Iterative Soft Decoding Algorithm for DNA Storage Using Quality Score and Redecoding

Jaeho Jeong, *Graduate Student Member, IEEE*, Hosung Park, *Member, IEEE*, Hee-Youl Kwak, *Member, IEEE*, Jong-Seon No, *Fellow, IEEE*, Hahyeon Jeon, Jeong Wook Lee, and Jae-Won Kim, *Member, IEEE*

***Abstract*— Ever since deoxyribonucleic acid (DNA) was considered as a next-generation data-storage medium, lots of research efforts have been made to correct errors occurred during the synthesis, storage, and sequencing processes using error correcting codes (ECCs). Previous works on recovering the data from the sequenced DNA pool with errors have utilized hard decoding algorithms based on a majority decision rule. To improve the correction capability of ECCs and robustness of the DNA storage system, we propose a new iterative soft decoding algorithm, where soft information is obtained from FASTQ files and channel statistics. In particular, we propose a new formula for log-likelihood ratio (LLR) calculation using quality scores (Q-scores) and a redecoding method which may be suitable for the error correction and detection in the DNA sequencing area. Based on the widely adopted encoding scheme of the fountain code structure proposed by Erlich et al., we use three different sets of sequenced data to show consistency for the performance evaluation. The proposed soft decoding algorithm gives 2.3%∼7.0% improvement of the reading number reduction compared to the state-of-the-art decoding method and it is shown that it can deal with erroneous sequenced oligo reads with insertion and deletion errors.**

Manuscript received 24 October 2022; revised 21 March 2023 and 4 June 2023; accepted 5 June 2023. Date of publication 9 June 2023; date of current version 3 January 2024. This work was supported in part by the Samsung Research Funding and Incubation Center of Samsung Electronics under Project SRFC-IT1802-09; and in part by the Pioneer Research Center Program through the National Research Foundation of Korea funded by the Ministry of Science, ICT and Future Planning under Grant NRF-2022M3C1A3081366. *(Corresponding author: Jae-Won Kim.)*

Jaeho Jeong and Jong-Seon No are with the Department of Electrical and Computer Engineering, Institute of New Media and Communications (INMC), Seoul National University, Seoul 08826, South Korea (e-mail: jaehoj@ccl.snu.ac.kr; jsno@snu.ac.kr).

Hosung Park is with the Department of Computer Engineering and the Department of ICT Convergence System Engineering, Chonnam National University, Gwangju 61186, South Korea (e-mail: hpark1@jnu.ac.kr).

Hee-Youl Kwak is with the School of Electrical Engineering, University of Ulsan, Ulsan 44610, South Korea (e-mail: hykwak@ulsan.ac.kr).

Hahyeon Jeon is with Clinomics, Ulsan 44919, South Korea (e-mail: jeonhh0061@gmail.com).

Jeong Wook Lee is with the Department of Chemical Engineering, POSTECH, Pohang 37673, South Korea (e-mail: jeongwook@postech.ac.kr).

Jae-Won Kim is with the Department of Electronic Engineering, Engineering Research Institute (ERI), Gyeongsang National University, Jinju 52828, South Korea (e-mail: jaewon07.kim@gnu.ac.kr).

Digital Object Identifier 10.1109/TNB.2023.3284406

***Index Terms*— DNA storage, FASTQ file, fountain code, iterative decoding, luby transform code, oligo, quality score, soft decoding.**

I. Introduction

As HUGE amounts of data are rapidly produced in the era of big data, there is a need for a new type of high-density storage media that is more competitive than existing ones such as magnetic tapes or hard disk drives. Major information technology companies are spending enormous financial resources for building new data centers to cope with the increased storage demand. With the development of new synthesis and sequencing technologies, deoxyribonucleic acid (DNA) has been considered as a competitive candidate for new storage media [1]. DNA is known to have up to more than hundreds of million times information density compared to the existing storage media and can survive thousands of years when the temperature is kept appropriately [2]. Since the pioneering study in [1], the idea of using DNA as a storage medium has attracted a significant attention and a lot of studies are going on to make use of the advantages of its longevity and high information density nowadays.

As the data storage device, we consider single-stranded DNA consisting of four nucleotides (i.e., bases). These four bases are Adenine, Cytosine, Guanine, and Thymine, which are simply represented as A, C, G, and T, respectively. We refer to this single-stranded DNA as an oligonucleotide or simply an oligo. For the DNA storage system, writing and reading processes are called DNA synthesis and sequencing, respectively. In order to read information stored in oligos through sequencing, polymerase chain reaction (PCR) that amplifies the information is needed in advance. In fact, each oligo is synthesized in large numbers and the synthesized number for each oligo is not uniform. Furthermore, each piece of information stored in the oligo can be amplified to a different extent during PCR amplification. Using the sequencing equipment, sequenced data is usually produced in computer file formats. Among them, FASTQ files had become one of the common formats in the DNA sequencing area [3] by providing a quality score (Q-score), which is a probability of how reliably each base is predicted and sequenced. In all steps described above, three types of errors can occur, that is, substitution errors,

insertion errors, and deletion errors. The error distributions and behaviors are very different from those of the conventional memory devices. While substitution errors have mainly been considered in the conventional memory devices, the other types of errors are rarely considered. To handle these errors, it is necessary to adopt carefully designed error correcting codes (ECCs) for data.

To guarantee data reliability for DNA storage, there are some biochemical constraints on data such as the oligo length, maximum homopolymer-run length, and GC-content [4], [5]. The lengths of oligos are typically from hundreds to a few thousands and it is reported that longer oligos are hard to be synthesized and prone to errors during several steps. In [6] and [7], the effects of the maximum homopolymer-run length and GC-content were discussed. It was shown that long homopolymer-run length of oligos increases error rates and a moderate GC-content is needed for PCR amplification to avoid high dropout rates. Also, studies in [8], [9], [10], and [11] suggested that some constraints such as the minimum free energy (MFE) constraint, end-constraint, self-complementary constraint, distances between oligo sequences, and correlation between addresses should be considered to reduce the error rate of DNA storage.

In addition, there are many works on new coding schemes for the DNA storage system [12], [13], [14], [15], [16], [17], [18], [19], [20], [21], [22]. Organick et al. [15] designed a large number of primers and validated that the individual recover of files is possible. In the study of Yazdi et al. [16], an iterative alignment algorithm was introduced and they converted all types of errors into deletion errors, while Erlich and Zielinski [17] achieved 86% of the theoretical limit (i.e., capacity) through the encoding scheme using fountain codes and Reed-Solomon (RS) codes. Also, Chandak et al. [18] developed a single large block code structure using low-density parity-check (LDPC) codes with soft decoding using log-likelihood ratio (LLR). Press et al. [19] handled insertion and deletion errors using a convolutional code with a hash function. Additionally, Wang et al. [20] designed an oligo structure using only one primer binding site (PBS) for higher information density and Cao et al. [21] applied adaptive coding using fountain codes to satisfy several constraints mentioned above [6], [7], [8], [9], [10], and [11]. Recently, Rasool et al. [23] introduced a new approach based on the moth-flame optimization (MFO) algorithm [24] to improve DNA coding rates and the lower bounds of DNA coding sets. In the end, our previous work [22] achieved a decoding performance improvement by combining several preprocessing components and decoding techniques while the encoding scheme is the same as that of [17].

