Using Soft Information to Improve Error Tolerance of Motif-Based DNA Storage Systems

Eugenio Marinelli
Eurecom

Virginie Magnone, Marie-Charlotte Dumargne, Pascal Barbry
IPMC

Raja Appuswamy
Eurecom

***Abstract*—The surge in demand for cost-effective, durable long-term archival media, coupled with density limitations of contemporary magnetic media, has resulted in synthetic DNA emerging as a promising new alternative. Today, the limiting factors for DNA-based data archival are the high read/write cost and low throughput. New motif-based DNA storage solutions are being developed to address these limitations. However, these new techniques often reduce cost at the expense of reliability, as they introduce more errors during writing and reading data. In order to deal with such errors, it is important to design efficient pipelines that can carefully use redundancy to mask errors without amplifying overall cost. In this paper, we present OligoArchive-DSM (OA-DSM), an end-to-end, motif-based, DNA storage solution that provides high error tolerance at low read/write costs. The use of motifs as building blocks in OA-DSM creates new challenges related to the computation of soft information for improving error-correction decoding performance. Thus, we present heuristics for scaling LLR that rely on various design aspects of OA-DSM and demonstrate their utility using simulation studies and wet lab experiments.**

I. Introduction

In recent years, the growth of digital data has been exponential, with the global datasphere predicted to exceed 125 Zettabytes by 2025 [1]. However, almost 80% of the data created today is considered "cold" or rarely accessed, and companies need to archive it for long periods to comply with safety, legal, and regulatory requirements [2]. Unfortunately, traditional storage media face fundamental limitations in terms of density scaling and durability [3], [4], making them unsuitable for cost-efficient, long-term data archival. As a result, researchers have started exploring alternative media for long-term archival preservation. One such medium that has received a lot of attention recently due to its longevity and density is synthetic Deoxyribo Nucleic Acid (DNA) [3], [5].

Most prior approaches to DNA data storage follow the same basic steps. Data is written to DNA by encoding bits into sequences of four nucleotides (nts): Adenine, Guanine, Cytosine, and Thymine. These sequences are used to manufacture short DNA molecules, also referred to as oligos, using a process called *synthesis*. Reading the data back involves a process called *sequencing*, which generates a dataset of *reads*. The reads are fed as input to a decoding pipeline that infers the original sequences and uses them to restore back data by converting from nts into bits. Using these aforementioned steps, several researchers demonstrated the feasibility of using DNA as an archival medium. The main difference in these approaches is the type of algorithms and coding techniques used in various stages of the read/write pipeline.

Despite these advances, high read/write costs and low throughput have become a significant obstacle to DNA's adoption as a storage medium. Over the past few years, researchers have proposed new motif-based approaches to DNA data storage [6], [7] as a potential solution. Instead of using nts (A,C,G,T) as building blocks, these solutions use motifs, which are short oligonucleotide sequences that are drawn from a fixed library, as building blocks for assembling longer oligos. Using motifs as building blocks, one can scale logical density (the number of bits written per synthesis cycle) by storing $log_2(M)$ data bits per synthesis cycle [7]. The use of a fixed library of motifs similar to a typesetting press can simplify miniaturization and automation [8]. Further, new chemical synthesis methods (enzymatic assembly/ligation [9]) can be used to assemble oligos from motifs. While all these aspects are expected to contribute to reduce write cost and improving throughput, they do so at the expense of increased error rate in the synthesized oligos. Thus, effective error management is important for motif-based DNA storage to realize its potential.

In this work, we present our ongoing work on OligoArchive-DSM (OA-DSM), an end-to-end motif-based DNA storage system that significantly reduces read/write costs compared to SOTA approaches [10], [11], [12]. OA-DSM uses a novel, columnar encoding method for DNA storage together with an integrated consensus and decoding technique that leverages the motif structure and columnar organization of oligos to reduce read and write cost. In particular, we focus on the use of soft information for improving error correction capacity of OA-DSM. OA-DSM uses large-block-length, low-density parity-check (LDPC) codes to provide resilience against DNA storage errors. Prior work [10] in non-motif-based DNA storage has demonstrated that providing soft information, such as log-likelihood ratio (LLR), as input to LDPC decoders can substantially improve their error-correction capability. However, as we describe later, the use of motifs as building blocks and the columnar organization in OA-DSM requires a completely different approach to calculate soft information. In this work, we present a few estimation techniques, empirically validate them using real-world wet lab experiments.

