Automaton learning methods based on the decomposition of a Hankel matrix for a DNA information storage application

M1 research project supervised by Romaric Gaudel and Dominique Lavenier

Gauvain Thomas

1 Introduction

Rapid advances in technology have led to an increasing need for efficient and reliable methods of data storage. One emerging approach is the use of DNA as a data storage medium due to its high storage density and long-term stability. However, encoding and retrieving the information encoded in DNA sequences poses significant challenges, as both writing and reading of sequences can introduce noise, resulting in errors in the sequences. The aim of this research is to develop a method to accurately retrieve the consensus sequence from noisy copies.

Various methods have been proposed to address the challenge of recovering the consensus sequence from noisy DNA sequences. One prominent approach involves the use of error-correcting codes to mitigate the effects of insertions, deletions and substitutions during the sequencing process. In this study, we present an alternative approach that uses probabilistic automata to model the underlying probability distribution of DNA sequences.

The unknown DNA sequence is characterised by random insertions, deletions and substitutions, which together define a probability distribution over words. This probability distribution can be represented by a Probabilistic Finite Automaton (PFA), which captures the stochastic nature of the sequence generation process. However, recent learning approaches do not necessarily produce interpretable models, making it difficult to analyse and understand the learned probability distributions. To overcome this limitation, we restrict the model to a subclass of probabilistic automata, Probabilistic Residual Finite Automata (PRFA), from which we can extract meaningful interpretations.

This article provides an overview of existing related work in the field of DNA storage, focusing on different approaches used to address the challenges associated with encoding and reading DNA sequences. We present the necessary background information to understand our chosen method and its application in this specific domain. Furthermore, we perform extensive experimental evaluations to assess the effectiveness of our method. However, our results show that the method used did not provide satisfactory results in our particular case.

2 Related Work

In recent years, the field of DNA data storage has had significant attention, leading to the development of various techniques and approaches aimed at addressing the challenges associated with reliable and efficient data retrieval from DNA sequences. In this section, we provide an overview of some notable research papers that have contributed to the state-of-the-art in DNA data storage.

[5] presents a high density and reliable method to recover information in synthetic DNA when very few copies are stored.

[3] presents a comprehensive approach to implement DNA-based storage systems. Their method utilizes Deep Neural Networks (DNN) to reconstruct sequences based on imperfect clusters of copies generated during the synthesis and sequencing processes. To combat the patterns of errors, Error-Correcting Code (ECC) is employed.

[7] proposes a modulation-based DNA storage architecture that ensures encoded DNA sequences follow biological constraints and enables efficient and robust error detection, addressing the challenge of reliable data recovery in synthetic DNA storage.

3 Background

3.1 Data generation

The process of storing data in DNA sequences involves two main steps: writing the DNA sequences and then reading them. Errors can occur in both steps, including insertions, deletions and substitutions of characters. Individually, these errors are usually quite small (less than 5%), but for longer words, the probability of recovering the exact consensus sequence quickly approaches zero.

To illustrate this, let's consider a word denoted as $w \in \Sigma^\star$, where Σ represents the alphabet (in this case, $\Sigma = \{A, C, G, T\}$). The length of the word is given by $|w| = L \in \mathbf{N}^\star$. During the writing process, several noisy copies of the word w can be produced. As each letter of w is written, there are possibilities for that letter to remain unchanged, to be replaced by another letter, to be removed entirely, or to have another letter inserted after it. These variations contribute to the potential errors introduced during the writing process. This process thus defines a probability distribution over words, such that given a word to write w, the word $\hat{w}$ is instead written with a certain probability $p(w)$, and this distribution can be described using a probabilistic automaton, which we'll define in the next section.

The reading process can be modelled in a similar way to the writing process. Therefore, when we combine successive writing and reading operations, additional errors may accumulate, resulting in more complex variations beyond simple insertions, deletions and substitutions. For the sake of simplicity, however, we will focus on only one of these processes for the remainder of this paper and assume that the other process is error-free.

In the next sections, we will focus on the definition of a Probabilistic Finite Automaton (PFA) that can effectively model this process, along with a learning method to construct the automaton from a given sample. However, we will observe that the automaton learned by this method does not inherently capture the probabilistic nature, making it difficult to retrieve the consensus sequence. Therefore, we will introduce a specific subclass of PFA and a corresponding learning algorithm that allows us to recover the sequence from the learned automaton.