In this paper, we propose an iterative soft decoding algorithm based on redecoding using RS codes. We follow the encoding scheme of [17] like our previous works [22], [25] and show the performance improvement by modifying the decoding algorithm. In particular, we decode the received codewords (i.e., sequencing results) using soft information provided by FASTQ files, that is, Q-scores for oligo reads. In the previous work [17], they used RS codes as error detecting codes to check whether sequenced reads of oligos are erroneous. If the sequenced oligo reads are erroneous, those oligo reads are discarded, which means that the decoding method in [17] requires a larger number of sequenced oligo reads.

From the decoding performance of the proposed algorithm, we show that the soft information acquired from FASTQ files can reduce the required number of sequenced oligo reads and accordingly save the reading cost of the sequencing process. While the LLR calculation in [18] for soft decoding was based on the hard values (i.e. the difference between the numbers of 0s and 1s), we determine the LLR values based on the probability of the correct basecall from the Q-scores and channel statistics for entire oligo reads. Since FASTQ files only offer a Q-score for one nucleotide in each position, we estimate the probabilities for the other nucleotides with the channel statistics considering edit distances. From the basecalling probabilities of the four bases in each position of oligo reads, we can obtain an accurate estimate of the LLR values and perform soft decoding with high performance.

Compared to our previous work [22] in which the main decoding process stays unchanged from [17], this work presents a novel decoding process summarized as follows:

- We first propose a new formula for LLR caluclation utilizing soft information from FASTQ files, which is the first attempt in the DNA storage field and can inspire diverse studies in the DNA sequencing area.
- We introduce a new iterative soft decoding algorithm using the redecoding method based on RS decoding results. It is shown that this redecoding method can extract and discard erroneous sequenced oligo reads with insertion and deletion errors.
- We use three different sets of sequenced data to show the performance improvement of the proposed decoding algorithm compared to the state-of-the-art algorithm [22].

This paper is organized as follows. In Section II, we cover our experimental design including the synthesis and sequencing information. We also explain the proposed iterative soft decoding algorithm and how each technique works. In Section III, we show the superiority of the proposed methods compared to existing decoding methods and give analysis on the experimental results. In Section IV, we discuss several considerations that we could not include in this paper and expects for the future works. Finally, conclusion is given in Section V.

II. Materials and Methods

Most of the previous works in DNA storage rely on hard decoding for the decoding process after the sequencing process is done. However, research in [18] showed that soft decoding can be applied for the DNA storage system. Even though we employ a different code structure from [18], we adopt LLR calculation and the soft decoding into our experiment.

A. Experimental Design

We first use two different sequenced data sets to show the flexibility and consistency of our proposed methods. One sequenced data set is the experimental data used for our

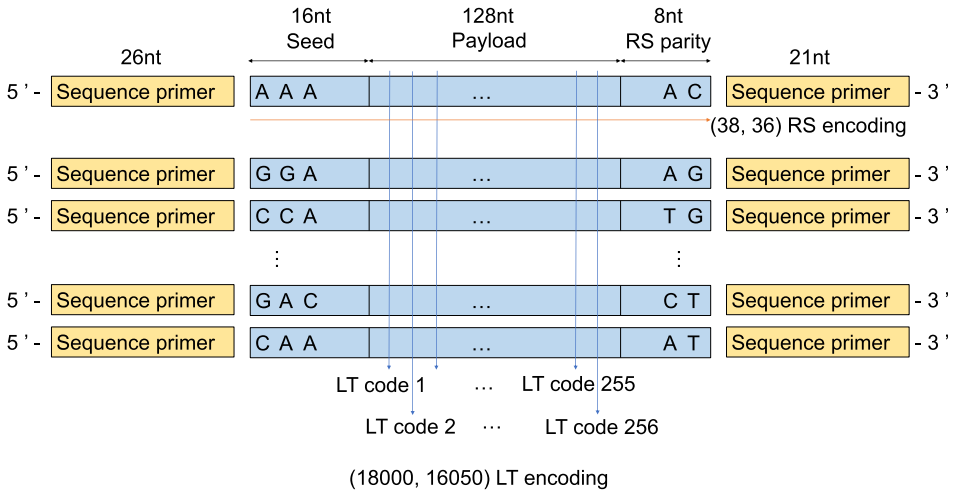

Fig. 1. The encoding structure of our experiment. We use the (18000, 16050) LT code for the inter-oligo code and the (38,36) RS code for the intra-oligo code, in which four nucleotides make one RS symbol.

previous work [22] and we call it $dataA$. For the other set, called $dataB$, we conduct another sequencing experiment from the remaining synthesized pool in [22] with the same environmental conditions.

Our encoding structure is shown in Fig. 1. As we mentioned in [22], we use the Luby transform (LT) code [26] for the inter-oligo code and the RS code for the intra-oligo code. The length of each oligo is 152 nucleotides (nt), where each oligo is composed of 16nt, 128nt, and 8nt for the seed, payload, and RS parity, respectively. For the RS code, a parity symbol is 4nt each and we use the (38, 36) RS code in the finite field of $GF(2^8)$, which makes a minimum Hamming distance be 3, implying 1-symbol correction or 2-symbol detection is available. The RS code has powerful error correction performance on a small length of messages and is appropriate for burst-error channels, and thus the RS code is suitable for dealing with insertion and deletion errors as long burst-errors. Using LT encoding with the robust-soliton distribution [17], we make 18000 oligo sequences from 16050 information sequences converted from the image file of 513.6 KB. From the mapping between the binary data and DNA bases, we transform two consecutive bits into a base, that is, $00 = \text{A}$, $01 = \text{C}$, $10 = \text{G}$, and $11 = \text{T}$. We add two sequence primers at the ends of each oligo sequence:

- 5' side: GTTCAGAGTTCTACAGTCCGACGATC,
- 3' side: TGGAATTCTCGGGTGCCAAGG,

and these are the same sequence primers used in the previous works [17], [22].

In [22], two oligo pools were synthesized to show the effects of homopolymer-run length and GC-content constraints. Among them, we first consider the non-constrained pool ($dataA$ and $dataB$) in this section, which allows us to have the seed information of the total oligo pool during the decoding process. The non-constrained pool means that we do not contemplate any homopolymer-run length or GC-content constraints for encoding. Considering the encoding structure for the non-constrained pool [22], it is easily noticed that we just directly create 18000 different seed structures of LT codes on the encoding process. It means that we use the first 18000 seeds of the robust-soliton distribution [17] and we can get exactly the same seed structures when we follow the first 18000 seed calculation at the decoding process. Since the purpose of this paper is to improve the decoding performance, we try to restrict other conditions such as exactness of seed information and several preprocessing steps. For this reason, the non-constrained pool is more suitable for comparison between different decoding algorithms. Further, the decoding performance of the non-constrained pool was comparable to that of the constrained pool in the previous study [22], which means that errors induced by not satisfying the homopolymer-run length and GC-content constraints can be reasonably covered by decoding algorithms.

