# An Encoding Table Corresponding to ASCII Codes for DNA Data Storage and a New Error Correction Method HMSA

Xuncai Zhang and Fuzhen Zhou

***Abstract*— DNA storage stands out from other storage media due to its high capacity, eco-friendliness, long lifespan, high stability, low energy consumption, and low data maintenance costs. To standardize the DNA encoding system, maintain consistency in character representation and transmission, and link binary, base, and character together, this paper combines the encoding method with ASCII code to construct an ASCII-DNA encoding table. The encoding method can encode not only pure text information but also audio and video information and satisfies the GC content constraint and the homopolymer constraint, with the encoding density reaching 1.4 bits/nt. In particular, when encoding textual information, it directly skips the binary conversion process, which reduces the complexity of encoding, and increasing the encoding density to 1.6 bits/nt. In order to solve the problem of errors in sequences, under the influence of heuristic algorithms, this paper proposes a new error correction method (HMSA) by combining minimum Hamming distance, multiple sequence alignment, and encoding scheme. It can correct not only substitution, insertion, and deletion errors in Reads but also consecutive errors in Reads. It greatly improves the utilization of the Reads and avoids the waste of resources. Simulation results show that the recovery rate of Reads increases with the increasing number of sequencing times. When the number of erroneous bases in a 150nt sequence reaches 5nt, the error correction rate can exceed 96% by sequencing the base sequence only 10 times regardless of whether the errors are consecutive or not. Additionally, the HMSA error correction method is applicable to all coding schemes for lookup code table types.**

***Index Terms*— DNA storage, ASCII code, encoding table, Hamming distance, HMSA.**

## I. Introduction

NOWADAYS we are in the era of the great explosion of data, by the current trend of development, it is expected that in 2025 the amount of data generated globally throughout the year will be 175ZB, which is equivalent to the daily amount of data generated will reach 491EB [1]. In the face of a huge amount of data, traditional storage media such as CD, Floppy Disks and USB Flash Drives are not able to guarantee the capacity and durability of the storage [2]. DNA stands out as a natural storage medium among many storage media by a variety of advantages. These include (1) High stability and long life: under particular conditions of storage, DNA has a half-life of about 521 years, and if DNA is stored in silica, it can be stored continuously for more than 2 million years [3], [4]. (2) High storage density and high capacity: the theoretical capacity of DNA digital information storage is 455 EB bytes of data per gram of single-stranded DNA [5].

Due to the rapid development of DNA amplification (PCR) and sequencing technologies (Sanger sequencing, first generation-sequencing, third-generation sequencing, single-cell sequencing) [6], large-scale storage of DNA data has become a breakthrough storage technology [7], [8] in the background of today's era of data explosion.

The concept of DNA data storage was introduced as early as 1959 by the physicist Feynman, who in his book "There's Plenty of room at the bottom (data storage): An Invitation to Enter a New Field of Physics" talks about the future construction of microcosm-like objects (including living beings) with the ability to store data [9]. The main process by which DNA stores and reads information consists of five parts: encoding and storage, synthesis, amplification, sequencing, and decoding. The flowchart is shown in Fig. 1.

In the early days when the basic encoding rules ([00:A;01:C;10:G;11:T]) were used, a large number of errors were produced in the base sequences during synthesis and sequencing. For example, Antkowiak et al. [10] found that more than 99.9% of the 15 million sequences retrieved had errors, and Organic [11] reported that insertion and deletion errors are generated during the synthesis and sequencing of more than 88% of Reads in third-generation sequencing methods. In addition, base loss in large segments of the sequence may also occur during DNA synthesis, amplification, and sequencing. It has been shown that GC content and homopolymer length in DNA sequences are the main reasons affecting the generation of these errors [12]. Therefore, to reduce the generation of these errors, most of the previous studies introduced some constraints in designing the coding scheme, such as GC content between 40% and 60%; homopolymer

Manuscript received 1 November 2023; revised 31 December 2023; accepted 10 January 2024. Date of publication 22 January 2024; date of current version 3 April 2024. This work was supported by the National Natural Science Foundation of China under Grant 62072417 and Grant 62102374. *(Corresponding author: Xuncai Zhang.)*

The authors are with the College of Electrical and Information Engineering, Zhengzhou University of Light Industry, Zhengzhou 450002, China (e-mail: zhangxuncai@pku.edu.cn; 2848395178@qq.com).

This article has supplementary downloadable material available at https://doi.org/10.1109/TNB.2024.3356522, provided by the authors.