II. System model

As mentioned earlier, state-of-the-art (SOTA) pipelines to encode data on DNA follow the same basic steps, although they differ in their implementation. Data is written to DNA by encoding bits into sequences of nts. This encoding is internally a two-step process. First, data is grouped into blocks of bits that are fed as input to an error-control coding module to produce parity bits. The data and parity bits together form a unit of recovery. The bits belonging to the block are mapped to nts. Due to limitations in synthesizing DNA, the length of a single DNA strand is limited to a few hundred nts at best. Thus, each block of data and parity is converted into a set of DNA sequences. These sequences are synthesized chemically to form actual DNA strands, also known as oligos.

979-8-3503-3959-8/23/$31.00 ©2023 IEEE

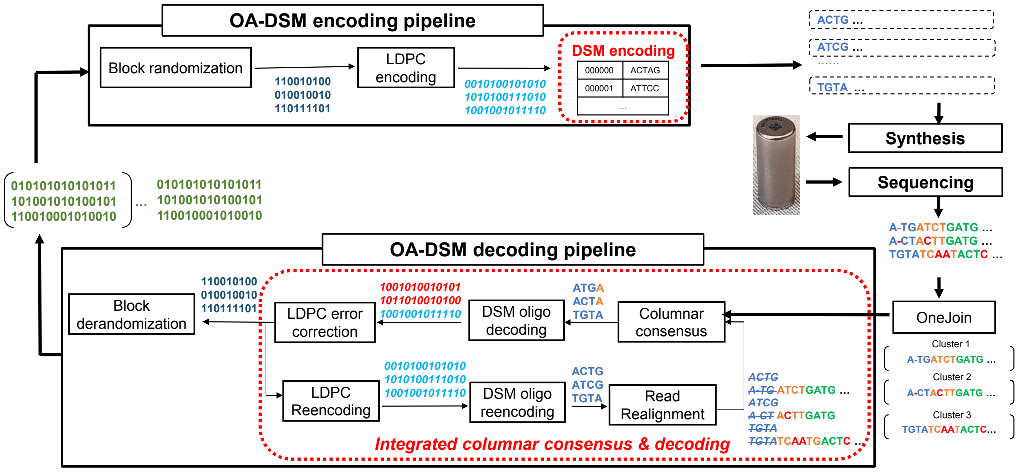

Fig. 1. OA-DSM data writing pipeline (top) showing binary to DNA encoding path, and long-term DNA storage in an encapsulated container like Imagene DNAShell™. OA-DSM reading pipeline (bottom) showing DNA to binary decoding path. The blocks in red are unique to OA-DSM (versus SOTA).

Reading the data back involves sequencing the DNA to produce reads. In the ideal case, sequencing would produce one read per oligo created during the writing phase. However, due to errors in sequencing and synthesis, reads are noisy duplicates of the starting sequences as they can contain substitution, insertion, and deletion errors. The number of reads corresponding to each oligo (also called *coverage*) can also vary by an order of magnitude, with some oligos being covered by thousands of reads, and others completely missing. In order to recover the original data from this noisy dataset, a consensus step is performed first to infer original sequences from noisy reads. The inferred sequences produced by the consensus step are then passed to the error-control decoder for conversion back from nts into bits. It is important to note here that these consensus sequences need not be error-free, accurate reproductions of original oligos. It is the job of the error-control decoder to use the additional parity bits to recover the original input data despite these errors. Thus, in all SOTA techniques, the decoding and consensus stages are independent of each other. This is an inherent consequence of how oligos are encoded. When a block of data is mapped to a group of oligos, that group becomes a unit of recovery that must be reassembled entirely (through consensus) before decoding.

OA-DSM, differs from the SOTA approaches by using a library of motifs as building blocks rather than individual nts. OA-DSM uses the motif library to perform encoding and decoding of data column-by-column instead of oligo by oligo. The principal advantage of this approach, as we demonstrate in subsequent sections, is that it enables the integration of decoding and consensus into a single step, where the error-correction provided by decoding improves consensus accuracy, which, in turn, increases the decoding performances, thereby producing a synergistic effect.