3.2 Learing Probabilistic Finite Automaton (PFA)

3.2.1 PFA

Definition : A PFA [2] is a tuple $\langle \Sigma, Q, \{A_o\}_{o\in\Sigma}, \boldsymbol{\alpha}_0, \boldsymbol{\alpha}_\infty \rangle$ where Σ is an alphabet and Q is a finite set of states. Matrices $A_o \in \mathbf{R}^{+|Q|\times|Q|}$ contain the transition weights. The vectors $\alpha_\infty \in \mathbf{R}^{+|Q|}$ and $\alpha_0 \in \mathbf{R}^{+|Q|}$ contain respectively the terminal and initial weights. These weights should verify,

$$\mathbf{1}^\top \boldsymbol{\alpha}_0 = 1 \qquad \boldsymbol{\alpha}_\infty + \sum_{o\in\Sigma} A_o \mathbf{1} = \mathbf{1} \tag{1}$$

We also define $\forall w = o_1 ... o_n \in \Sigma^\star, A_w \coloneqq A_{o_1} ... A_{o_n}$. In this case with

Such a process defines a function $f : \Sigma^\star \to \mathbf{R}$, such that $\forall w \in \Sigma^\star, f(w) = \boldsymbol{\alpha}_0 A_w \boldsymbol{\alpha}_\infty$. With the addition of constraints (1), f defines a probability distribution on $\Sigma^\star$.

We call Multiplicity Automaton (MA) a PFA without constraints (1) on the weights. Thus the weights can be negative, which looses the probabilistic meaning of the PFA.

3.2.2 Spectral learning

We use the Hankel matrix representation $H \in \mathbf{R}^{\Sigma^\star \times \Sigma^\star}$ of a function $f : \Sigma^\star \to \mathbf{R}$. H is a bi-infinite matrix with rows and columns indexed by $\Sigma^\star$, such that $H(u, v) = f(uv)$. In the case of PFA, f defines a probability distribution on $\Sigma^\star$ and can be estimated from samples by counting occurrences of sequences. We call $\hat{f}$, $\hat{H}$ all subsequent estimators. Thus H can be estimated by counting the occurrences of sequences from a sample and normalizing them. We also define $H_o \in \mathbf{R}^{\Sigma^\star \times \Sigma^\star}$ such that $H_o(u, v) = f(uov)$ and $\boldsymbol{h} \in \mathbf{R}^{\Sigma^\star}$ such that $\boldsymbol{h}(u) = f(u)$.

For a MA $\langle \Sigma, Q, \{A_o\}_{o\in\Sigma}, \boldsymbol{\alpha}_0, \boldsymbol{\alpha}_\infty \rangle$ with n states, we have for $u, v \in \Sigma^\star$:

$$H(u, v) = (\boldsymbol{\alpha}_0 A_u)^\top (A_v \boldsymbol{\alpha}_\infty)$$

. We then define the matrices :

$$P = ((\boldsymbol{\alpha}_0^\top A_u)^\top)_{u\in\Sigma^\star}^\top, \qquad S = (A_v \boldsymbol{\alpha}_\infty)_{v\in\Sigma^\star},$$

and see that $H = PS$.

We also have :

$$H_o = PA_oS, \quad \boldsymbol{h}^\top = \boldsymbol{\alpha}_0^\top, \quad \boldsymbol{h} = P\boldsymbol{\alpha}_\infty$$

We can then solve these equations using a truncated Singular Value Decomposition (SVD) of a sub-block $H_\mathcal{B}$ of H, according to a basis $\mathcal{B} = (\mathcal{P}, \mathcal{S})$ of prefixes and suffixes. We have the estimate $\hat{H}_\mathcal{B} \approx \hat{U}\hat{D}\hat{V}^\top$

Thus,

$$\begin{aligned}
\hat{A}_o &= \hat{D}^{-1}\hat{U}^\top \hat{H}^o_\mathcal{B}\hat{V}, \\
\hat{\boldsymbol{\alpha}_0}^\top &= \hat{\boldsymbol{h}}^\top \hat{V}, \\
\hat{\boldsymbol{\alpha}_\infty} &= \hat{D}^{-1}\hat{U}^\top \hat{\boldsymbol{h}}_\mathcal{P}.
\end{aligned}$$

However, the automaton learned by the spectral algorithm is usually only a MA and therefore not necessarily a PFA. This means that the function $\hat{f}$ described by the MA has no probabilistic meaning, making it difficult to extract a consensus sequence from it. For this reason, we introduce a subclass of PA for which we present a learning algorithm that, this time, actually returns a PA.