Digital Object Identifier 10.1109/TNB.2024.3356522

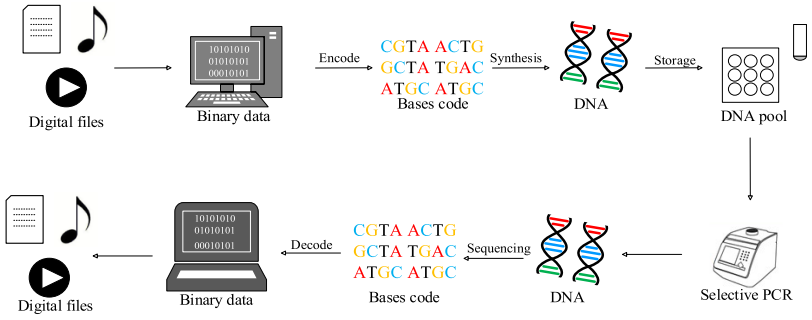


Fig. 1. Flowchart of DNA data storage.

length of no more than 4; and minimizing the occurrence of secondary structures in the DNA sequences [13], and so on. Earlier in 2012, Church et al. [5] proposed an encoding scheme: encoding 1 bit corresponding to 1 base (A or C for 0, G or T for 1) to encode the information in a variety of ways, which is used to avoid extreme GC content, homopolymer number, and secondary structure. Goldman et al. [14] proposed a trinary rotational encoding scheme in 2013, which uses rotational encoding to cleverly remove all homopolymers from DNA sequences after compressing the data into trinary information. In 2015, Grass et al. [3] combined the Galois Field of GF(47) with the triplet of bases and proposed an encoding algorithm that avoids homopolymers of lengths greater than 3nt. Meanwhile, in terms of error correction, Grass et al. [3] introduced the first error-correcting code by adding vertical and horizontal Reed-Solomon coding for correcting errors with DNA sequences and errors between DNA sequences. In 2016 Blawat et al. [15]. proposed a forward error correction encoding scheme where a two-rule coding method encodes bases at different positions. Finally, different filter rules are used to select the encoded base segment as a way to satisfy the homopolymer constraints. In 2017, Erlich and Zielinski [16] proposed a DNA Fountain algorithm based on Luby Transform code, which filters out DNA sequences that do not satisfy the constraints through a filtering mechanism. The error correction machine RS (Reed-Solomon) code was also introduced. However, the fundamental problems of Luby Transform [17] make the fountain codes risk decoding failure when dealing with specific binaries. In 2022, Ping et al. [18] proposed a Yin-Yang code based on the Chinese Taoist idea of yin and yang. This coding method uses two rules to codify 2 bits of binary into one base, constructs filtering rules about satisfying GC content, homopolymer, and free energy [19], and filters out DNA sequences that meet the constraints.

Reducing the occurrence of errors is only on one hand, limited by the present level of science and technology, the occurrence of errors is inevitable, so it is necessary to incorporate error correction mechanisms after the DNA sequence. Existing error correction mechanisms in terms of redundancy can be roughly divided into two categories. The first category is error correction methods that do not add redundancy: most of them are Multiple Sequence Alignment (MSA) and its extensions. There are many algorithms currently used for MSA, which can be broadly categorized as progressive alignment algorithms, iterative algorithms, heuristic algorithms, machine learning algorithms, and divide and conquer algorithms [20]. Progressive alignment algorithms are a method that helps in alignment by constructing a guide tree [21]. However, due to the lack of accuracy of progressive alignment algorithms, iterative algorithms were introduced to progressive alignment to solve the problem [22]. Heuristic algorithm is a generalized iterative algorithm. However, unlike traditional iterative algorithms, heuristic algorithms, due to their randomized nature, focus on providing fast results with an acceptable range of accuracy rather than providing near-perfect solutions. The application of machine learning in MSA is still in its infancy. Divide and conquer is also widely used in MSA, the algorithm divides a set of sequences into multiple sets of sequences for processing, which greatly improves the efficiency of the algorithm [23]. These methods often require a large number of Reads, and due to the synchronization problem between sequences when insertion and deletion errors occur, these methods may also require high-performance clustering algorithms to solve the problem. For example, Xie et al. [24] and Antkowiak et al. [10]. used clustering algorithms to align DNA sequences. And in 2021 Zan et al. [25] researchers were inspired by the divide and conquer algorithms to improve the MSA idea and designed a three-layer error correction method for textual information. The second category of methods to increase redundancy can be further divided into: (1) Adding physical redundancy, such as Goldman et al. [14] used DNA sequences that overlapped by 75% for error correction, which increased redundancy by a factor of four, and Bornholt et al. [26] added redundancy by a factor of two by storing two DNA sequences after they were XOR. (2) Adding logical redundancy, including traditional error correction codes such as Hamming codes [27], RS codes [28], and LDPC codes [29], [30]. These conventional error correction codes can correct substitution errors in DNA sequences but are powerless in the face of insertion errors and deletion errors. Insertions and deletions of sequences result in inconsistent lengths of DNA sequences, and when using RS codes for error correction, the DNA sequences with incorrect lengths are usually removed. However, this practice greatly reduces the utilization of sequencing read lengths, resulting in a great waste of resources. Therefore, it is necessary to develop an error correction method with high utilization of sequencing read length.