For the new experiment with $dataB$, we use the same sequencing process applied for the non-constrained pool in [22]. The ratio of the GC-content is $[0.3289, 0.6842]$ and the maximum homopolymer-run length is 13. We use RP1 and RPI1 primers for PCR amplification and Illumina Miseq Reagent v3 kit for the sequencing with 600 cycles, reading 151nt each in forward and reverse directions. Q30 of this new sequencing experiment is 96% and output contains 22 million forward-reads and 22 million reverse-reads, in total 44 million sequenced oligo reads, leaving 36% PhiX spike-in.

B. Proposed Decoding Algorithm

The proposed decoding process is shown in Fig. 2, where each process will be explained in the following subsections. In the previous work [22], we used the oligo clustering that binds similar sequenced oligo reads into a cluster by comparing the Hamming distance of the whole oligo sequences. In this work, we use seed clustering that binds sequenced oligo reads with exactly the same seed structure into a cluster by comparing only the seed part of the sequenced oligo reads. Since we have the seed information of the oligo pool by calculating the robust-soliton distribution, we can decide that which seed structure is erroneous and which seed structure is correct (although undetected errors can exist). Then, we only

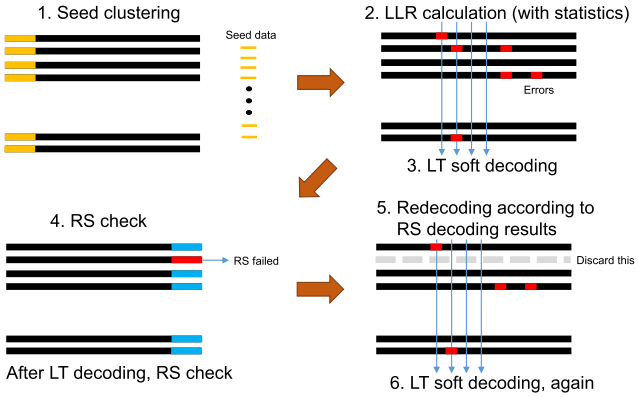

Fig. 2. The description for the proposed decoding process.

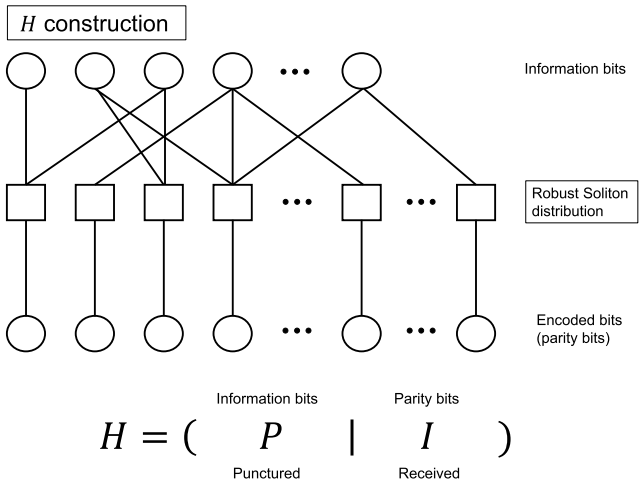

Fig. 3. The construction of a parity check matrix H.

utilize the clusters with the pre-determined seed structure and discard clusters that have different seed parts. Using this seed information, we make a parity check matrix H for LT soft decoding as shown in Fig. 3. In order to construct the valid parity check matrix, the decoder has to know the pre-determined seed values and it is why we consider the non-constrained pool.

For LT codes, the number of symbols K required for LT decoding with a failure rate δ is derived in Theorem 12 of [26] as follows:

$$K = k\beta = k(\sum_{i} \rho(i) + \tau(i)) = k + \sum_{i=1}^{k/R-1} \frac{R}{i} + R\ln(R/\delta),$$

where k equals to the number of information packets and $R = c \times \ln(\frac{k}{\delta}) \times \sqrt{k}$ for a constant value $c > 0$. By setting parameters capturing our experimental conditions, we could have $K = 16951$ with $\delta = 0.001$ and $c = 0.025$. Based on the calculated value K, we need to make sure that the number of clusters participating for LT decoding should be at least K or more during the decoding process. In Fig. 3, the number of information bits equals to 16050 and the number of parity bits should be in a range $16951 \leq K \leq 18000$. After constructing H matrix using seed information, we perform belief propagation (BP) decoding [27] with this H matrix for 500 decoding iterations.

Our DNA oligo pool has a product structure of the LT code and RS code, and thus we can choose which code to be decoded first. The previous work [22] decoded the RS code first and LT code later following the similar decoding process in [17]. At that work, we used the RS code for the error correction and if the decoded codeword had errors in it, this cluster would give a terrible effect over the whole LT decoding process. This is because the LT code only worked as a prevention for a loss of each oligo sequence while not giving any help for the error correction. In this work, however, we decode the LT code first and RS code later. We use soft decoding for the LT code and use it for the error correction and use the RS code for both the error correction and error detection. In this scenario, if there are errors in the LT correction, we can find them using the RS code. Specifically, we utilize the RS code for two purposes: i) error detection for seed positions and ii) error correction for other positions. If the RS decoding fails or error is detected for the seed positions, we discard that erroneous information and repeat the LT soft decoding again by constructing a new H matrix while removing the erroneous oligo reads. In the following subsections, we explain these decoding processes in detail.

1) LLR Calculation: Since the inputs of BP decoding are LLR values for each bit, we need to extract LLR values from the soft information [28]. In the previous work using soft decoding [18], they used hard values for calculating LLR values based on basecalling. By mapping the basecalled hard values into 0s and 1s, they counted the numbers of 0s and 1s in a cluster for each position, and calculated LLR values according to the numbers of 0s and 1s with the heuristically determined crossover probability. On the other hand, we try to extract soft values provided in the sequenced FASTQ files, that is, Q-scores.

To obtain accurate LLR values, we use the error distribution statistics of the entire sequenced data from the oligo pool. We collect all the correct-length (152nt) stitched oligo reads using the PEAR algorithm [29] and compare them with the original encoded data to figure out the occurrence of errors in each sequenced oligo read. Then, for each position, we calculate conditional rates for every combination of base transition, i.e., A→C, A→G, A→T, C→A, C→G, C→T, G→A, G→C, G→T, T→A, T→C, and T→G, where X→Y denotes the event that X is stored but Y is sequenced instead ($X \neq Y$). Since these probabilities are conditional (for errors occuring), we calculate the conditional probabilities only in the cases of sequenced oligo reads with any error occurrences. For the decoding process, we only consider position errors, which means that we handle the insertion and deletion errors by considering them as burst substitution errors. This is reasonable because the insertion and deletion error rates are very low compared to the substitution error rate when collecting only the correct-length stitched oligo reads. Furthermore, there is no way to correct insertion and deletion errors in the perspective of ECCs because we employ the encoding scheme in [17]. For this reason, we only consider and utilize correct-length

