}^2) = G(\boldsymbol{s}^3) = \boldsymbol{y}^1$. However, $\boldsymbol{y}^1$ cannot be synthesized using (AT, CCG) for only complete shortmers are added to the strands in each cycle. Next, let $S = \{\boldsymbol{y}^2 = ATAT, \boldsymbol{y}^3 = CAAT, \boldsymbol{y}^4 = CTAT\}$. Then, the synthesis sequence $\boldsymbol{s}^4 = (C, A, T, A, T), \boldsymbol{s}^5 = (C, A, T, AT)$ enables synthesizing in parallel the 3 strands of S in $5, 4$ cycles, respectively. Furthermore, as mentioned in Observation 1, for every $2 \leqslant i \leqslant 4$, there exists a subsequence $\boldsymbol{z}^i$ of $\boldsymbol{s}^4$ that satisfies $G(\boldsymbol{z}^i) = \boldsymbol{y}^i$, and the same holds for $\boldsymbol{s}^5$ too.

B. Problems Pertaining to the DNA Synthesis Study

This paper extends the study of the case $B = \Sigma$ as published in [11] and [12] to investigate the general case of $B \subseteq \Sigma^*$. As demonstrated throughout the paper, when $B \subseteq \Sigma^*$ provides a uniquely decodable presentation, many problems can be easily solved using the techniques of [11] and [12]. It hence remains to tackle the challenging case of B providing a non-uniquely decodable presentation, as done in the upcoming sections.

By Observation 1, the length of the synthesis sequence determines the number of the cycles, consequently the runtime and monetary cost too. Thus, we aim to use the shortest possible synthesis sequences as formalized in Problem 1.

Problem 1 Let Σ, B, and S be as given in Definition 2 and define $F(S, B)$ to be a set of shortest possible sequences in B^* that enable synthesizing in parallel all of the sequences of S. Design an algorithm that receives Σ and B, and outputs an arbitrary sequence in $F(S, B)$.

For instance, in continuation of Example 2, $F(\{\boldsymbol{y}^1\}, B) = \{\boldsymbol{s}^3\}$ and $F(S, B) = \{\boldsymbol{s}^5\}$. However, in DNA storage systems, usually S is very large, which makes any solution to Problem 1 impractical. As a result, $\boldsymbol{s}$ is usually chosen (independently) to be a fixed semi-infinite synthesis sequence and a prefix of it is used to synthesize S. The length of this prefix determines the number of the synthesis cycles. For example, when $\Sigma = \{A, C, G, T\}$ and $\boldsymbol{s} = (A, C, G, T, A, C, G, T, \ldots)$ is a semi-infinite periodic alternating sequence, $4, 13$ cycles are required to synthesize $ACGT, AAAA$, respectively. Therefore, it is preferable to use synthesis sequences that enable synthesizing many strands using short prefixes. The following definition taken from [11] and [12] formalizes this discussion.

Definition 3. *Let $\boldsymbol{s} \in B^*$ be a semi-infinite synthesis sequence and $N_{\boldsymbol{s}}(t)$ be the number of distinct sequences over Σ that can be synthesized using $\boldsymbol{s}_{[1,t]}$. Then the information rate of $\boldsymbol{s}$ is given by $R_{\boldsymbol{s}} = \limsup_{t \to \infty} \frac{\log_2 N_{\boldsymbol{s}}(t)}{t}$.*

Problem 2, which will be addressed in Section IV, seeks sequences with optimal information rates.

Problem 2 Given $b \in \mathbb{N}$ and Σ, find a set of shortmers $B \subseteq \Sigma^*$ of size b and a semi-infinite synthesis sequence $\boldsymbol{s} \in B^*$ that attain the maximum possible information rate $R_{\boldsymbol{s}}$, which we denote by $F_1(\Sigma, b)$.

Problem 2 allows using arbitrary sets B. However, to enable synthesizing any strand of any length, B should include all of Σ in addition to other shortmers, thereby we pose Problem 3 and tackle it in Section IV.

Problem 3 Given $k \in \mathbb{N}$ and Σ, find $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_k \in \Sigma^*$ such that the set of shortmers $B = \Sigma \cup \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_k\}$ and the semi-infinite periodic alternating synthesis sequence $\boldsymbol{s} = (\sigma_1, \ldots, \sigma_q, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_k, \sigma_1, \ldots, \sigma_q, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_k, \ldots)$ attain the maximum possible information rate $R_{\boldsymbol{s}}$, which we denote by $F_2(\Sigma, k)$.