TABLE I
RULE A CODING SCHEME

| Rule A | | | | |
|---|---|---|---|---|
| Binary | 00 | 01 | 10 | 11 |
| Base | C | G | T | A |

TABLE II
RULE A CODING SCHEME

| Rule B | | | | |
|---|---|---|---|---|
| Binary | 00 | 01 | 10 | 11 |
| Base | AA | AC | AG | AT |
| | CC | CG | CT | CA |
| | GG | GT | GA | GC |
| | TT | TA | TC | TG |

In this paper, inspired by Blawat, we created an ASCII-DNA encoding table for DNA data storage that satisfies GC content and homopolymer constraints and corresponds one-to-one with the ASCII code, with a coding density of 1.4 or 1.6 bits/nt.

In terms of the types of error correction methods, to improve the utilization of sequencing Read lengths, influenced by heuristic algorithms, this paper chooses to improve the first type of error correction methods, combining the minimum Hamming distance with MSA, and proposes a new error correction method (HMSA), which is focused on providing a result with an acceptable accuracy range, rather than providing near-perfect error correction results. HMSA can solve the synchronization problem of sequences, avoiding the waste of resources of a large number of sequencing Reads while not using clustering algorithms. Meanwhile, simulation results show that the HMSA correction method can effectively correct not only substitution, insertion, and deletion errors in DNA sequences but also some of the mixed errors. In addition, the HMSA correction method applies to all coding schemes of lookup code table types.

## II. ENCODING AND DECODING METHODS

### A. Constructing an ASCII-DNA Encoding Table

Inspired by Blawat et al. [15], in combination with ASCII codes, we constructed an ASCII-DNA encoding table that corresponds one-to-one with ASCII codes. The encoding table not only satisfies the GC content and homopolymer constraints but also, similar to ASCII codes, standardizes the coding system, ensuring that characters are represented and transmitted consistently across different computers and communication devices. In constructing the coding table, we used mapping rules similar to those of Meinolf Blawat. However, to satisfy the GC content constraints of the encoded base sequences, we modified the mapping rules, as shown in TABLE I and TABLE II. In particular, the binary in Rule B corresponds to the four possible mapped base combinations.

To be associated with the ASCII code, we first convert the stored information into binary information, then operate every 7-bit slice on the binary information, and finally add a "0" to the first bit of each slice, so that all the split 7-bit binary fragments become 8 bits binary fragments, and are all in the set {00000000, 00000001, 00000010, ……,01111111}, so that we establish a connection with the ASCII code table.

So when constructing the ASCII-DNA encoding table, we take {00000000, 00000001, 00000010, ……,01111111} as inputs and encode each of these 128 binary fragments into corresponding bases. Taking one of the fragments as an example, first, we group the first 6 bits of the 8 bits binary fragment two by two, and the 7th and 8th are each divided into a group, using $G = \{g_1, g_2, g_3, g_4, g_5\}$ to represent it. The encoded base fragments are represented using $B = \{b_1, b_2, b_3, b_4, b_5, b \in [A, T, G, C]\}$. Use Rule A to encode the binary information $g_1, g_2, g_3$ into the base information $b_1, b_2, b_4$. Use Rule B to encode the binary information $g_4, g_5$ into the base information $b_3, b_5$. Since there are 4 choices for Rule B, The encoded base fragment $B$ is also then combined in 4 ways. To meet the requirements of homopolymers and GC content, three screening conditions are developed in this paper, and the specific steps are shown in Fig. 2. The screening conditions are as follows:

Condition 1: Check if there are three consecutive identical bases or if the last two bases are identical in the four base combinations after the code. If present, the base combination will be removed; if not, go to condition two.

Condition 2: Check whether the GC content of the base combination at that 5 position is 40% or 60%. If yes, place the result in the encoding table of the corresponding order and go to Condition 3; if no, keep the result and go to Condition 3.

Condition 3: Check if the ASCII-DNA encoding table is empty. If it is empty, put the result retained in condition two into the corresponding ASCII-DNA encoding table; if it is not empty, output the ASCII-DNA encoding table.

After passing the three filtering conditions, the filtered results can constitute two ASCII-DNA encoding tables. When using these two encoding tables to encode the same file, their GC contents are around 45% and 55%, respectively, both of which satisfy the constraints of GC contents. In this paper, the encoding table with GC content of around 55% is chosen for the subsequent simulation experiments. Some of the encoding tables are shown in TABLE III.

### B. Encoding Method

In general, the coding steps when coding all types of files are as follows:

Step 1: Convert the input file to 8 bits binary.

Step 2: Group the binary information in 7-bit slices, and if the last group is less than 7-bit, add "0" at the end until it reaches 7-bit.

Step 3: Adding a "0" to the first bit of the binary information in each group ensures that all grouped binary fragments are within {00000000, 00000001, 00000010, ……, 01111111}, i.e., within the ASCII-DNA encoding table.