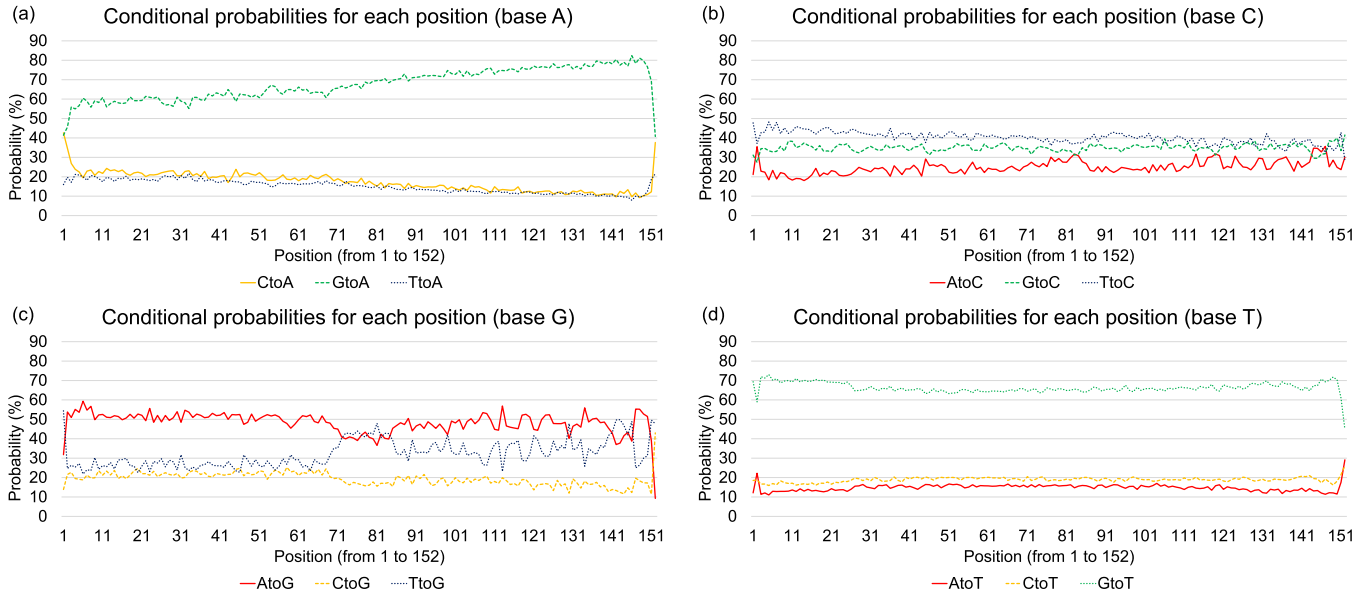

Fig. 4. The graphs of conditional probabilities of base transitions in position errors for each position in *dataB*. XtoY in the graphs means a probability that X is stored but Y is sequenced instead (X $\neq$ Y). The probabilities are given from position 1 to 152. (a) Conditional probabilites for base A. (b) Conditional probabilites for base C. (c) Conditional probabilites for base G. (d) Conditional probabilites for base T. Note that the sums of conditional probabilities of {CtoA, GtoA, TtoA}, {AtoC, GtoC, TtoC}, {AtoG, CtoG, TtoG}, and {AtoT, CtoT, GtoT} all become 100% because they mean conditional base transition probabilities when an error occurs for each position.

(152nt) stitched oligo reads for decoding. By arranging the error distributions for each position, one can notice that there are some tendencies of transition probabilities (Fig. 4) and this result motivates us to use conditional probabilities of base transitions in position errors from entire sequenced data for LLR calculation.

Now, we explain how to derive conditional base transition probabilities from entire sequenced data. Let $Z[n] = \{1, 2, \ldots, n\}$ for a positive integer n and N_{read} be the number of stitched oligo reads after processing the PEAR algorithm for entire sequenced data. For each stitched oligo index $j \in Z[N_{\text{read}}]$, $g(j)$ denotes an index of the nearest encoded oligo sequence based on the edit distance. For each position index $i \in Z[152]$, we define $f_i(b^{(1)} \to b^{(2)}) = \Sigma_j I(x^{i,g(j)} = b^{(1)}, y^{i,j} = b^{(2)})$, where $x^{m,n}$ is a base of the mth position in the nth encoded oligo sequence, $y^{m,n'}$ is a base call of the mth position in the n'th stitched oligo read for $m \in Z[152]$, $n \in Z[18000]$, and $n' \in Z[N_{\text{read}}]$, $I(\cdot)$ denotes an indicator function, and $b^{(1)}, b^{(2)} \in \{\text{A, C, G, T}\}$ such that $b^{(1)} \neq b^{(2)}$.

Let $h(j)$ denote an index of the real encoded oligo index corresponding to the jth stitched oligo read. Then, we define the ith conditional base transition rates as follow:

$$P_i(x^{i,h(j)} = b^{(1)} | y^{i,j} = b^{(2)}) = \frac{\frac{f_i(b^{(1)} \to b^{(2)})}{N_i(b^{(1)})}}{\frac{f_i(b^{(1)} \to b^{(2)})}{N_i(b^{(1)})} + \frac{f_i(b^{(3)} \to b^{(2)})}{N_i(b^{(3)})} + \frac{f_i(b^{(4)} \to b^{(2)})}{N_i(b^{(4)})}},$$

where four distinct $b^{(1)}, b^{(2)}, b^{(3)}, b^{(4)} \in \{\text{A, C, G, T}\}$ and $N_i(b^{(k)}) = \Sigma_j I(x^{i,g(j)} = b^{(k)})$, that is, the number of $b^{(k)}$s in the ith position of encoded sequences corresponding to the stitched oligo reads for $k \in Z[4]$. The conditional base transition probabilities for entire sequenced data are given in Fig. 4.

Now, we explain how to calculate LLRs using Q-scores and channel statistics. After classifying stitched oligo reads into clusters based on seed information, we calculate LLR values for each cluster. Let $y_1^{i,j}$ and $y_2^{i,j}$ denote the 1st and 2nd bits of the ith position of the jth stitched oligo read for a mapping $\{\text{A, C, G, T}\} \to \{00, 01, 10, 11\}$, $i \in Z[152]$, and $j \in Z[N_{\text{read}}]$. Let $P_i^{j,b}(b') = P(x^{i,h(j)} = b' | y^{i,j} = b)$ for $b, b' \in \{\text{A, C, G, T}\}$ and we determine this probability for the case with $b = b'$ from the FASTQ file. The probability is derived as follow [30]:

$$P_i^{j,b}(b) = 1 - 10^{-\frac{Q_i^j(b)}{10}},$$

where $Q_i^j(b)$ is a Q-score value of the ith position of the jth stitched oligo read (and the basecall is b) in the FASTQ file. Although Q-scores do not represent all types of errors in DNA storage, it is natural to believe the higher Q-score value represents the higher reliability.

Then, we can calculate the other three probabilities of bases for a base call b in that position as follow:

$$P_i^{j,b}(b^{(k)}) = (1 - P_i^{j,b}(b)) \times P_i(x^{i,h(j)} = b^{(k)} | y^{i,j} = b),$$

where $k \in Z[3]$ and $b^{(k)} \in \{\text{A, C, G, T}\}$ such that $b^{(k)} \neq b$.