C. Calculating the Capacity of Non-deterministic Costly Constrained Graphs

When $B = \Sigma$, [11] and [12] developed a technique to calculate the information rate by computing the capacity of some costly constrained graph. According to Section V, the information rate can be successfully calculated using this technique if $B \subseteq \Sigma^*$ provides a uniquely decodable presentation. Otherwise, this technique requires computing the capacity of lossy costly constrained graphs, which to the best of our knowledge has not been investigated before. Therefore, and since calculating capacities of such graphs is an interesting topic by itself, we raise Problem 4 and approach it in Section V. First we formally define lossy costly constrained graphs.

Definition 4. *A costly constrained graph $G = (V, E, \sigma, \tau)$ has vertices V and directed edges E. Each edge has a label, weight determined by the function $\sigma : E \to \Sigma, \tau : E \to \mathbb{N}$, respectively. Then a path $\boldsymbol{p} = (e_1, e_2, \ldots, e_n) \in E^n$ generates the sequence $\sigma(\boldsymbol{p}) = (\sigma(e_1), \sigma(e_2), \ldots, \sigma(e_n))$ and has cost $\tau(\boldsymbol{p}) = \sum_{i=1}^{n} \tau(e_i)$.*

Definition 5. *Let $G = (V, E, \sigma, \tau)$ be a costly constrained graph. G is called a non-deterministic graph if for some vertex there are two outgoing edges with the same label. Furthermore, a graph is said to be lossy if there exist two distinct paths $\boldsymbol{p}_1$ and $\boldsymbol{p}_2$ in G with the same starting and terminating vertices that satisfy $\sigma(\boldsymbol{p}_1) = \sigma(\boldsymbol{p}_2)$. Otherwise, G is a lossless graph.*

Definition 6. *Given a strongly connected graph $G = (V, E, \sigma, \tau)$ and state $v \in V$, define $N_G(v, t)$ to be the number of distinct sequences generated by paths of cost at most t outgoing from vertex v. Then the capacity of the graph G is defined as*

$C_G \triangleq \limsup_{t\to\infty} \frac{\log_2 N_G(v,t)}{t}$, *where v is an arbitrary vertex in the graph (as the same limit is obtained for all of the vertices).*

Problem 4 Let $G=(V,E,\sigma,\tau)$ be a non-deterministic costly constrained graph. Calculate C_G.

III. Unfixed Synthesis Sequence - Problem 1

This section tackles Problem 1, where the naive solution to this problem follows from Observation 1 and suggests to first construct for every $\boldsymbol{y}^i \in S$ the set $T(\boldsymbol{y}^i) \triangleq \{\boldsymbol{z}\in B^* : G(\boldsymbol{z}) = \boldsymbol{y}^i\}$, then to find an arbitrary sequence in $SCS\{\boldsymbol{z}^1,\dots,\boldsymbol{z}^m\}$ for every tuple $(\boldsymbol{z}^1,\dots,\boldsymbol{z}^m) \in T(\boldsymbol{y}^1)\times\cdots\times T(\boldsymbol{y}^m)$. A shortest SCS found for a tuple belongs to the set $F(S,B)$. When B provides a uniquely decodable presentation, $|T(\boldsymbol{y}^i)| \leqslant 1$ for every $\boldsymbol{y}^i\in S$, thereby Problem 1 translates to finding (once) an SCS of m sequences over B. In contrast, B providing a non-uniquely decodable presentation might incur an unfeasible runtime complexity due to a large number of tuples to consider. Therefore, and as a first step to optimize this algorithm one may ask whether it suffices to consider tuples $(\boldsymbol{z}^1,\dots,\boldsymbol{z}^m) \in F(\{\boldsymbol{y}^1\},B)\times\cdots\times F(\{\boldsymbol{y}^m\},B)$, i.e., does an SCS of the optimal sequences to synthesize separately each strand in S necessarily belong to $F(S,B)$? Example 2 answers this question negatively, for $F(\{\boldsymbol{y}^2\},B) = \{\boldsymbol{s}^6 = (AT,AT)\}, F(\{\boldsymbol{y}^3\},B) = \{\boldsymbol{s}^7 = (C,A,AT)\}$, and $F(\{\boldsymbol{y}^4\},B) = \{\boldsymbol{s}^8 = (C,T,AT)\}$. Hence the sequences in $SCS(\{\boldsymbol{s}^6,\boldsymbol{s}^7,\boldsymbol{s}^8\})$ are of length 5, where those in $F(S,B)$ are of length 4. Second, to solve Problem 1, note that applying the greedy algorithm to synthesize $\{\boldsymbol{y}^1\}$ from Example 2, i.e., using at each step the longest possible shortmer, results in the synthesis sequence $\boldsymbol{s}^2$. However, $\boldsymbol{s}^2 \notin F(\{\boldsymbol{y}^1\},B)$, which exhibits the flaw of the greedy algorithm. Subsection III-B resolves these difficulties by providing a solution to Problem 1, which follows the notations of Definition 2 and uses the graph $G(S,B)$ defined in Subsection III-A.