Step 4: Encode the binary information processed in Step 3 into bases using the ASCII-DNA encoding table. The encoding process is shown in Fig. 2. Since we add "0" before the 7-bit binary and "0" at the end of the last group, it is equivalent to adding redundant bits. As for the "0" added at the end, with the increasing size of the information, the added

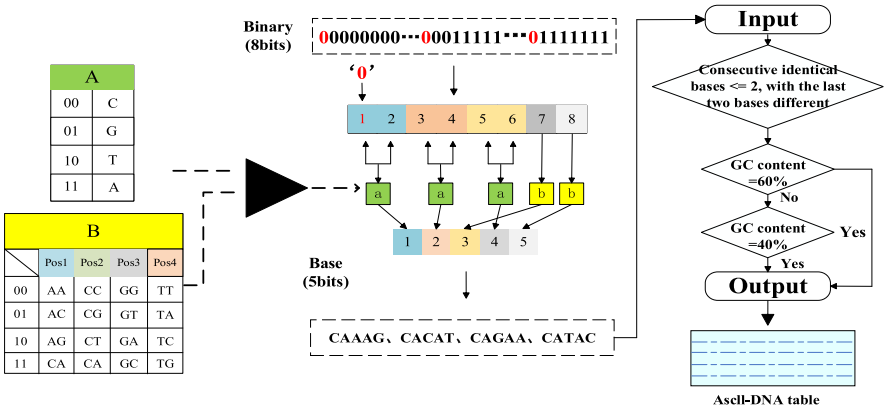


Fig. 2. Flowchart of ASCII-DNA encoding table construction.

redundancy is almost negligible. Therefore, the calculated encoding density is approximately 1.39 bit/nt. The encoding flow is shown in Fig. 3. Example of encoded character: "abc", general method of encoding using the ASCII-DNA encoding table:

(1) Binary conversion: '011000010110001001100011'.
(2) Slice grouping, add 0 at the end: ['0110000', '1011000', '1001100', '0110000'].
(3) Add 0 to the first position: ['00110000', '01011000', '01001100', '00110000'].
(4) Use ASCII-DNA encoding table to encode: 'CAACAGGATAGCCACCAACA'.

Since we are constructing the encoding table directly associated with the ASCII code, in particular, when encoding textual information, we can use the coding table to directly encode the textual information as a sequence of bases.

For example, to encode a pure character: "abc", directly use the ASCII-DNA encoding table to encode: 'GTCCGGTACGGTACT'. Compared with the general method, Step 1, Step 2, and Step 3 are skipped, which greatly reduces the time complexity and space complexity. Since no redundancy is added when encoding plain text information, it is equivalent to encoding 8 bits of valid binary information with 5 bases, and the coding density is increased to 1.6 bits/nt. Some of the encoding tables are shown in TABLE III. The full encoding table is in the supplemental materials.

## C. Decoding Method

Generally speaking, the decoding process is the inverse process of encoding. For all types of storage files, the unified decoding method is as follows: in the first step, the Reads obtained from sequencing are sliced every 5nt, and an ASCII-DNA encoding table is used to decode them. In the second step, the binary sequence obtained is sliced every 8 bits and the first 0 of each group is removed. In the third step, all the segments are merged to obtain the binary data. Finally, the binary data is converted into a storage file. Particularly, for text-based files, the binary conversion process can be skipped, the post-sliced (every 5 bases) base fragments were decoded into textual information directly using the ASCII-DNA encoding table.

## D. Methods of Error Correction

*1) Introduction to the HMSA:* Since the encoding method proposed in this paper is codebook type, the base fragments after grouping every 5 slices in the correct Read should all be in the ASCII-DNA encoding table. In other words, assuming that no error occurs in the Read, the minimum Hamming distance values calculated from the base segments grouped in the Read and the base segments in the coding table should all be 0. Once the minimum Hamming distance is not 0, we can determine that an error has occurred in the Read, and we can determine where the error occurred. Based on this idea, this paper develops a new error correction method (HMSA) by combining the minimum Hamming distance and MSA.

The HMSA error correction method can be summarized in the following four steps:

Step 1: Classify Reads into three groups (substitution error group, insertion error group, and deletion error group) by comparing the lengths between the Reads of sequencing results and the correct sequences.

Step 2: Group the Reads in the three groups one by one in slices, and calculate the minimum hamming distance between the base fragments in each group and the base fragments in the ASCII-DNA encoding table. Find the first group whose minimum hamming distance is not 0 (the position where the error occurred).

Step 3: Correct the substitution error group, insertion error group, and deletion error group respectively.

Step 4: The corrected Reads are aligned, put together, and error corrected again using MSA.