A. Write Pipeline: Motifs and Columnar Encoding

The top half of Figure 1 shows the OA-DSM data writing pipeline. The input to the write pipeline is a stream of bits. Thus, any binary file can be stored using this pipeline. OA-DSM uses large-block-length regular LDPC for error resilience. Thus, the input data is grouped into blocks of size 256,000 bits and LDPC-encoded to create a parity-extended data block with configurable redundancy. The encoded block is fed as input to the DSM-oligo-encoder which turns bits into oligos. While SOTA approaches design each oligo as a random collection of nts, the DSM-oligo-encoder designs oligos using motifs. Each motif is itself a short oligo that obeys all the biological constraints enforced by synthesis and sequencing. We use motifs rather than single nts as building blocks because, as we will see later in Section II-B, integration of decoding and consensus relies on alignment which cannot be done over single nts.

DSM-oligo-encoder converts bits into motifs by using an associative array with a 30-bit integer key and a 16 nt motif value. The encoder generates this array by generating all possible motifs of length 16 nt and eliminating motifs that do not meet certain biological criteria (up to two homopolymer repeats (AA,CC,GG, or TT) and GC content in the range of 0.25 to 0.75.) With these constraints, the encoder can use 1,405,798,178 valid motifs out of 4^{16} possible motifs. By mapping each motif to an integer in the range of 0 to 2^{30}, the encoder can encode 30-bits of data per motif. Thus, at the motif level, the encoding density is 1.875 bits/nt. Note that by using this associative array, we lose direct mapping between bits and nts. Instead, a group of 30 bits is mapped to a predefined set of nts. As described in Section II-C, this mapping makes SOTA approaches for computing LLR unsuitable in OA-DSM.

B. Read Pipeline: Sequence Alignment and Consensus

The bottom half of Figure 1 shows the OA-DSM read pipeline. Data stored in DNA is read back by sequencing the DNA to produce reads. As each oligo can be covered by multiple reads, the first step in decoding is clustering [13], [14] to separate the reads into a set of clusters, with each cluster corresponding to some unknown original oligo. After the clustering stage, other SOTA methods apply consensus in each cluster followed by decoding in two separate phases. In OA-DSM, we exploit the motif design and columnar layout of oligos to iteratively perform consensus and decoding in an integrated fashion. Unlike other approaches, OA-DSM processes the reads one column at a time. Thus, the first step is columnar consensus using a bit-wise majority algorithm [14], which takes as input the set of reads and produces one column of motifs. These motifs are fed to the *DSM-oligo-decoder* which maps the motifs into their 30-bit values using the same array of motifs used during encoding, which does not need to be transmitted but can be generated prior to decoding. Note here that despite consensus, the inferred motifs can still have errors. These wrong motifs will result in wrong 30-bit values. These errors are fixed by the standard BP *LDPC-decoder*, which takes as input the 256,000 bits corresponding to one LDPC block and produces as output the error-corrected input bits.

SOTA techniques do not employ the error-corrected input bits during decoding. In contrast, OA-DSM takes advantage of these bits to increase accuracy, as seen in the lower portion of Figure 1. The error-corrected, LDPC-decoder's bits are encoded again to create a column of correct motifs. Reads are realigned using the appropriate column of motifs so that the subsequent round of columnar decoding begins at the correct offset. This realignment's rationale is based on the following. Due to a difference in length, an insertion or deletion error in a consensus motif will not only damage that motif but also all downstream motifs. For instance, if we look at the example in Figure 1, we see a deletion error in read $A-TGATCTG..$ which should have been $ACTGATCTG....$ This results in the first motif being incorrectly interpreted as $ATGA$ (instead of $ACTG$, and the second motif as $TCTG$ (instead of $ATCT$). Thus, an error early in consensus keeps propagating. Without a knowledge of the correct motif, there is no way to fix this error. But in OA-DSM, by reencoding the error-corrected bits, we get the correct motifs. By aligning these motifs against the reads, we can ensure that consensus errors do not propagate. In the example shown in the red block of Figure 1, the alignment enables us to identify the precise starting position of the second motif, which can now be correctly interpreted as $ATCT$. Note here that such realignment is only possible because we use motifs, as two sequences can be aligned accurately only if they are long enough to identify similar subsequences. Thus, columnar layout without motifs, or with just nts, would not make realignment possible. Similarly, integrating consensus and decoding is possible only because of the columnar layout, as the SOTA layout that spreads a LDPC block across several oligos cannot provide incremental reconstruction.