A. Definition of the Graph $G(S,B)$

Definition 7. *Let Σ, B, and S be as given in Definition 2. Define $G(S,B)=(V,E,\sigma,\tau)$ to be a constrained graph in which*

$$V = \{(i_1,i_2,\dots,i_m) : \forall j\in[m], i_j \in [n_j+1]\}$$

and an edge $((i_1,i_2,\dots,i_m),(i'_1,i'_2,\dots,i'_m))$ of label $\boldsymbol{x}^h\in B$ is added if $(i_1,i_2,\dots,i_m)\neq(i'_1,i'_2,\dots,i'_m)$ and the following conditions hold for every $j\in[m]$.

- $i'_j \in \{i_j, i_j+\ell_h\}$.
- *If $i'_j = i_j+\ell_h$, then $\boldsymbol{y}^j_{[i_j,i_j+\ell_h-1]} = \boldsymbol{x}^h$.*

The function τ sets cost zero to all of the edges in $G(S,B)$.

The following example illustrates the construction of the graph $G(S,B)$.

Example 3 Let $\Sigma=\{A,C,G,T\}, B=\{A,C,AC,G,T\}$, and $S=\{\boldsymbol{y}^1=AC,\boldsymbol{y}^2=AAC\}$. Then, the graph $G(S,B)$ is depicted in Figure 1. Note that in this case $F(S,B)=\{(A,AC)\}$.

B. Solution to Problem 1

The key idea of Algorithm 1 is using the vertices of $G(S,B)$ to represent the intermediate states of the synthesis process. Namely, each vertex is a set of indices where index i_j points to the first symbol in $\boldsymbol{y}^j$ that has not been synthesized yet, and n_j+1 means that the whole sequence $\boldsymbol{y}^j$ was synthesized. For example, $(1,1,\dots,1)$ represents the initial state in which all of the strands should

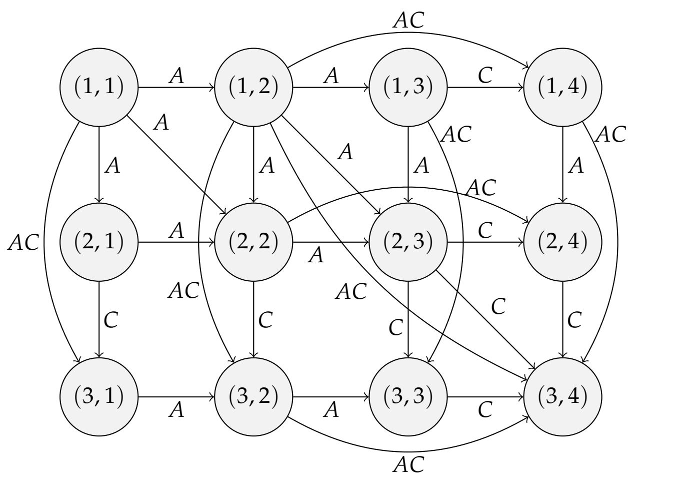


Fig. 1: Graph $G(S,B)$ when $\Sigma=\{A,C,G,T\}, B=\{A,C,AC,G,T\}$, and $S=\{\boldsymbol{y}^1=AC,\boldsymbol{y}^2=AAC\}$.

be synthesized, and $(n_1+1,n_2+1,\dots,n_m+1)$ indicates that all of the strands have been synthesized. An edge $((i_1,i_2,\dots,i_m),(i'_1,i'_2,\dots,i'_m))$ with label $\boldsymbol{x}^h$ implies that the sequences $\left\{\boldsymbol{y}^j_{[i_j,n_j]} : i_j\neq i'_j\right\}$ begin with $\boldsymbol{x}^h$, hence the corresponding strands can be expanded by appending the shortmer $\boldsymbol{x}^h$ in the next cycle. As a result, a length-k path in $G(S,B)$ starting at $(1,1,\dots,1)$ and terminating at $(n_1+1,n_2+1,\dots,n_m+1)$ represents a length-k synthesis sequence that enables synthesizing all of the strands in parallel. Therefore, Algorithm 1 produces an optimal sequence by finding a shortest path from $(1,1,\dots,1)$ to $(n_1+1,n_2+1,\dots,n_m+1)$. For instance, in continuation of Example 3, the shortest path $(1,1)\xrightarrow{A}(1,2)\xrightarrow{AC}(3,4)$ in $G(S,B)$ represents the optimal synthesis sequence (A,AC). Next, the Breadth-first Search algorithm (Chapter 22.2 in [6]) is used to find a shortest path between two vertices.