The three scenarios for 3) are as follows:

Case 1: Correct the DNA sequences in the substitution error group, the Reads in the substitution error group are defined as when the length of the Read is equal to the length of the design DNA sequence, 150nt, and an error occurs during the decoding process, then the Read is put into the substitution error group. Since there is no sequence synchronization problem for the Reads in the substitution error group, we can directly adopt the method of MSA. However, the number of sequencing required for sequence comparison may be higher, and the corresponding sequencing cost is higher. Therefore, we add the calculation of the minimum hamming distance to process Reads before sequence comparison, that is HMSA, the error

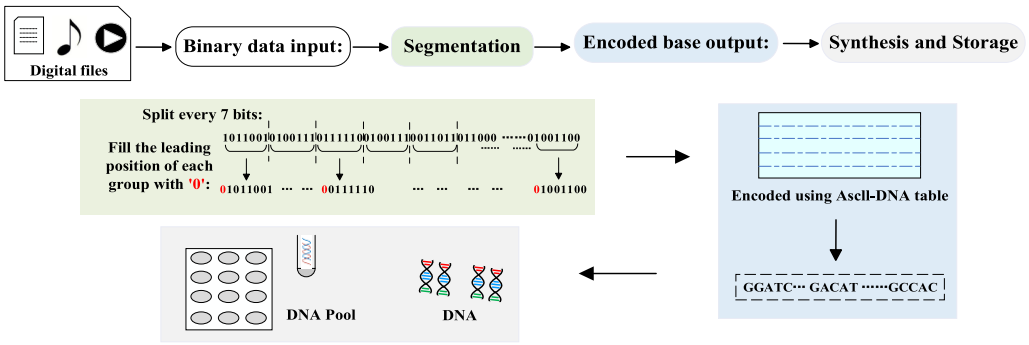


Fig. 3. Flowchart of encoding using ASCII-DNA encoding table.

correction method proposed in this paper, which not only can correct a part of the errors in Reads in advance, and then reduce the number of sequencing times required for MSA, but also improve the error correction ability of HMSA.

Taking any base sequence Read in the substitution group as an example for processing, we first slice and group every 5 bases of Read, where we denote the base fragments in the ASCII-DNA encoding table by $O_{Table} = \{o_1,\ o_2,\ o_3, \ldots,\ o_j, \ldots,\ o_{128} | j \in [1, 128],\ j \in N^+\}$. Each set of base fragments after slicing is represented by the set $O_{sub} = \{o_1,\ o_2,\ o_3, \ldots,\ o_i, \ldots,\ o_{26} | j \in [1, 26],\ i \in N^+\}$ using the list $H_{min} = [h_1,\ h_2,\ h_3,\ \ldots,\ h_i, \ldots,\ h_{26} | i \in [1, 26],\ i \in N^+, h_i \geq 0]$ to represent the set of minimum hamming distances computed for each set of base fragments from the base fragments in $O_{Table}$. The base fragment $o_i$ corresponding to the minimum hamming distance $h_i$ not 0 is selected as the position where the substitution error is located. Flip $o_i$ to the base fragment $o_j, (i \neq j)$ corresponding to the minimum hamming distance calculated. Until all the values in the list $H_{min}$ are 0, all the base fragments in all the Reads after slicing and grouping are all in $O_{Table}$, all base fragments $o_i$ in $O_{sub}$ are all in $O_{Table}$. At this point the $O_{sub}$ of the slice grouping is merged is the initial completion of the corrected DNA sequence, as shown in Fig. 4a.

Case 2: Correct the DNA sequence in the insertion error group. The Reads in the insertion error group are defined as when the length of the Read is greater than the length of the design DNA sequence, 150nt, and an error occurs during the decoding process, the Read is put into the insertion error group. It should be noted that the Reads in the insertion error group will have sequence synchronization problems when using the MSA error correction method, there will be no way to align the bases at the same position, resulting in the MSA being unable to correct the erroneous bases. The following are specific ways to solve the sequence synchronization problem existing in the insertion error group using HMSA.