3.3 Learning Probabilistic Residual Finite Automata

3.3.1 Probabilistic Residual Finite Automata

We define in this section a subclass of PFA : Probabilistic Residual Finite Automata (PRFA).

First we define the linear operator such that $\forall v \in \Sigma^\star, \dot{u}p(v) = p(uv)$, $\bar{p}(u)$ the probability that a word starts with prefix $u \in \Sigma^\star$, and p_u, the conditional probability of observing v after u. Thus if $\bar{p}(u) > 0$, we have

$$\forall v \in \Sigma^\star, p_u(v) = \frac{\dot{u}p(v)}{\bar{p}(u)}$$

Similarly, we define for $u \in \Sigma^\star$, $p_q(u) = \mathbf{1}_q A_u \boldsymbol{\alpha}_\infty$ the conditional probability of observing u starting from state $q \in Q$ of a PFA $\langle \Sigma, Q, \{A_o\}_{o\in\Sigma}, \boldsymbol{\alpha}_0, \boldsymbol{\alpha}_\infty \rangle$

A PRFA is a PFA $(\Sigma, Q, \alpha_0, A, \alpha_\infty)$ such that for all state $q \in Q$, there exists a word $u \in \Sigma^\star$ such that $\bar{p}(u) > 0$ and $p_q = p_u$.

This means that for each state q, there is a prefix of an accepted word that only ends in state q, which therefore can be used to identify the state.

3.3.2 CH-PRFA algorithm

The CH-PRFA algorithm aims to learn a Probabilistic Finite Automaton (PFA) by decomposing the given distribution under certain constraints. We assume d the minimal number of states of the PRFA, and a basis $(\mathcal{P}, \mathcal{S})$ of prefixes and suffixes (see [4] for conditions on the basis) are provided as an input. We call $\hat{p}_u$ and $\dot{o}\hat{p}_u$ the corresponding estimated probabilities for p_u and $\dot{o}p_u$, as within the previous seciton. The algorithm consists of three main steps:

- Estimating $\hat{\boldsymbol{p}}_u$ and $\dot{o}\hat{\boldsymbol{p}}_u$: This step involves estimating the probabilities $\hat{\boldsymbol{p}}_u$ and $\dot{o}\hat{\boldsymbol{p}}_u$ for each prefix $u \in \mathcal{P}$.
- Compute $\hat{\mathcal{R}}$ a subset of d prefixes in $\mathcal{P}$, using near-separable Non-negative Matrix Factorization (NMF). This set will ultimately characterize the states of the PRFA.
- Optimization under constraints: This step involves retrieving the coefficients of the PFA through linear regressions. Non-negativity and linear constraints are added to ensure that the parameters define a valid PFA. The algorithm employs quadratic optimization problems under linear constraints to solve this optimization task.

The CH-PRFA algorithm first estimates the probabilities for each prefix, then identifies the set $\hat{R}$. The selection of $\hat{R}$ is based on the estimated $\hat{p}_u$ values, which are computed using near-separable Non-negative Matrix Factorization (NMF). In this case, the Successive Projection Algorithm (SPA) is used to compute the NMF. Linear regressions are then used to retrieve the parameters of the PFA. Non-negativity and linear constraints are imposed to ensure the obtained PFA is valid, although it may not necessarily be a PRFA.

The minimization problems in the algorithm can be formulated as quadratic optimization problems under linear constraints with convex costs.

4 Contribution

4.1 DNA sequencing as a probabilistic automaton

Our approach involves representing the sequencing process using a Probabilistic Finite Automaton (PFA). This can be achieved by constructing an automaton with a maximum size of $d = 2L + 1$, where $L = |w|$ corresponds to the length of the DNA sequence. The automaton consists of states representing the actual consensus sequence, along with additional transitions to account for substitutions and deletions, as well as states for modelling the insertions. It's important to note that the automaton only describes one aspect of the storage process. In this paper we assume that the other process is perfect, i.e. only the original sequence is modified, and we do not consider further modifications to the already modified sequence. For example, we can consider the encoding process to be perfect, while the multiple readings of the unmodified sequences introduce noise.

In this section, we present our contribution to the problem of sequencing DNA using a PFA. We propose an approach that makes use of the CH-PRFA algorithm to retrieve the consensus sequence from noisy DNA data. Our contribution builds upon the observation that the probabilities associated with DNA sequencing events, such as substitutions, insertions, and deletions, are typically very small, often less than 5%. Due to these low probabilities, the distribution described by the PFA is close to a PRFA in practice.