Algorithm 1 Finding an arbitrary sequence in $F(S,B)$.

```
Input: Alphabet $\Sigma$ and sets $S, B \subseteq \Sigma^*$.
Output: Arbitrary sequence in $F(S,B)$.
1: Construct the graph $G(S,B)$.
2: Using BFS, find a shortest path $\boldsymbol{p}$ from $(1,1,\dots,1)$ to
   $(n_1+1,n_2+1,\dots,n_m+1)$ in $G(S,B)$.
3: return The concatenation (over $B$) of $\boldsymbol{p}$'s labels.
```

Observe that $G(S,B)$ does not contain self loops. In addition, from vertex $(1,1)$ in Figure 3b, there are 3 outgoing edges with label A, though a reader might expect to have only the edge $(1,1)\xrightarrow{A}(2,2)$. Due to the lack of space we do not elaborate on the importance of this construction, yet it is exhibited in Example 3 as the shortest path traverses the edge $(1,1)\xrightarrow{A}(1,2)$. Interestingly, Algorithm 1 imitates (and extends) the classic SCS dynamic programming algorithm [8]. More specifically, Algorithm 1 is equivalent to the SCS dynamic programming algorithm when $B=\Sigma$. Lastly, the following theorem concludes the solution to Problem 1.

Theorem 8. *Let Σ, B, and S be as given in Definition 2. Then Algorithm 1 outputs an arbitrary sequence in $F(S,B)$ in runtime complexity $O\left(b2^m\Pi_{j=1}^m(n_j+1)\right)$.*

Proof: The proof follows from the construction of the graph $G(S,B)$ and the runtime complexity of BFS, as $|V|\leqslant \Pi_{j=1}^m(n_j+1)$ and $|E|\leqslant |V|b2^m$. ■

Note that an extended dynamic programming algorithm searches for the shortest path from $(1,1,\dots,1)$ to $(n_1+1,n_2+1,\dots,n_m+1)$ by starting from vertex

$(n_1+1, n_2+1, \dots, n_m+1)$ and each time checking at most $b2^m$ previous vertices (just as we have three edges outgoing from state $(1,1)$ with label A in Figure 1). Hence, both of the BFS and the dynamic programming algorithm have the same runtime complexity.

IV. Fixed Synthesis Sequence - Problems 2 and 3

In the following, Subsections IV-A and IV-B address Problems 2 and 3 respectively.

A. Solution to Problem 2

The following theorem solves Problem 2.

Theorem 9. *Following the notations of Problem 2 the maximum information rate is attained when the b shortmers provide a uniquely decodable presentation and $\boldsymbol{s}$ is a periodic alternating sequence over B. In this case, by Theorem 3 in [11], $F_1(\Sigma, b) = -\log z_b$ where z_b is the largest root of the polynomial $\sum_{i=1}^{b} z^i - 1$.*

For example, given $\Sigma = \{0,1\}$ and $b = 3$, the maximum information rate $F_1(\Sigma, b)$ is attained when $B = B' \triangleq \{0, 10, 11\}$ and $\boldsymbol{s} = \boldsymbol{s}' \triangleq (0, 10, 11, 0, 10, 11, \dots)$. That is, $F_1(\Sigma, b) = R_{\boldsymbol{s}'}$ and for every B of size 3 and $\boldsymbol{s} \in B^*$, $R_{\boldsymbol{s}} \leqslant R_{\boldsymbol{s}'}$. Moreover, as explained in Section V, since B' provides a uniquely decodable presentation, $R_{\boldsymbol{s}'}$ can be calculated using the techniques of [11] and [12], specifically by Theorem 3 in [11].

B. Partial Solution to Problem 3

First note that B that includes all of Σ provides a non-uniquely decodable presentation, thereby, Theorem 9 does not ensure such B attains the maximum information rate with some synthesis sequence. Moreover, as mentioned in Section V, it is still unclear how to derive the information rate in this case. Therefore, Theorem 10 tackles this case and provides a partial solution to Problem 3 when $k = 1$.

Theorem 10. *Let Σ be as given in Definition 2 and $\ell, t \in \mathbb{N}$ be such that $\ell > q+1$ and $q+1 | t$. For $\boldsymbol{x} \in \Sigma^{\ell}$, define $\boldsymbol{s}_{\boldsymbol{x}} = (\sigma_1, \sigma_2, \dots, \sigma_q, \boldsymbol{x}, \sigma_1, \sigma_2, \dots, \sigma_q, \boldsymbol{x}, \dots, \sigma_1, \sigma_2, \dots, \sigma_q, \boldsymbol{x})$ to be a length-t periodic alternating sequence. In addition, define $M_1 = \left\{\sigma_i^k \sigma_{i-1}^{\ell-k} : 2 \leqslant i \leqslant q, 1 \leqslant k \leqslant \ell-1\right\}$, $M_2 = \left\{\sigma_i^{\ell} : 1 \leqslant i \leqslant q\right\}$, and $M_3 = \left\{\sigma_1^k \sigma_q^{\ell-k} : 1 \leqslant k \leqslant \ell-1\right\}$. Then, for sufficiently large t, $\arg\max_{\boldsymbol{x} \in \Sigma^{\ell}} N_{\boldsymbol{s}_{\boldsymbol{x}}}(t) \in M_1 \cup M_2 \cup M_3$. Moreover, for every $\boldsymbol{u}, \boldsymbol{u}' \in M_1$, $N_{\boldsymbol{s}_{\boldsymbol{u}}}(t) = N_{\boldsymbol{s}_{\boldsymbol{u}'}}(t)$, and for every $\boldsymbol{v}, \boldsymbol{v}' \in M_3$, $N_{\boldsymbol{s}_{\boldsymbol{v}}}(t) = N_{\boldsymbol{s}_{\boldsymbol{v}'}}(t)$.*