C. Improving OA-DSM with Soft Information

The error correction capabilities of LDPC code can be improved by feeding the LDPC decoder with soft information such as the log-likelihood ratio (LLR). However, the calculation of LLR depends on the channel model, which presents a challenge for DNA storage channels due to the lack of an exact channel model with a specific distribution. To address this issue, Chandak et al. [10] proposed a simplified method for LLR computation that is defined under the assumption that DNA storage mimics a binary symmetric channel with error probability epsilon and an ideal Poisson random sampling model. Under this assumption, the transition probability for the channel is given as:

$$p\left(\left(k_0,k_1\right)|0\right)=\frac{e^{-\lambda}\lambda^{k_0,k_1}}{(k_0,k_1)!}\begin{pmatrix}k_0+k_1\\k_0\end{pmatrix}(1-\epsilon)^{k_0}\,\epsilon^{k_1} \quad (1)$$

$$p\left(\left(k_0,k_1\right)|1\right)=\frac{e^{-\lambda}\lambda^{k_0,k_1}}{(k_0,k_1)!}\begin{pmatrix}k_0+k_1\\k_1\end{pmatrix}(1-\epsilon)^{k_1}\,\epsilon^{k_0} \quad (2)$$

where λ is the ratio between reading cost and writing cost, the input of the channel is a bit, the output is a tuple (k_0, k_1) where k_b is the number of times that the bit is read as b. From this, the LLR is computed as:

$$\log\frac{p\left(\left(k_0,k_1\right)|0\right)}{p\left(\left(k_0,k_1\right)|1\right)}=(k_0-k_1)\log\frac{1-\epsilon}{\epsilon} \quad (3)$$

Chandak et al. [10] use nts as the building blocks of encoding with a direct mapping between two bits and four nts (A-00, C-01, G-10, T-11). During the consensus stage of the read pipeline, they perform read clustering to group similar reads into buckets such that each bucket corresponds to an original sequence. Then, for each position, they count the number of occurrences of nts, and use it to directly determine k_0 and k_1 values for the consensus LLR. For example, to determine the LLR for the first bit, the count of As and Cs is used as k_0 as they would result in first bit being 0, and the count Gs and Ts is used as k_1 as they would result in first bit being 1. Chandak et al. [10] showed that such an LLR computation improves the read/write cost of DNA data storage, as it allowed LDPC decoders to tolerate more errors while using fewer parity bits.

Unfortunately, this approach cannot be directly applied to OA-DSM; instead of mapping two bits to 1 nt like Chandak et al. [10], OA-DSM maps 30 bits to 16 nts using an associative array. Thus, using a per-position nt count to compute per-bit k_0 and k_1 is not correct for OA-DSM. The straightforward extension of Chandak et al. [10]'s approach to motif design is to extract motifs from the reads, identify which motifs contribute to zero and one bits for each position, and use their count for k_0 and k_1. Unfortunately, this approach does not work well due to two problems. First, it causes error amplification. When nts are used as building blocks, an error in a single nt will result in the corresponding two bits being wrong. However, with OA-DSM, a single error can result in a completely different 30-bit pattern compared to the correct motif. Thus, using the wrong motif to derive a count of zeroes and ones will result in poor LLR. Second, this approach is slow, as it requires tens of billions of array lookups to convert each motif in each read into its bit sequence.

Given these issues, we implemented three heuristic approaches in OA-DSM for approximating LLR in a scalable fashion. Our first heuristic exploits the fact that we apply a consensus procedure to infer each motif in the read pipeline. When each inferred motif is converted back into bits, we apply Equation 3 for each bit, where the difference $\alpha = k_0 - k_1$, is set to 1 or -1. The intuition behind this is that we view the majority consensus output as the only bits emerging out of the DNA storage channel; a zero bit results in k_0 being 1 and k_1 being 0, and a one bit results in k_1 being 0 and k_0 being 1. Thus, α is ±1. Although this definition works in practice (as we will show in Section III), it does not provide any additional information about the reliability of the bits given by consensus.

Our second heuristic extends the first one by taking each consensus motif, counting the number of reads that exactly with it, and using the count as α. The intuition behind this heuristic is that in the general case, with sufficient coverage, the count of "correct" motifs will be higher than the count of wrong motifs. In such a case, the number of correct motifs can be used as k_0 for a zero bit, and k_1 for a one bit. As majority consensus will identify the correct motif, we can get this count by comparing the consensus motif with every read. Thus, α is $\pm$(exact match count). The drawback of this method is that it requires motifs in reads to exactly match the consensus motif. However, in cases of high error rates, it is possible that the consensus produces the correct motif, but the consensus result does not match even a single read.