The CH-PRFA algorithm, as described in [4], offers a faster learning approach compared to other state-of-the-art algorithms for PRFA inference. Even if the target probability distribution does not precisely correspond to a PRFA, the learned PFA still approximates the original distribution effectively. This algorithm serves as a suitable choice for our sequencing problem, allowing us to recover the consensus sequence efficiently.

4.2 Choosing Prefixes and Suffixes

The selection of appropriate prefixes and suffixes is crucial for the CH-PRFA algorithm, as it assumes a certain basis of prefixes and suffixes as its output. While [4] proposed using the most frequent prefixes and suffixes, we propose an alternative approach that considers the most common prefixes and suffixes for each size. This strategy aims to preserve information about longer prefixes and suffixes, which tend to become increasingly rare as the word size increases. By including the most common prefixes and suffixes of varying sizes in the basis, we can prevent the loss of valuable information in the resulting Hankel matrix.

4.3 Pattern Prefixes and Suffixes

In addition to the conventional prefixes p (e.g. $ACACGCACA \in \Sigma^*$), we introduce pattern prefixes (resp. suffixes) of the form X^4m (e.g. $X^kACAG \in X^\star\Sigma^4$) (resp., mX^4), where X denotes any symbol in the alphabet Σ. These pattern prefixes and suffixes capture subsequences m of length $|m| = l$ ($l = 4$ in all of our experiments), positioned at $k + 1$ within the word. This approach allows us to represent information about a specific subsequence while disregarding the exact start of the word. It becomes particularly useful as longer prefixes and suffixes become increasingly infrequent. By condensing the information of multiple lines in the original Hankel matrix into a few entries, the resulting Hankel matrix is denser while preserving the computational complexity. Indeed, during the estimation of the Hankel matrix, a constant amount of entries are added to the matrix instead of just one, as the size of the subsequences are bounded.

5 Experimental Evaluation

5.1 Generation of the training set

To generate the training samples for our algorithm, we follow a specific procedure. First, we randomly select a consensus sequence of length L. We then generate words by adding errors to this consensus sequence, with small probabilities assigned to each type of error. Throughout our experiments, we use the following probabilities for each step :

- Substitution : 3%
- Insertion : 1%
- Deletion : 1%

These probabilities determine the likelihood of errors being introduced at each letter of the word during the generation process. By incorporating a controlled level of noise, we aim to simulate realistic conditions and evaluate the performance of the algorithm in dealing with errors commonly found in DNA sequences.

5.2 Algorithms

In this experiment, we compare three different methods:

A naive method, which consists of selecting the most frequent word from the generated training set. This method is quite efficient and performs well if L is small enough and the size of the training set is large enough, in which case the most frequent words are almost certain to appear several times in the set. However, in realistic settings, the error probabilities are such that the consensus sequence almost never appears in the training set [1], and most are of incorrect length [6].

Classic CH-PRFA : our first proposed approach, which uses the CH-PRFA algorithm with classic prefixes and suffixes.

CH-PRFA with patterns : an extension of the previous algorithm that extends the set of prefixes and suffixes with the patterns introduced earlier.

5.3 Measures and parameters

In our research, we examine several parameters and error measures to assess the performance of the proposed methods.

The parameter L represents the length of the consensus sequence we are trying to recover. This parameter plays a crucial role in determining the complexity of the problem.

We also consider the training size, which refers to the number of generated noisy words used for the algorithm.

Furthermore, we investigate the effect of the number of prefixes and suffixes included in each set of the basis, denoted as $\mathcal{B} = (\mathcal{P}, \mathcal{S})$. We vary this parameter as a proportion of the training size to examine its influence on the results.

To evaluate the accuracy of the reconstructed sequences, we use two measures. The success rate indicates whether the retrieved word matches the actual consensus sequence. In addition, we use the Levenshtein distance, which quantifies the dissimilarity between the retrieved word and the consensus sequence by taking into account the number of insertions, deletions and substitutions required to transform one into the other.

5.4 Results analysis

In this section, we discuss the obtained results and analyze the performance of the different methods used in our study.

Figure 1 consists of two heatmaps comparing the success rates of the naive method and our proposed method with classic prefixes and suffixes. The heatmaps show the variation in success rates as a function of the length (L) of the consensus sequence and the training size. Figure 2 is the same with pattern prefixes.