Take any one of the base sequences Read in the insertion group as an example, first slice and group the bases in Read every 5nt (the last group is less than 5nt, and is temporarily counted as a group). Calculate the minimum hamming distance between each group of sliced base fragments and the base fragments in the ASCII-DNA encoding table $O_{Table}$. Slices grouped into each set of base fragments are indicated using the set $O_{ins} = \{o_1,\ o_2,\ o_3, \ldots,\ o_i, \ldots,\ o_m | m = \lceil len\,(Read)\,/5 \rceil, i \leq m, i \in N^+\}$, The set of minimum hamming distances inserted into the error group is represented using the list $H_{min} = [h_1,\ h_2,\ h_3,\ \ldots,\ h_i, \ldots,\ h_m | m = \lceil len\,(Read)\,/5 \rceil, i \leq m, i, m \in N^+, h_i \geq 0]$ When computing the minimum hamming distance $h_i$, if the last set of $o_m$ is less than 5nt, the minimum hamming distance $h_m$ cannot be computed, and temporarily make $h_m = 2$ until the truth value of the minimum hamming distance $h_m$ is computed when $o_m$ becomes 5nt in the continuous correction process. Here is the correction process, find the base fragment $o_i$ corresponding to the first non-zero $h_i$ from the list $H_{min}$ to start the error correction (this group of 5nt bases $o_i$ is not in $O_{Table}$). Also, find the corresponding $o_j, (i \neq j)$ in $O_{Table}$ when calculating $h_i$, and delete one of the bases in $o_i$ (the first base where $o_i$ is different from $o_j$). After deletion, merge all base fragments $o_i$ in $O_{ins}$. Re-group the merged base sequences in slices of every 5nt, and repeat the above process to start the cycle of error correction until two conditions are met all values in the list $H_{min}$ are 0 (after the Read slices are grouped all base fragments are in $O_{Table}$), and the length of the corrected Read is equal to 150nt, and the correction of a single Read is complete. If the two cannot be satisfied at the same time, the next Read will be processed. It should be noted that the above correction process does not necessarily achieve perfect error correction, because some of the base fragments in the $O_{Table}$ are very similar, there may be cases where the incorrect base fragment is still in the encoding table after the correction is completed, or the corrected base fragment is not the correct base fragment, is another base fragment in the ASCII-DNA encoding table.

For example, the correct fragment is $o_i, o_{i+1}$ = CTAGA, CCACA, if after an insertion error occurs: $o_i, o_{i+1}$ = CTATGA, CCACA, since CTATG is in the ASCII-DNA encoding table, the error correction starts from ACCAC, and 'A' will be deleted when correcting. and after correction is completed $o_i, o_{i+1}$ = CTATG, CCACA This case does not achieve error correction for this fragment, but we have corrected all the wrong base fragments into the encoding table, which will not prevent the decoding of later base fragments. The final result of the output is the base sequence after correcting the insertion error. This is shown in Fig. 4b.

Case 3. The Reads in the delete error group are defined as the Reads that are placed in the delete error group when the length of the Read is less than the length of the design DNA sequence, 150nt, and an error occurs during the decoding process. The Reads in the deletion error group have the same

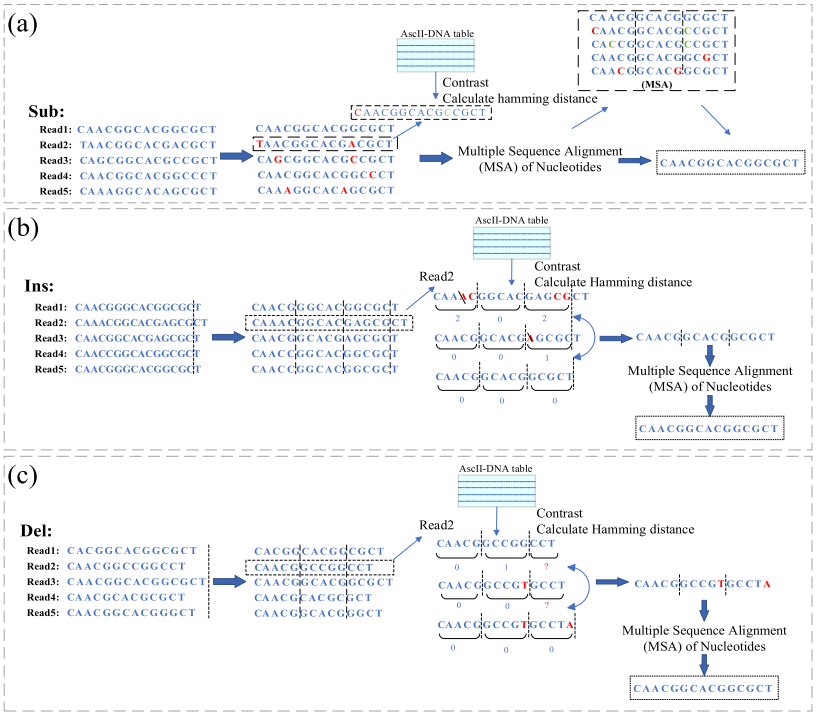


Fig. 4. Corrections for the three error types. (a) Flow of HMSA when dealing with substitution error groups, (b) flow of HMSA when dealing with insertion of error groups, (c) flow of HMSA when dealing with deletion of error groups.

TABLE III
ASCII-DNA ENCODING TABLE

| Binary | ASCII | Base | Binary | ASCII | Base | Binary | ASCII | Base | Binary | ASCII | Base |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 01010100 | T | GGAGA | 01010101 | U | GGAGC | 01010110 | V | GGCGT | 01010111 | W | GGAGT |
| 01011000 | X | GGATA | 01011001 | Y | GGATC | 01011010 | Z | GGATG | 01011011 | [ | GGCTA |
| 01011100 | \ | GGCAC | 01011101 | ] | GGAAC | 01011110 | ^ | GGAAG | 01011111 | _ | GGAAT |
| 01100000 | ` | GTACA | 01100001 | a | GTCCG | 01100010 | b | GTACG | 01100011 | c | GTACT |
| 01100100 | d | GTAGA | 01100101 | e | GTAGC | 01100110 | f | GTCGT | 01100111 | g | GTAGT |

synchronization problem of sequences when using MSA error correction. To solve the synchronization problem, here is what we do to correct the Reads in the deletion error group using HMSA.