For example, suppose that base A is sequenced in the sth position of the tth stitched oligo read with a Q-score of 10 in the FASTQ file. Further, assume that we have error statistics of the sth position such that the conditional probabilities are $P_s(x^{s,h(t)} = \text{C} | y^{s,t} = \text{A}) = 50\%$, $P_s(x^{s,h(t)} = \text{G} | y^{s,t} = \text{A}) = 25\%$, $P_s(x^{s,h(t)} = \text{T} | y^{s,t} = \text{A}) = 25\%$. Then, we can obtain the probabilities as follows:

$$P_s^{t,\text{A}}(\text{A}) = 1 - 10^{-\frac{10}{10}} = 0.9,$$
$$P_s^{t,\text{A}}(\text{C}) = (1 - P_s^{t,\text{A}}(\text{A})) \times 0.5 = 0.05,$$

$$P_s^{t,\mathrm{A}}(\mathrm{G}) = (1 - P_s^{t,\mathrm{A}}(\mathrm{A})) \times 0.25 = 0.025,$$
$$P_s^{t,\mathrm{A}}(\mathrm{T}) = (1 - P_s^{t,\mathrm{A}}(\mathrm{A})) \times 0.25 = 0.025.$$

After all, we can calculate LLRs of bits for a base call b considering a mapping $\{\mathrm{A}, \mathrm{C}, \mathrm{G}, \mathrm{T}\} \rightarrow \{00, 01, 10, 11\}$ as follows:

$$\mathrm{LLR}(y_1^{i,j}) = \log \frac{P_i^{j,b}(\mathrm{A}) + P_i^{j,b}(\mathrm{C})}{P_i^{j,b}(\mathrm{G}) + P_i^{j,b}(\mathrm{T})},$$
$$\mathrm{LLR}(y_2^{i,j}) = \log \frac{P_i^{j,b}(\mathrm{A}) + P_i^{j,b}(\mathrm{G})}{P_i^{j,b}(\mathrm{C}) + P_i^{j,b}(\mathrm{T})},$$

which are from the previous work [31].

Based on the pre-determined seed information, we can discard stitched oligo reads with incorrect seed information (basecall) for LLR calculation and decoding. For oligo reads with the same seed information, we simply add LLRs of them for LT soft decoding. Since soft decoding is used only for the LT code and seed positions utilize the basecall information, LLR calculation is only needed for 128 payload positions. For RS parity, we exploit Q-scores and it will be explained in Section II-B.3.

2) Redecoding Based on RS Decoding Results: As aforementioned, we first proceed LT soft decoding and then we use the RS code for both error correction and error detection. After LT soft decoding using the parity check matrix H constructed from the seed information, we perform RS decoding for the decoded oligos. If RS decoding succeeds for all of the decoded oligos without any seed position errors, we recover the data from the doubly decoded oligos and declare an overall decoding success or failure. However, most of the cases do not satisfy the above conditions at once. Here, we propose the redecoding method that performs redecoding based on RS decoding results. For LT decoded oligos with RS decoding failures or any error detection in the seed positions, we discard the corresponding oligo clusters and then we proceed LT soft decoding again. This means that we do not use these wrong-RS-checked clusters into the decoding process, making a new H matrix with the same LLR calculation for the other oligo reads. Further, it is natural to regard the RS corrected oligos with seed position errors as wrongly corrected oligos because we only utilize oligo reads with the pre-determined seed value. Later, we actually find out that there are still some RS-corrected oligo clusters with the correction on the seed part, and they get removed accurately during our experiment.

In fact, this redecoding operation helps us to prevent the incorrect information from participating into the decoding process and this shows that it is better not to use the wrong information. After redecoding without these wrong-RS-checked clusters, there can also exist other oligo clusters that do not satisfy above conditions. Therefore, we repeat the redecoding method several times iteratively.

3) Iterative Decoding: Now, we introduce the proposed iterative soft decoding method shown in Algorithm 1. For inputs, we use stitched oligo reads with the correct length (152nt) after the PEAR algorithm [29] and we only use oligo reads with exactly the same seed information compared to the pre-determined seed values from the encoding process.

Algorithm 1 Proposed Iterative Soft Decoding Algorithm

Input: L_m randomly sampled stitched oligo reads after the PEAR algorithm with the pre-determined seed values, the maximum re-decoding number n_{re}, and LLRs and RS parity of those stitched oligo reads from a preprocessing step.

Output: 16050×256 information bits.

Initialization: $i = 0$.

Step 1) Perform soft decoding of 256 binary LT codes (= 128nt).

Step 2) For each decoded oligo, do RS decoding.

Step 3) If all RS decoding succeeds without errors in seed positions, find punctured information bits using decoded parity bits and skip the following steps.

Step 4) If $i = n_{\mathrm{re}}$, declare decoding failure and skip the following step.

Step 5) If $i < n_{\mathrm{re}}$ and oligos with RS decoding failure or seed position errors exist, remove those oligos and reconstruct H matrix. Then, go to Step 1) with initial LLRs for the other bits of oligos and $i \leftarrow i + 1$.

As a preprocessing step, we calculate LLRs for 1st and 2nd bits of each position for the payload parts. On the other hand, we need to use hard values for the RS part, and thus we determine the hard values of nucleotides from the maximum values of $\Pi_j P_i^j(\mathrm{A})$, $\Pi_j P_i^j(\mathrm{C})$, $\Pi_j P_i^j(\mathrm{G})$, and $\Pi_j P_i^j(\mathrm{T})$, where $P_i^j(b)$ is $P_i^{j,b'}(b)$ for a basecall b' of the ith position of the jth stitched oligo read corresponding to the target oligo cluster and $b, b' \in \{\mathrm{A}, \mathrm{C}, \mathrm{G}, \mathrm{T}\}$. Also, in Step 4 of Algorithm 1, we use $n_{\mathrm{re}} = 3$ for our redecoding count. In most cases, increasing the number of redecoding count n_{re} improves the decoding performance. However, considering both the improvement of decoding performance and computational complexity, we choose an adequate number $n_{\mathrm{re}} = 3$ heuristically. This makes the maximum number of iterative decoding be 3 in the worst case, however, this parameter can be adjusted if we exploit the increased computational complexity.

C. Comparison of Two Decoding Algorithms

For an easy understanding, we briefly summarize the proposed soft decoding method and previous hard decoding method [22], where the decoding performances of these two algorithms will be compared in the next section.