For example, to find the largest attainable information rate by a synthesis sequence $\boldsymbol{s}_{\boldsymbol{x}}$, it suffices to check all of the q sequences in M_2 in addition to two arbitrary sequences, one taken from M_1 while the other from M_3. We leave studying Problem 3 for larger values of k for future work. Meanwhile we expect the following approach to provide shortmers $\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_k$ with a large information rate. Choose the k shortmers so that each concatenation $\boldsymbol{x}_{i_1}\boldsymbol{x}_{i_2}\cdots\boldsymbol{x}_{i_h}$ where $1 \leqslant i_1 < i_2 < \cdots < i_h \leqslant k$ and $h \in [1,k]$ requires relatively a long prefix of $(\sigma_1, \sigma_2, \dots, \sigma_q, \sigma_1, \sigma_2, \dots, \sigma_q, \dots)$ to synthesize. For example, when $\Sigma = \{A, C, G, T\}$ and $k = 2$, we expect the shortmers $\boldsymbol{x}_1 = TG$ and $\boldsymbol{x}_2 = CA$ to yield the maximum possible information rate.

V. Calculating Capacities of Costly Constrained Graphs - Problem 4

This section focuses on Problem 4. First, Subsection V-A reviews some ideas from [11] and [12] to motivate studying this problem. Then, Subsection V-B derives bounds on the capacity. Throughout this section, given a costly constrained graph G, $\mathcal{P}_G(v,t)$ denotes the set of all paths of cost at most t outgoing from vertex v in G, and $P_G(v,t) = |\mathcal{P}_G(v,t)|$.

A. Connection between the Synthesis Problem and Costly Constrained Graphs

In [11] and [12] the information rate when $B = \Sigma$ was calculated by computing the capacities of some costly constrained graphs referred to as the *subsequence graphs*. The following applies this approach when $B \subseteq \Sigma^*$ too and the main result of this subsection is summarized in Theorem 12. However, for the sake of completeness we bring again the definition of the subsequence graphs as presented in [11] and [12] and illustrate the theorem in Example 4. Note that the construction of the subsequence graphs abuses Definition 4 by assigning sequences (and not only symbols) as labels. Nevertheless, $\sigma(\boldsymbol{p})$ given a path $\boldsymbol{p}$ is still seen as a sequence over Σ, and $N_G(v,t)$ still counts distinct sequences over Σ (and not over B).

Definition 11. *Let Σ and B be as in Definition 2. Moreover, let $\boldsymbol{s} = \boldsymbol{rrr}\cdots$ be a semi-infinite periodic synthesis sequence of seed $\boldsymbol{r} = (\boldsymbol{r}_1, \dots, \boldsymbol{r}_{\ell}) \in B^{\ell}$. Define the subsequence graph of $\boldsymbol{r}$, $G(\boldsymbol{r}) = (V, E, \sigma, \tau)$ as follows. V consists of ℓ vertices, where v_i is associated with the shortmer $\boldsymbol{r}_i$. An edge $e = (v_i, v_j)$ is added if either $i < j$ and $\boldsymbol{r}_k \neq \boldsymbol{r}_j$ for all $k \in \{i+1, \dots, j-1\}$, or $i \geqslant j$ and $\boldsymbol{r}_k \neq \boldsymbol{r}_j$ for all $k \in \{i+1, \dots, \ell, 1, \dots, j-1\}$. The label of such an edge is $\sigma(e) = \boldsymbol{r}_j$ and the cost is*

$$\tau(e) = \begin{cases} j - i, & i < j \\ \ell - i + j, & i \geqslant j \end{cases}.$$

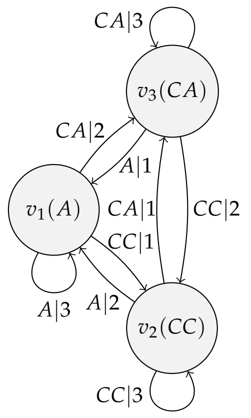
Let $e = (v_i, v_j)$ be an edge of label $\boldsymbol{r}_j$ and cost τ in the graph $G(\boldsymbol{r})$ constructed in Definition 11. Then, τ is the minimum number of cycles that should be skipped in order to synthesize the shortmer $\boldsymbol{r}_j$ after a cycle of $\boldsymbol{r}_i$.