Take any one of the base sequences Read in the deletion error group as an example, in the same way, first group the bases in Read in slices grouped every 5nt. The base slices in the ASCII-DNA encoding table are represented as $O_{Table}$. After slicing and grouping the Read where deletion errors occur, each set of bases is denoted by the set $O_{del} = \{o_1, o_2, o_3, \ldots, o_i, \ldots, o_m | m = \lceil len\,(Read)\,/5 \rceil, i \leq m, i \in N^+\}$, the minimum hamming distance calculated by deleting the error group is still represented using the list $H_{min}$. Error correction starts by finding the $o_i$ corresponding to the first 0 that is not from the list $H_{min}$. Also, find the corresponding $o_j (i \neq j)$ in the $O_{Table}$ at the time of calculating $h_i$, and insert a base in $o_i$ (so that the 5nt base fragment where the inserted base in $o_i$ is located is more similar to $o_j$, $h_i$ is decreasing or equal to 0). The number of bases in $o_i$ after the insertion becomes 6nt, merging all the base fragments $o_i$ in $O_{Read}$. Re-group the merged Read in every 5nt slices, repeat the above process, and again find the $o_i$ corresponding to the first $h_i$ that is not 0 from the list $H_{min}$ to start the error correction until the individual Read correction is completed when all the values in the list $H_{min}$ are satisfied to be 0 and the length of the corrected Read is equal to 150nt, and the next Read is processed if both of them are not satisfied. If both are not satisfied, the next Read will be processed until all the Reads are processed. It is important to note that the above correction process also does not determine the accuracy of the error correction. HMSA has a similar problem when dealing with deletion errors as it does when dealing with insertion errors.

For example, the correct segment is CACCG, after a deletion error occurs: CACG, and after correction: CACGG. In this case, HMSA does not implement error correction for this segment, but HMSA corrects the incorrect base segment into the ASCII-DNA encoding table, which no longer affects the decoding of later base segments. After all the Reads have been corrected, the output is the corrected deleted base sequence. This is shown in Fig. 4c.

Finally, all the DNA sequences output from Case1, Case2, and Case3 were put together, and using the multiple sequence alignment method, all the corrected individual base sequences were aligned, and the base with the largest percentage of bases at the same position was selected to be placed on a new base sequence. Since this paper is a codebook type of encoding method, subject to the limitation of base similarity in the encoding table, after correcting the Read using HMSA (the Read after finalizing the multiple sequence matching), subject to the limitation of base similarity in the encoding table, there is still a possibility that the base fragment therein may not be in the ASCII-DNA encoding table, which leads to errors in the decoding process. To prevent this problem, in this paper, if conditional statements are added to the decoding algorithm to decode the base fragments that are not in the encoding table as fixed GTAGC when decoding.

### E. The Approach of HMSA to Handling Mixed Error Types

When HMSA handles mixed errors, since substitution errors do not affect sequences with synchronization problems, when insertion errors are mixed with substitution errors in a Read, assign the Read to the insertion errors group for error correction. When a mix of deletion errors and substitution errors exists in Read, assign Read to the deletion error group for error correction. When insertion and deletion errors are mixed, HMSA handles them as follows:

(1) When the number of insertion errors in the Read is greater than the number of deletion errors, i.e., the length of the Read is greater than the length of the designed sequence, such Read is put into the insertion error group for error correction in the same way as Case2.
(2) When the number of insertion errors in the Read is equal to the number of deletion errors, i.e., the length of the Read is equal to the length of the design sequence, such Read is put into the substitution error group for error correction in the same way as Case1.
(3) When the number of insertion errors in the Read is less than the number of deletion errors, i.e., the length of the Read is less than the length of the designed sequence, such Read is put into the deletion error group for error correction in the same way as Case3.

Of course, the effect of HMSA in dealing with mixed errors is not as good as that of correcting sequences with single error types. In the following simulation experiments, we will mention the error correction ability of HMSA when facing mixed errors.