For the soft decoding algorithm, we need to calculate all the conditional probabilities of base transitions (A→C, A→G, A→T, C→A, C→G, C→T, G→A, G→C, G→T, T→A, T→C, and T→G) for each position, from position 1 to position 152. We calculate these statistics only with the correct-length (152nt) stitched oligo reads in the sequenced FASTQ files. For the clustering step, we collect the correct-length stitched oligo reads with the pre-determined seed values. For each cluster, we calculate the LLR values from position 17 to position 144 by using the calculations in Section II-B.1 and proceed the LT soft decoding. Note that the seed part is at position 1–16, the payload part is at position 17–144, and

TABLE I
EXPERIMENTAL RESULTS OF THE PROPOSED SOFT DECODING AND PREVIOUS HARD DECODING ALGORITHMS WITH 50 TRIALS EACH

	dataA		*dataB*	
Random sampling number	Proposed soft decoding	Previous hard decoding [22]	Proposed soft decoding	Previous hard decoding [22]
72k	16	11		
74k	41	22		
76k	45	38	16	10
78k	48	42	32	22
80k	**50**	45	43	32
82k	**50**	49	48	42
84k	**50**	49	49	48
86k	**50**	**50**	**50**	49
88k			**50**	**50**

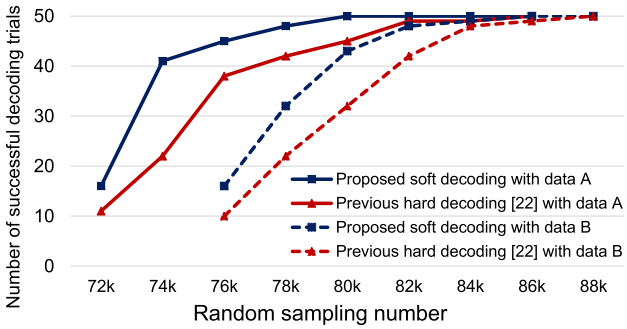

Fig. 5. Decoding results of the proposed soft decoding and previous hard decoding algorithms in both *dataA* and *dataB* with 50 trials each.

the RS parity part is at position 145–152. Then, we choose the RS symbols from position 145 to position 152 with the largest probabilities as in Section II-B.3. While we check the RS values, if there are some clusters with RS decoding failures or some clusters indicating the error correction at the seed position (position 1 to position 16), then change the LLR values of these clusters into 0s. We repeat the LT decoding and the RS checking steps up to three times, and if all the RS decoding succeeds, a decoding success is declared.

For the hard decoding algorithm, we do not need the statistics aforementioned. For the correct-length (152nt) stitched oligo reads of the sequenced FASTQ files, oligo reads that are the same or have one or two different bases are collected as the same cluster. In [22], the clusters are designed to have some distances apart each other, and thus clusters with differences in Hamming-distance smaller or equal to 79 are discarded. The size-1 clusters are sorted in the descending order based on the multiplication of the Q-scores, making smaller Q-scored clusters to be decoded later. Then, clusters without any errors are used first, clusters with any RS correction later, into the LT hard decoding, finally declaring a decoding success or failure. During these steps, we use the clusters with the pre-determined seed values for a fair comparison with the soft decoding algorithm.

III. RESULT

To show the superiority of the proposed decoding algorithm compared to the state-of-the-art algorithm, we employ two decoding algorithms for two sequencing data. One is the hard decoding method from the previous work [22] and the other is the proposed soft decoding method. For a fair comparison, we conduct these experiments with the assumption of knowing the seed information as we mentioned in Section II-A using $dataA$ and $dataB$.

A. Experimental Result

We randomly sample a small number of sequenced oligo reads first and add 2000 sequenced oligo reads for each sampling point to find a perfect recovery point. Note that L_m in Algorithm 1 is smaller or equal to the random sampling numbers in Table I due to the conditions of the exact oligo length (152nt) and exact seed position information. We conduct 50 trials for each random sampling and count the number of decoding success out of 50 trials. The result is summarized in Table I and shown in Fig. 5.

The result shows that the proposed method reaches the perfect recovery point at the lower random sampling number, implying a better decoding performance. Compared to the hard decoding method in [22], we could reduce 2000∼6000 sequenced oligo reads, which are 2.3%∼7.0% decrease of reading numbers. In fact, we make sure that a smaller number of sequenced oligo reads can imply a reduction of the reading cost in our experiment. Our experiments are conducted with the Illumina sequencing equipment, and one Illumina flow cell is always needed during each sequencing process. However, a smaller number of required oligo reads for decoding success means that we can use a smaller portion of the flow cell for reading our data. Unlike the Nanopore sequencing environment, it is not a concept of stopping sampling or sequencing; using the less part of the flow cell can reduce the reading cost in the point of a memory device with a large amount of data. We expect to use less substrates if we try our algorithm in the Nanopore sequencing environment and it is one of our goals in the future.

Also, our decoding algorithm is superior even though we do not introduce any other preprocessing methods during the clustering step. In contrast, the hard decoding algorithm [22] contains lots of preprocessing methods such as Hamming-distance based clustering and Hamming-distance based discarding wrong sequenced oligo reads. In order to figure out the impact of those preprocessing methods, we have conducted the hard decoding without using them. Then, it is noted that the perfect recovery point in $dataB$ is changed from 88000 to 90000 in Table I. If we add and modify the preprocessing methods as in [22] suitable for our soft decoding method, we may have more improved decoding performance than the current results shown in Table I. Note that we have total 18000 kinds of oligo sequences in the experiments and the differences of 2000∼6000 sequenced oligo reads are a deep portion of the sequencing environments. Also, the proposed soft decoding algorithm outperforms the state-of-the-art algorithm for both of the two sequencing data, which implies the consistency of our algorithm.

TABLE II
COMPARISON OF STATISTICAL PROPERTIES OF *dataA* AND *dataB* FOR STITCHED OLIGO READS WITH THE CORRECT LENGTH (152NT)

Statistics	$dataA$	$dataB$
Correct-length (152nt) PEAR passing rate	83.83%	78.64%
Average number of position errors per base	1.100×10^{-3}	9.604×10^{-4}
Average number of insertion & deletion errors per base	1.237×10^{-5}	1.744×10^{-5}
Average number of substitution errors per base	9.858×10^{-4}	8.352×10^{-4}

TABLE III
EFFECTS OF EACH DECODING TECHNIQUE IN THE PROPOSED SOFT DECODING METHOD IN *dataB* (LLR CALCULATION AND REDECODING)

LLR calculation	Chandak's method [18]		Proposed method	
Random sampling number	without redecoding	with redecoding	without redecoding	with redecoding
76k	0	2	0	**16**
78k	0	21	4	**32**
80k	9	38	10	**43**
82k	19	46	19	**48**
84k	23	48	21	**49**
86k	22	48	24	**50**

TABLE IV
EXPERIMENTAL RESULT OF THE PROPOSED SOFT DECODING AND PREVIOUS HARD DECODING ALGORITHMS WITH 50 TRIALS EACH FROM THE DATA IN THE CONSTRAINED POOL OF [22]

Experimental data of the constrained pool in [22]		
Random sampling number	Proposed soft decoding	Previous hard decoding [22]
72k	12	10
74k	28	22
76k	43	37
78k	49	43
80k	**50**	49
82k	**50**	**50**

B. Analysis

The main reason why the decoding performance of $dataA$ is better than $dataB$ at the same random sampling point is that the ratios of perfectly stitched oligo reads after the alignment algorithm are different for $dataA$ and $dataB$ as shown in Table II. We use the PEAR algorithm [29] for stitching the forward-reads and reverse-reads, and 83.83% of the sequenced oligo reads in $dataA$ are merged with the correct length of 152nt but 78.64% of the sequenced oligo reads in $dataB$ are merged with 152nt. Since we sample the sequenced oligo reads before the PEAR algorithm, $dataA$ would have more stitched oligo reads after the PEAR algorithm than $dataB$.