Example 4 Let $\Sigma = \{A, C, G, T\}, B = \{A, CC, CA\}, B' = \{A, C, AC\}, \boldsymbol{r} = (A, CC, CA)$, and $\boldsymbol{r}' = (A, C, AC)$. Moreover, let $\boldsymbol{s} = (A, CC, CA, A, CC, CA, \dots), \boldsymbol{s}' = (A, C, AC, A, C, AC, \dots)$, be the semi-infinite periodic synthesis sequence of *seed* $\boldsymbol{r}, \boldsymbol{r}'$, respectively. The subsequence graph $G(\boldsymbol{r}), G(\boldsymbol{r}')$ is depicted in Figure 2a, 2b, respectively. Let $\boldsymbol{p}$ be a path of cost t outgoing from vertex v_3 in $G(\boldsymbol{r})$. Then $\sigma(\boldsymbol{p})$ can be synthesized using $\boldsymbol{s}_{[1,t]}$. For instance, $\sigma\left(v_3 \xrightarrow{A|1} v_1 \xrightarrow{A|3} v_1\right) = (A, A)$ can be synthesized using the first 4 cycles of $\boldsymbol{s}$. In summary, $N_{\boldsymbol{s}}(t) = N_{G(\boldsymbol{r})}(v_3, t)$, and consequently $R_{\boldsymbol{s}} = C_{G(\boldsymbol{r})}$. The same holds for $\boldsymbol{s}'$ too. Since B is a uniquely decodable code, then distinct paths outgoing from the same vertex generate distinct sequences, which means, $N_{G(\boldsymbol{r})}(v_3, t) = P_{G(\boldsymbol{r})}(v_3, t)$. In fact, one can consider the labels of $G(\boldsymbol{r})$ as symbols over the alphabet B. In this case $G(\boldsymbol{r})$ is a deterministic graph and $C_{G(\boldsymbol{r})}$ can be calculated just as in [9], [12] and [14]. In contrast, B' is a non-uniquely decodable code. As a result, some sequences over Σ, such as (A, C), may be generated by distinct paths outgoing from vertex v_3. Mathematically speaking, $N_{G(\boldsymbol{r}')}(v_3, t) < P_{G(\boldsymbol{r}')}(v_3, t)$. Existing literature reveals how to calculate the value of $P_{G(\boldsymbol{r}')}(v_3, t)$, yet not $N_{G(\boldsymbol{r}')}(v_3, t)$ [9], [12], [14]. In other words, if we look at the sequences generated in $G(\boldsymbol{r}')$ over Σ, then $G(\boldsymbol{r}')$ is a lossy graph and

$$R_{\boldsymbol{s}'} = C_{G(\boldsymbol{r}')} \leqslant \limsup_{t\to\infty} \frac{\log_2 P_G(v,t)}{t},$$

hence the motivation to study Problem 4.

Lastly Theorem 12 generalizes the observations of Example 4 to any semi-infinite periodic synthesis sequence. It even extends the study of [11] and [12] to the case of $B \subseteq \Sigma^*$.


(a) Graph $G(\boldsymbol{r})$.

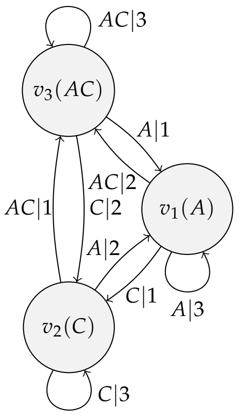

(b) Graph $G(\boldsymbol{r}')$.

Fig. 2: Subsequence graphs mentioned in Example 4.

Theorem 12. *Let $\boldsymbol{s}$ be a semi-infinite periodic synthesis sequence of seed $\boldsymbol{r}$ and let v be an arbitrary vertex in $G(\boldsymbol{r})$. Then,*

$$R_{\boldsymbol{s}} = C_{G(\boldsymbol{r})} \overset{(a)}{\leqslant} \limsup_{t\to\infty} \frac{\log_2 P_{G(\boldsymbol{r})}(v,t)}{t},$$

where inequality (a) holds with equality when B provides a uniquely decodable presentation.

B. Calculating the Capacity of Non-deterministic Costly Constrained Graphs

To tackle Problem 4, first note that the following lemmas are immediate.