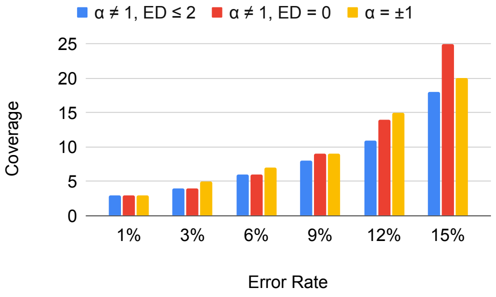

Fig. 2. Figure shows the minimum coverage required to guarantee 100% data recovery of various LLR methods under a range of error rates.

Our third heuristic extends the second one by exploiting the fact that OA-DSM uses a bitwise majority consensus algorithm that internally aligns motifs to compute consensus motifs. Alignment is done to deal with insertion and deletion errors that can make motifs in a read longer or shorter than the original motif. Such an alignment also produces the edit distance, or Levenshtein distance, between motifs. Thus, our third heuristic uses this information by changing the exact match count to the count of motifs having an edit distance lower or equal to a configurable threshold. We set this threshold to 15% to tolerate a sufficiently high error rate.

III. Evaluation

In this section, we present the experimental results to highlight the effectiveness of using soft information in OA-DSM. First, we compare the performance of our columnar decoding with respect to various LLR heuristics. Then, we validate the end-to-end encoding and decoding pipeline by presenting the results of our wet lab experiment. Finally, we compare OA-DSM to other SOTA approaches with respect to read/write costs. All experiments were conducted on a server equipped with a 12-core Intel (R) CPU and 128GB of RAM.

To compare the heuristically defined LLRs in Section II-C, we encoded a binary file of size 609KB (a relational database generated by using TPC-H DBGEN utility) into 22,188 oligos of 160 nts each. Using a custom simulator, we constructed from this set of oligos multiple simulated reads datasets, by uniformly injecting errors with an error rate ranging from 1% to 15% and varying coverage levels. For each simulated read dataset and heuristic LLR, we conducted multiple runs to determine which LLR approach requires the smallest amount of coverage to decode the entire file. Figure 2 reports the minimum coverage that our OA-DSM decoder needs to recover data for different types of errors, depending on the LLR it uses. As can be seen, LLR having $\alpha = \pm 1$ is capable of decoding data. At low error rates, it performs as well as other methods. But at high error rates, it is able to recover data only at a much higher coverage. The exact match LLR (shown as $\alpha \neq 1$ ED=0) marginally outperforms the $\alpha = \pm 1$ LLR for 9% and 12% error rate, but underperforms at 15%. This proves our intuition that at high error rates exact matches might not be found; at 15% error rate, there is an error in every single motif. While consensus can still recover the original motif, the lack of matches would drive the LLR to zero, resulting in poor performance. On the other hand, the LLR using alignment-derived edit distance (shown as $\alpha \neq 1$, ED ≤ 2) outperforms other LLRs.

	OA-DSM	LDPC-30%	RS+RLL	Fountain+RS
Read Cost	2.10	4.46	4.50	6.8
Write Cost	0.70	0.78	0.92	0.65

TABLE I. OA-DSM vs. SOTA rd/wt costs: LDPC [10], RS-RL [11], Fountain+RS [12]

Having identified the LLR that allows for the lowest coverage in decoding, we validated OA-DSM (with $\alpha \neq 1$, ED ≤ 2 LLR) using a real wet-lab experiment. In this wet-lab experiment, we encoded a 1.2MB binary file representing a compressed relational database archive with 30% LDPC redundancy to generate 44376 oligos, with each oligo having a length of 160 nts. Twist Biosciences synthesized the oligos, which we sequenced using the Oxford Nanopore PromethION platform, generating approximately 43 million noisy reads. We ran the pipeline with all 43 million reads, corresponding to an average coverage of 951×, and were able to fully reconstruct the original data. In order to prove the OA-DSM's capability to handle lower coverage, we subsampled 200K reads from the original dataset, generating a new dataset with an average coverage of 4×. We found that OA-DSM was able to perform full recovery of the original data despite nearly 3500 oligos being completely missing in the subsampled dataset, that is, not covered by any read. With our wetlab experiment, we validated 4× as the least coverage OA-DSM can manage, as further reduction in coverage resulted in data loss.