For error statistics, we collect all the correct-length stitched oligo reads and compare them with our original data in Table II. There exist differences between the error conditions for the two sequencing data and this would have effected the decoding performances as well. Also, this implies that for the same synthesized pool and the same sequencing method, the error statistics for each sequencing may be different a little. By the way, the error rates in both sequenced data are in an acceptable level of the Illumina sequencing environment.

To show the effects of each decoding technique in the proposed decoding algorithm, we carry out some more experiments with $dataB$ in Table III. For the proposed LLR calculation method, we compare it with Chandak's soft decoding method [18] based on the basecalling system. For the redecoding method, we conduct these two soft decoding methods in a way that incorporates and does not incorporate the redecoding system. Table III shows that the proposed soft decoding method makes the best performance and each technique is necessary for our contributions. Specifically, we find that without the redecoding system, the channels of the DNA storage system are very unstable and we could even find that Chandak's method makes the performance reversal in some points. For LLR calculation, it is noted that our method is superior in most cases. Therefore, we can claim that the soft decoding method using soft information (Q-scores extracted from the sequenced data and channel statistics) is very useful for the DNA storage system. Further, we note that the redecoding method makes the bigger contribution here compared to the new formula of the LLR calculation. Without the redecoding system, our performance is even worse than the previous work [22] in Table I.

IV. DISCUSSION AND FUTURE WORK

In Section III, we consider two non-constrained data ($dataA$ and $dataB$) due to the condition that the decoder knows the pre-determined seed values. However, one may wonder whether the performance improvement of the proposed decoding algorithm is valid only for non-constrained data. Therefore, we employ our decoding algorithm using the experimental data of the constrained pool in [22] with 50 trials, only adding an assumption that we know the seed information of the constrained pool (Table IV). We can see that our soft decoding algorithm also performs better than the hard decoding algorithm in the constrained pool. This implies that our decoding algorithm is superior regardless of homopolymer-run length and GC-content constraints.

Besides the aforementioned algorithms, we have tried various attempts to improve the decoding performance. As we once used a multiplication of Q-score values for ordering the oligo cluster groups in the previous work [22], we also try to use it in the soft decoding algorithm. The expression becomes:

$$P_j = \prod_{i=1}^{152} P_i^{j,b}(b) = \prod_{i=1}^{152} (1 - 10^{-\frac{Q_i^j(b)}{10}}),$$

and we also calculate the correlation between this P_j value and the number of position errors for each oligo cluster from the original encoded data in Fig. 6. This time, we find out that some of the high Q-scored oligo reads with large errors (marked with red circles in Fig. 6) are discarded during the redecoding process and many of them have insertion and deletion errors in them. It means that the proposed iterative soft decoding method can be effective for handling some insertion and deletion errors. Meanwhile, we try to discard some of the low Q-scored oligo reads during the decoding stage but the effect is relatively marginal (0 or 1 more decoding success

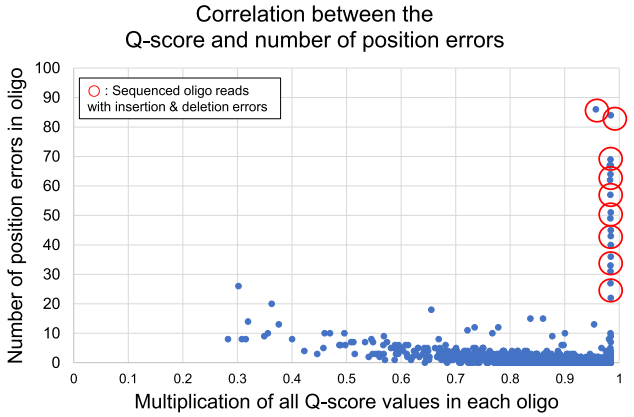

Fig. 6. Correlation between the Q-score and number of position errors in each oligo cluster. This graph is from one of the 50 trials at the successful decoding point, 86k of *dataB*.

out of 50 trials in Table I) and it is effective in large random sampling points but sometimes gives worse performance in low random sampling points. However, since there exists the correlation between the multiplication of Q-scores and the number of errors, discarding some of the low Q-scored oligo reads may be effective. Thus, we believe that figuring out this correlation to the decoding performance is an important future work.

Moreover, when we determine the LLR calculation formula, we observe that providing LLR reliability differentially for each oligo cluster according to the size gives a better decoding performance for some random sampling points. Specifically, we assign the lowest LLR values for the clusters size of one, a little larger LLR values for the clusters size of two, and even more larger LLR values for the clusters size of three and more and this eventually improves our decoding performance a bit. We think this is because information in bigger size clusters is more trustful than information in smaller size clusters. We could not find any other theoretical clues and experimental criteria to determine the amount of variables to be multiplied to reduce the LLR values, leaving the differential distribution for LLR calculation for the future work.

In fact, soft decoding may require the more cost in the perspective of the computing power and computational complexity compared to the hard decoding method. However, synthesis and sequencing processes in DNA storage need much more resources and time and thus using a little more computing operations does not have a significant impact in terms of the overall procedure. By taking a little cost in computing power, we can save a lot of reading cost in DNA sequences and may even affect to increase the code rate of the entire data. Since the main target of DNA storage is cold data, we believe this assumption is reasonable.

Indeed, there are several ECCs that have better performance than the LT code for soft decoding. In the fountain code, a RaptorQ code [32] has been developed as an advanced code from the LT code. Also, an LDPC code is well known to have one of the best error correcting performances in soft decoding [33]. Our coding structure of the LT code and RS code in [17] is designed for the hard decoding method, not caring about the soft decoding method. Therefore, we are sure that if we conduct another experiment with well designed ECCs to fit with calculated soft information, our proposed algorithm would improve the performance dramatically compared to existing decoding methods. Further, the advantage of using LT codes is that we can control some constraints on the homopolymer-run length and GC-content. However, as shown in [22], those constraints can also be managed through signal processing algorithms. Thus, other methods without using LT codes should be considered for DNA storage. The extreme case in the perspective of error correcting codes may be using ECCs with the best soft decoding performance dealing with position errors regardless of any other constraints.

V. Conclusion

In this paper, we proposed a new iterative soft decoding algorithm for the DNA storage system, where soft information can be obtained from FASTQ files and channel statistics. By using conditional base transition rates and Q-scores, we deduced four probabilities of bases for each position of stitched oligo reads and suggested a new way of LLR calculation which can be used in the error correction and detection in the DNA sequencing area. From calculated LLR values, soft decoding of LT codes with the redecoding system was used in the proposed decoding algorithm. The proposed decoding algorithm outperformed the state-of-the-art algorithm in [22] and the existing soft decoding algorithm in [18] in terms of the required number of sequenced oligo reads, which means that the reading cost of the sequencing process can be reduced by our proposed decoding algorithm.