Lemma 13. *Let G be a costly constrained graph, then $C_G \leqslant \limsup_{t\to\infty} \frac{\log_2 P_G(v,t)}{t}$, where equality holds when G is a lossless graph.*

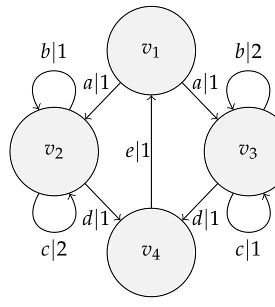
Note that $\limsup_{t\to\infty} \frac{\log_2 P_G(v,t)}{t}$ can be calculated following existing literature [9], [12], [14]. Moreover, the proof of Lemma 13 imitates the proof of Lemma 3.6 in [14], as the lossless property implies that $\frac{P_G(v,t)}{|V|} \leqslant N_G(v,t) \leqslant P_G(v,t)$.

Lemma 14. *Let $G = (V,E,\sigma,\tau)$ be a costly constrained graph. It τ sets the same nonzero cost to all of the edges, then C_G can be derived by constructing a determinizing graph as explained in Section 2.2.1 in [14] and calculating its capacity.*

Recall that a determinizing graph defined in Section 2.2.1 in [14] is a deterministic graph that generates the same constrained system as the original graph. To calculate C_G one may try to apply this approach even when τ is not a constant function. Example 5 reveals the difficulty of such application and the bounds that may be established nonetheless.

Example 5 Let $\Sigma = \{a,b,c,d\}$. In addition, let $G = (V,E,\sigma,\tau)$ be the graph depicted in Figure 3a. Constructing the determinizing graph of G, G' as suggested in Section 2.2.1 in [14] results in the graph depicted in Figure 3b. As marked in this graph, it is not clear which cost to assign to the self loops of $\{v_2,v_3\}$. However, the capacity of the graph obtained by assigning the minimum, maximum possible cost $1,2$ to each self loop upper, lower bounds C_G, respectively.

The following theorem generalizes the conclusion of Example 5. Let $G = (V,E,\sigma,\tau)$ be a non-deterministic costly constrained graph, and let $G' = (V',E',\sigma')$ be the determinizing graph of G constructed (while ignoring τ) as defined in Section 2.2.1 in [14]. Note that G' is an unweighted graph, for [14] defines determinizing graphs for unweighted ones. Furthermore, recall that the vertices of G' are subsets of vertices in G. Thus, we denote the vertices of G' by capital letters.


(a) Graph G.

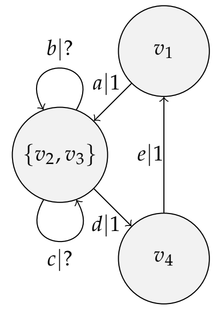

(b) Graph G'.

Fig. 3: Determinizing graph when τ is not constant.

In addition, recall that an edge $(U,U') \in E'$ with label σ implies that each vertex of G in U' is accessible by an edge of label σ from some vertex in U. Next, let G^+, G^- be the graph obtained by setting the following cost function τ^+,τ^- to the graph G', respectively. That is, $G^+ = (V',E',\sigma',\tau^+)$ and $G^- = (V',E',\sigma',\tau^-)$.

$$\tau^+((U,U')) = \max_{e=(u,u')\in E: u\in U, u'\in U', \sigma(e)=\sigma'((U,U'))} \tau(e).$$
$$\tau^-((U,U')) = \min_{e=(u,u')\in E: u\in U, u'\in U', \sigma(e)=\sigma'((U,U'))} \tau(e).$$

Theorem 15. *Let $G = (V,E,\sigma,\tau)$ be a non-deterministic costly constrained graph. Then, $C_{G^+} \leqslant C_G \leqslant C_{G^-}$.*

Lastly, Theorem 17 follows the analysis of [18] to establish a lower bound on C_G. The key idea of Theorem 17 is constructing a graph of $P_G(v,t)$ vertices, where each vertex represents a path in $\mathcal{P}_G(v,t)$, and connecting two vertices if the represented paths generate distinct sequences. Then $N_G(v,t)$ equals the size of the largest clique in the graph, which motivates employing Turan's theorem as in [18]. The following definition is used in the statement of Theorem 17.