The results clearly show a phase transition in the success rates of the naive method. This observation confirms our earlier discussion of the limitations of the naive approach, which simply selects the most frequent word from the generated training set. As expected, the naive method proves to be ineffective in realistic settings where the error probabilities result in the consensus word rarely appearing in the training set. Furthermore, since the generation of each word in the set is unique, there is no longer a single most common word. This phase transition is also visible with classic CH-PRFA, and even more with pattern prefixes.

Interestingly, both of our proposed methods show relatively poor performance compared to the naive method. Although a phase transition is also observed in the success rates of these methods, it is less pronounced compared to the naive method. Further analysis is required to understand the underlying factors contributing to this phenomenon and to identify potential avenues for improvement.

However, both proposed methods in this paper unexpectedly performed poorly, yielding results that fell short of the anticipated outcomes.

In our study, we conducted an investigation into the effect of the number of prefixes and suffixes on the performance of our algorithm. However, the results did not present a clear and definitive conclusion regarding this relationship. While there were some observations that indicated a potential impact, further investigation is required to fully understand and interpret these findings.

One interesting trend that emerged from our analysis was that adding too much prefixes and suffixes to the base $\mathcal{B}$ appeared to slightly lower the success rate of the algorithm. This observation suggests that there might

Figure 1: Sucess rates for naive and classic CH-PRFA

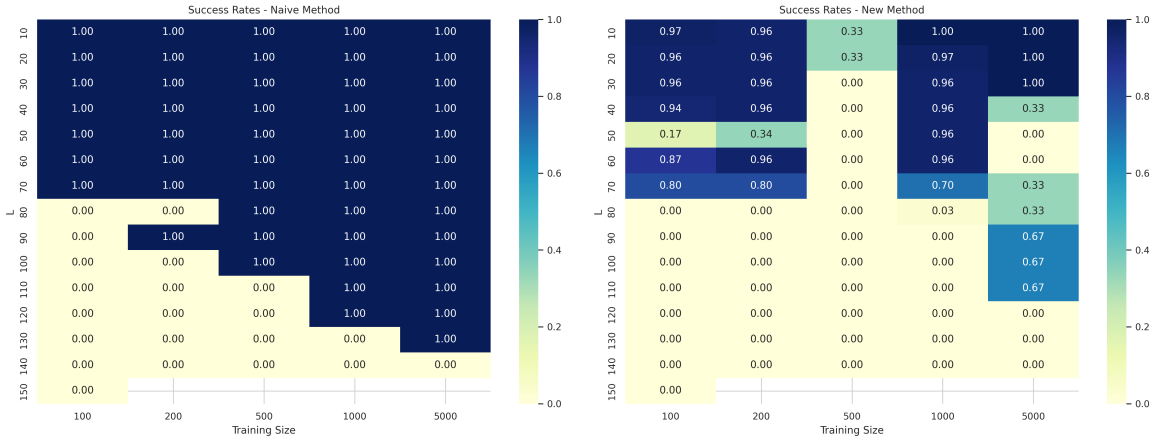
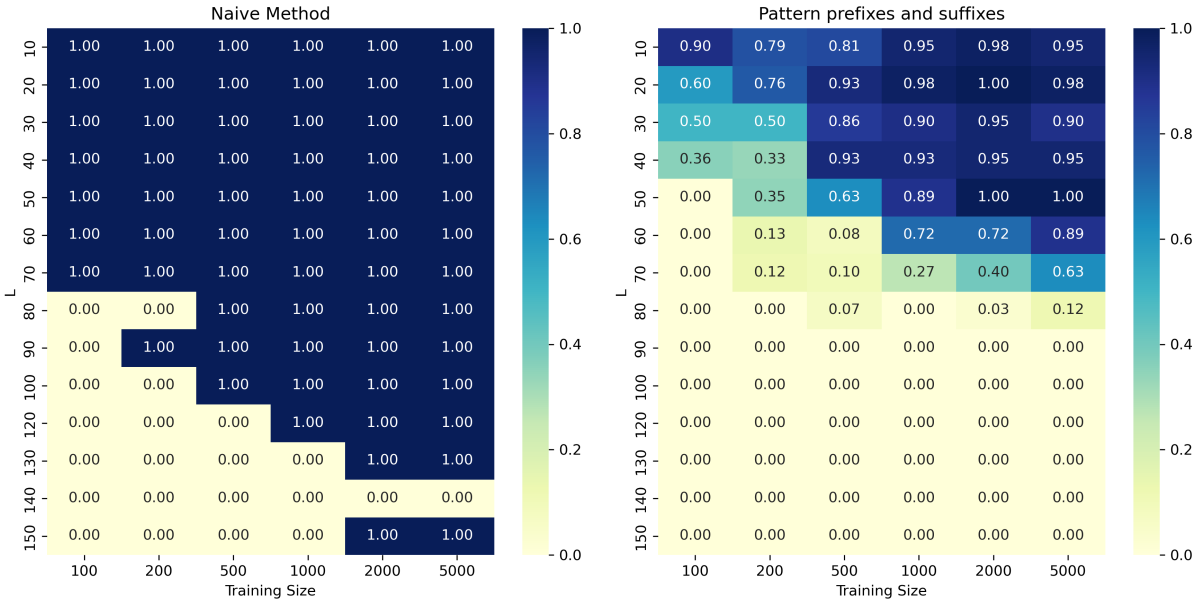