References

[1] G. M. Church, Y. Gao, and S. Kosuri, "Next-generation digital information storage in DNA," *Science*, vol. 337, no. 6102, p. 1628, Sep. 2012.
[2] Y. Dong, F. Sun, Z. Ping, Q. Ouyang, and L. Qian, "DNA storage: Research landscape and future prospects," *Nat. Sci. Rev.*, vol. 7, no. 6, pp. 1092–1107, Jun. 2020.
[3] P. J. A. Cock, C. J. Fields, N. Goto, M. L. Heuer, and P. M. Rice, "The Sanger FASTQ file format for sequences with quality scores, and the Solexa/Illumina FASTQ variants," *Nucleic Acids Res.*, vol. 38, no. 6, pp. 1767–1771, Apr. 2010.
[4] C. Xu, C. Zhao, B. Ma, and H. Liu, "Uncertainties in synthetic DNA-based data storage," *Nucleic Acids Res.*, vol. 49, no. 10, pp. 5451–5469, Jun. 2021.
[5] A. El-Shaikh, M. Welzel, D. Heider, and B. Seeger, "High-scale random access on DNA storage systems," *NAR Genomics Bioinf.*, vol. 4, no. 1, Jan. 2022, Art. no. lqab126.
[6] M. G. Ross et al., "Characterizing and measuring bias in sequence data," *Genome Biol.*, vol. 14, no. 5, pp. 1–20, 2013.
[7] J. J. Schwartz, C. Lee, and J. Shendure, "Accurate gene synthesis with tag-directed retrieval of sequence-verified DNA molecules," *Nature Methods*, vol. 9, no. 9, pp. 913–915, Sep. 2012.
[8] B. Cao, X. Zhang, J. Wu, B. Wang, Q. Zhang, and X. Wei, "Minimum free energy coding for DNA storage," *IEEE Trans. Nanobiosci.*, vol. 20, no. 2, pp. 212–222, Apr. 2021.
[9] B. Cao, X. Ii, X. Zhang, B. Wang, Q. Zhang, and X. Wei, "Designing uncorrelated address constrain for DNA storage by DMVO algorithm," *IEEE/ACM Trans. Comput. Biol. Bioinf.*, vol. 19, no. 2, pp. 866–877, Mar. 2022.
[10] J. Wu, Y. Zheng, B. Wang, and Q. Zhang, "Enhancing physical and thermodynamic properties of DNA storage sets with end-constraint," *IEEE Trans. Nanobiosci.*, vol. 21, no. 2, pp. 184–193, Apr. 2022.
[11] Q. Yin, Y. Zheng, B. Wang, and Q. Zhang, "Design of constraint coding sets for archive DNA storage," *IEEE/ACM Trans. Comput. Biol. Bioinf.*, vol. 19, no. 6, pp. 3384–3394, Nov. 2022.

[12] N. Goldman et al., "Towards practical, high-capacity, low-maintenance information storage in synthesized DNA," *Nature*, vol. 494, no. 7435, pp. 77–80, Feb. 2013.

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

[14] J. Bornholt, R. Lopez, D. M. Carmean, L. Ceze, G. Seelig, and K. Strauss, "A DNA-based archival storage system," in *Proc. 21st Int. Conf. Architectural Support Program. Lang. Operating Syst. (ASPLOS)*, 2016, pp. 637–649.

[15] L. Organick et al., "Random access in large-scale DNA data storage," *Nature Biotechnol.*, vol. 36, no. 3, pp. 242–248, Mar. 2018.

[16] S. M. H. T. Yazdi, R. Gabrys, and O. Milenkovic, "Portable and error-free DNA-based data storage," *Sci. Rep.*, vol. 7, no. 1, pp. 1–6, Jul. 2017.

[17] Y. Erlich and D. Zielinski, "DNA fountain enables a robust and efficient storage architecture," *Science*, vol. 355, no. 6328, pp. 950–954, Mar. 2017.

[18] S. Chandak et al., "Improved read/write cost tradeoff in DNA-based data storage using LDPC codes," in *Proc. 57th Annu. Allerton Conf. Commun., Control, Comput. (Allerton)*, Sep. 2019, pp. 147–156.

[19] W. H. Press, J. A. Hawkins, S. K. Jones, J. M. Schaub, and I. J. Finkelstein, "HEDGES error-correcting code for DNA storage corrects indels and allows sequence constraints," *Proc. Nat. Acad. Sci. USA*, vol. 117, no. 31, pp. 18489–18496, Aug. 2020.

[20] Y. Wang, M. Noor-A-Rahim, J. Zhang, E. Gunawan, Y. L. Guan, and C. L. Poh, "Oligo design with single primer binding site for high capacity DNA-based data storage," *IEEE/ACM Trans. Comput. Biol. Bioinf.*, vol. 17, no. 6, pp. 2176–2182, Nov. 2020.

[21] B. Cao, X. Zhang, S. Cui, and Q. Zhang, "Adaptive coding for DNA storage with high storage density and low coverage," *npj Syst. Biol. Appl.*, vol. 8, no. 1, pp. 1–12, Jul. 2022.

[22] J. Jeong et al., "Cooperative sequence clustering and decoding for DNA storage system with fountain codes," *Bioinformatics*, vol. 37, no. 19, pp. 3136–3143, Oct. 2021.

[23] A. Rasool, Q. Jiang, Y. Wang, X. Huang, Q. Qu, and J. Dai, "Evolutionary approach to construct robust codes for DNA-based data storage," *Frontiers Genet.*, vol. 14, p. 415, Mar. 2023.

[24] S. Mirjalili, "Moth-flame optimization algorithm: A novel nature-inspired heuristic paradigm," *Knowl.-Based Syst.*, vol. 89, pp. 228–249, Nov. 2015.

[25] S. Kang et al., "Generative adversarial networks for DNA storage channel simulator," *IEEE Access*, vol. 11, pp. 3781–3793, 2023.

[26] M. Luby, "LT codes," in *Proc. 43rd Annu. IEEE Symp. Found. Comput. Sci.* Washington, DC, USA: IEEE Computer Society, Feb. 2002, p. 271.

[27] Y. Han and W. E. Ryan, "Low-floor decoders for LDPC codes," *IEEE Trans. Commun.*, vol. 57, no. 6, pp. 1663–1673, Jun. 2009.

[28] R. Palanki and J. S. Yedidia, "Rateless codes on noisy channels," in *Proc. Int. Symp. Inf. Theory (ISIT)*, Jul. 2004, p. 37.

[29] J. Zhang, K. Kobert, T. Flouri, and A. Stamatakis, "PEAR: A fast and accurate Illumina paired-end reAd mergeR," *Bioinformatics*, vol. 30, no. 5, pp. 614–620, Mar. 2014.

[30] *bcl2fastq Conversion User Guide*, Illumina Inc., San Diego, CA, USA, 2013, pp. 23–24.

[31] X. Lu et al., "Error rate-based log-likelihood ratio processing for low-density parity-check codes in DNA storage," *IEEE Access*, vol. 8, pp. 162892–162902, 2020.

[32] A. Shokrollahi, "Raptor codes," *IEEE Trans. Inf. Theory*, vol. 52, no. 6, pp. 2551–2567, Jun. 2006.

[33] D. J. C. MacKay and R. M. Neal, "Near Shannon limit performance of low density parity check codes," *Electron. Lett.*, vol. 33, no. 6, pp. 457–458, 1997.