Definition 16. *Let $G = (V,E,\sigma,\tau)$ be a costly constrained graph. Given a path $\boldsymbol{p} \in \mathcal{P}_G(v,t)$, define $P_G(v,t,\boldsymbol{p})$ to be the number of all distinct paths outgoing from vertex v of cost at most t that generate the same sequence as path $\boldsymbol{p}$. Moreover, define*

$$P_G^{avg}(v,t) \triangleq \mathbb{E}[P_G(v,t,\boldsymbol{p})] = \frac{\sum_{\boldsymbol{p}\in\mathcal{P}_G(v,t)} P_G(v,t,\boldsymbol{p})}{P_G(v,t)}.$$

Theorem 17. *Let $G = (V,E,\sigma,\tau)$ be a costly constrained graph and let v be an arbitrary vertex in G. Then,*

$$C_G \geqslant \limsup_{t\to\infty} \frac{\log_2 P_G(v,t)}{t} - \frac{\log_2 P_G^{avg}(v,t)}{t}.$$

For example, consider the graph G depicted in Figure 3a and let $\boldsymbol{p} \in \mathcal{P}_G(v_1,t)$. $\boldsymbol{p}$ traverses the edge (v_4,v_1) at most $\frac{t}{3}$ times, thereby $P_G(v,t,\boldsymbol{p}) \leqslant 2^{\frac{t}{3}}$. Hence, $P_G^{avg}(v_1,t) \leqslant 2^{\frac{t}{3}}$ and by Theorem 17, $C_G \geqslant \limsup_{t\to\infty} \frac{\log_2 P_G(v,t)}{t} - \frac{1}{3} = \frac{2}{3}$. In summary, by Lemma 13 and Theorem 17, $\frac{2}{3} \leqslant C_G \leqslant 1$. However, $P_G^{avg}(v_1,t)$ is expected to be smaller than $2^{\frac{t}{3}}$, for most of the paths in $\mathcal{P}_G(v_1,t)$ traverse the edge (v_4,v_1) fewer times. Due to the lack of space we leave improving the lower bound for future work.

Acknowledgment

The authors would like to thank Professor Ron M. Roth and Professor Paul H. Siegel for the helpful discussions on the capacity of non-deterministic costly constrained graphs. The research was Funded by the European Union (ERC, DNAStorage, 865630). Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the EuropeanResearch Council Executive Agency. Neither the European Union nor the granting authority can be held responsible for them.

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 Biotechnology*, vol. 37, no. 10, pp. 1229–1236, Oct. 2019.

[2] L. Anavy, Z. Yakhini, and R. Amit, 2021. “Molecular data storage systems and methods.” *United States of America Patent US20210141568A1.*

[3] M. H. Caruthers, “The chemical synthesis of DNA/RNA: Our gift to science,” *Journal of Biological Chemistry*, vol. 288, no. 2, pp. 1420–1427, Jan. 2013.

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

[5] Y. Choi, T. Ryu, A. C. Lee, H. Choi, H. Lee, J. Park, S. H. Song, S. Kim, H. Kim, W. Park, and S. Kwon, “High information capacity DNA-based data storage with augmented encoding characters using degenerate bases,” *Scientific Reports*, vol. 9, no. 1, pp. 1–7, 2019.

[6] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, “Introduction to algorithms,” *Mit Press*, 2001.

[7] O. Elishco and W. Huleihel, “Optimal reference for DNA synthesis,” *arXiv preprint arXiv:2204.07013*, 2022.

[8] S. Y. Itoga, “The string merging problem,” *BIT Numerical Mathematics*, vol. 21, no. 1, pp. 20–30, 1981.

[9] A. Khandekar, R. McEliece, and E. Rodemich, “The discrete noiseless channel revisited,” *Coding, Communications, and Broadcasting*, pp. 115–137, 2000.

[10] 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.

[11] A. Lenz, Yi. Liu, C. Rashtchian, P. H. Siegel, A. Wachter-Zeh, and E. Yaakobi, “Coding for efficient DNA synthesis,” *IEEE International Symposium on Information Theory*, pp. 2885–2890, Los Angeles, CA, USA, June 2020.

[12] A. Lenz, S. Melczer, C. Rashtchian, and P. H. Siegel “Multivariate analytic combinatorics for cost constrained channels and subsequence enumeration,” *https://arxiv.org/abs/2111.06105*, 2021.

[13] K. Makarychev, M. Z. Rácz, C. Rashtchian, and S. Yekhanin, “Batch optimization for DNA synthesis,” *IEEE International Symposium on Information Theory*, pp. 1949–1954, Melbourne, Victoria, Australia, July 2021.

[14] 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

[15] I. Preuss, Z. Yakhini, and L. Anavy, “Data storage based on combinatorial synthesis of DNA shortmers,” *bioRxiv*, 2021.

[16] N. Roquet, H. Park, and S.P. Bhatia, 2017, “Nucleic acid-based data storage.” *United States Patent Application Patent 20180137418.*

[17] N. Roquet, S. P. Bhatia, S. A. Flickinger, S. Mihm, M. W. Norsworthy, D. Leake, and H. Park, “DNA-based data storage via combinatorial assembly,” *bioRxiv*, 2021.

[18] L. M. G. M. Tolhuizen, “The generalized Gilbert-Varshamov bound is implied by Turan’s theorem,” *IEEE Transactions on Information Theory*, vol. 43, no. 5, pp. 1605–1606, Sep. 1997.