We conclude this section by presenting the comparison between OA-DSM and SOTA methods, including LDPC coding by S. Chandak et al. [10], large-block Reed-Solomon coding by Organick et al. [11], and fountain codes by Erlich et al. [12], in terms of reading and writing costs. Writing cost is defined as $\frac{\#nts-in-oligos}{\#bits}$, where the numerator is the product of the number of encoding oligos by their length and the denominator is the input file size expressed in bits. Similarly, reading cost is defined as $\frac{\#nts-in-reads}{\#bits}$, i.e., as the ratio between the sum total of all read lengths and the input size in bits. The higher the redundancy and encoding overhead, the higher the write cost, while the higher the coverage required, the higher the read cost. Table 1 compares the read/write cost for OA-DSM and other SOTA approaches. We computed the costs for OA-DSM using the values from the wet-lab experiment. Table 1 shows that OA-DSM outperforms other SOTA methods substantially in terms of read cost. The write cost of OA-DSM is slightly higher than that of fountain codes but significantly lower than large-block Reed-Solomon coding. The focus of this work was on soft information for improving decoding performance, and hence the read cost. OA-DSM can achieve further reductions in write cost by reducing redundancy and scaling the motif set to use more motifs. We leave open these optimizations to future work. These results suggest that OA-DSM can achieve a good balance between read and write costs, making it a promising solution for efficient DNA data storage.

IV. Conclusion

In this work, we presented our ongoing work on OA-DSM– a motif-based DNA storage system. Considering the mismatch of motif-based design used by OA-DSM with nucleotide-based LLR computation proposed by SOTA methods, we proposed three strategies for computing LLR based on various design aspects of OA-DSM. Using results from simulation studies and real-world wet lab experiments, we demonstrated the ability of soft information to reduce read/write costs in OA-DSM.

References

[1] J. G. David Reinsel and J. Rydning, "Data age 2025: the digitization of the world from edge to core."

[2] Intel, "Cold Storage in the Cloud: Trends, Challenges, and Solutions," White Paper.

[3] S. R. Corporation, "2018 semiconductor synthetic biology roadmap," https://www.src.org/program/grc/semisynbio/ssb-roadmap-2018-1st-edition_e1004.pdf, 2018.

[4] R. Appuswamy, K. Lebrigand, P. Barbry, M. Antonini, O. Madderson, P. Freemont, J. MacDonald, and T. Heinis, "OligoArchive: Using DNA in the DBMS storage hierarchy," in *CIDR*, 2019.

[5] G. M. Church, Y. Gao, and S. Kosuri, "Next-Generation Digital Information Storage in DNA," *Science*, vol. 337, no. 6102, 2012.

[6] 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.

[7] Y. Yan, N. Pinnamaneni, S. Chalapati, C. Crosbie, and R. Appuswamy, "Scaling logical density of dna storage with enzymatically-ligated composite motifs," *bioRxiv*, 2023.

[8] "Catalog dna," https://www.catalogdna.com, accessed: 2022-11-20.

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

[10] S. Chandak, K. Tatwawadi, B. Lau, J. Mardia, M. Kubit, J. Neu, P. Griffin, M. Wootters, T. Weissman, and H. Ji, "Improved read/write cost tradeoff in dna-based data storage using ldpc codes," in *2019 57th Annual Allerton Conference on Communication, Control, and Computing*, 2019.

[11] L. Organick, S. D. Ang, Y.-J. Chen, R. Lopez, S. Yekhanin, K. Makarychev, M. Z. Racz, G. Kamath, P. Gopalan, B. Nguyen *et al.*, "Random access in large-scale dna data storage," *Nature biotechnology*, vol. 36, no. 3, pp. 242–248, 2018.

[12] Y. Erlich and D. Zielinski, "Dna fountain enables a robust and efficient storage architecture," *science*, vol. 355, no. 6328, pp. 950–954, 2017.

[13] E. Marinelli and R. Appuswamy, "Onejoin: Cross-architecture, scalable edit similarity join for dna data storage using oneapi," in *ADMS*, 2021.

[14] E. Marinelli, E. Ghabach, Y. Yan, T. Bolbroe, O. Sella, T. Heinis, and R. Appuswamy, *Digital Preservation with Synthetic DNA*, 2022.