Figure 2: Sucess rates for naive and CH-PRFA with pattern methods



be an optimal balance between the number of prefixes and the overall performance of the algorithm. However, it is important to note that this finding is preliminary and requires more in-depth investigation to draw concrete conclusions.

6 Conclusion

In this paper, we presented a method for sequencing DNA using probabilistic automata (PA) based on the CH-PRFA algorithm. Our approach aimed to accurately retrieve the consensus sequence from noisy DNA data by learning the underlying probability distribution captured by the PA.

We proposed an alternative strategy for selecting prefixes and suffixes, considering the most common prefixes and suffixes of varying sizes, including pattern prefixes and suffixes. This approach aimed to preserve information about longer prefixes and suffixes, which tend to become increasingly rare as the word size increases.

Through our experimental evaluation, we compared the performance of our proposed method against a naive approach of selecting the most common word in the training set. The results showed that the naive method performs well for smaller training sets, but its success rate significantly decreases for larger lengths and training sizes. However, our proposed method did not improved performance.

For future works, [4] showed that initializing Baum-Welch algorithm - an iterative algorithm used to learn the hidden parameters of a Hidden Markov Model (HMM)- with the output of CH-PRFA can yield better results.

References

[1] Philipp L. Antkowiak et al. "Low cost DNA data storage using photolithographic synthesis and advanced information reconstruction and error correction". en. In: *Nature Communications* 11.1 (Oct. 2020), p. 5345. ISSN: 2041-1723. DOI: 10.1038/s41467-020-19148-3. URL: https://www.nature.com/articles/s41467-020-19148-3 (visited on 05/16/2023).

[2] Borja Balle, Ariadna Quattoni, and Xavier Carreras. "Spectral Learning Techniques for Weighted Automata, Transducers, and Grammars". In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing: Tutorial Abstracts*. Doha, Qatar: Association for Computational Linguistics, Oct. 2014. URL: https://aclanthology.org/D14-2002.

[3] Daniella Bar-Lev et al. *Deep DNA Storage: Scalable and Robust DNA Storage via Coding Theory and Deep Learning*. 2021. arXiv: 2109.00031 [cs.IT].

[4] Hadrien Glaude and Olivier Pietquin. "PAC learning of Probabilistic Automaton based on the Method of Moments". In: *International Conference on Machine Learning (ICML 2016)*. New York, United States, June 2016. URL: https://inria.hal.science/hal-01406889.

[5] Lee Organick et al. "Probing the physical limits of reliable DNA data retrieval". en. In: *Nature Communications* 11.1 (Jan. 2020), p. 616. ISSN: 2041-1723. DOI: 10.1038/s41467-020-14319-8. URL: https://www.nature.com/articles/s41467-020-14319-8 (visited on 05/08/2023).

[6] Lee Organick et al. "Random access in large-scale DNA data storage". en. In: *Nature Biotechnology* 36.3 (Mar. 2018), pp. 242–248. ISSN: 1087-0156, 1546-1696. DOI: 10.1038/nbt.4079. URL: http://www.nature.com/articles/nbt.4079 (visited on 05/16/2023).

[7] Xiangzhen Zan et al. "A super robust and efficient DNA storage architecture based on modulation encoding and decoding". In: *bioRxiv* (2022). DOI: 10.1101/2022.05.25.490755. eprint: https://www.biorxiv.org/content/early/2022/05/25/2022.05.25.490755.full.pdf. URL: https://www.biorxiv.org/content/early/2022/05/25/2022.05.25.490755.