## III. DNA Fragment Design and Analysis of Simulation Experiments

### A. DNA Fragment Design

In this paper, a novel named Wandering Earth in English (about 368KB) is used as the text material for simulation experiments. Fig. 5 shows the DNA fragments designed in this paper. 150nt of encoded data information, that is the payload. The novel is encoded as 14318 DNA sequences with a length of 150nt. 25nt index bits can encode indexes

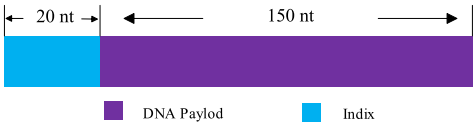


Fig. 5. Fig DNA sequence design.

from 0 to 999999. Since the number of DNA sequences is 14318, choosing 25nt not only results in a waste of resources, but also reduces the density of encoding, so we choose 20nt index bits (encoding index bits from 0 to 9999), and the textual material is stored in two test tube containers can meet the storage requirements. In this paper, the encoding and decoding as well as the performance analysis are simulated using the Python language to achieve.

### B. Properties of DNA Sequences

In analyzing the DNA sequence properties of the encoded, this paper compares the mapping encoding method of the ASCII-DNA encoding table with the basic encoding method (00-A, 01-C, 10-G, 11-T) and six encoding methods such as Church, Goldman, Grass, Blawat, DNA Fountain and Yin - Yang Code. The number of DNA sequences that satisfy the GC content constraints and the number of maximal homopolymers after encoding the same textual material, Wandering Earth, were compared using different methods of encoding. The results of the comparison are shown in Fig. 6.

From Fig. 6a, it is easy to see that the DNA sequences encoded using the coding method proposed in this paper are almost always in the 40% to 60% interval. This is inextricably linked to the characteristics of the ASCII-DNA encoding table constructed in this paper, in which the GC share of base fragments in the ASCII-DNA encoding table is almost always 40% or 60%. In terms of homopolymer constraints, as shown in Fig. 6b, the maximum homopolymer of all DNA sequences is controlled at 3nt. The coded DNA sequences satisfy both GC content constraints and homopolymer constraints, which indicates that the use of ASCII-DNA encoding table coding can achieve storage-friendly coding mode, which not only facilitates DNA synthesis and polymerase chain reaction (PCR), but also reduces the number of erroneous bases to some extent.

### C. Analysis of HMSA Error Correction Performance

We simulated actual sequencing results containing errors by performing base mutations (insertions, deletions, and substitutions) of a unit number of bases on the coded base sequences. In the simulation experiments, each simulated sequencing result has equal error types and number of errors. By increasing the number of simulated sequencing (i.e., increasing the number of erroneous sequences generated under the same conditions), we observed and compared the error correction performance of HMSA under different sequencing numbers. To test the error correction performance of HMSA in the face of three different types of errors, the error types in the simulated sequencing Reads are all of one type and non-contiguous.

For Reads with substitution errors as shown in Fig. 7a and Fig. 7b, this paper compares the average error correction performance of two methods, MSA and HMSA, when dealing

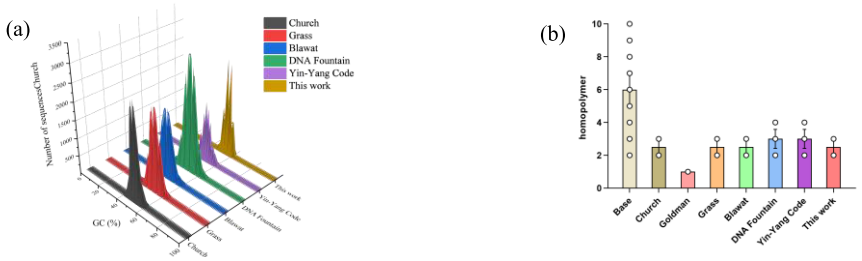


Fig. 6. Comparison of DNA sequence properties. (a) comparison of GC content (b) comparison of homopolymer sizes of DNA sequences.

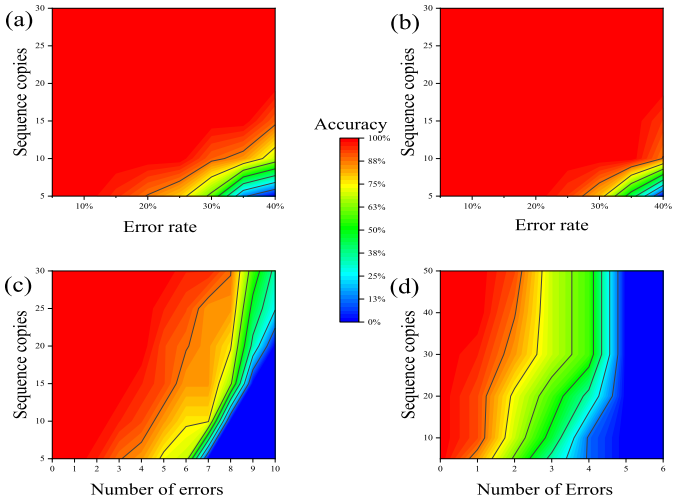


Fig. 7. HMSA error correction performance. (a) The recovery rate of decoding after correcting substitution errors using MSA for different sequencing copies. (b) Recovery rate of decoding after correcting substitution errors using HMSA. (c) Recovery rate of decoding after correcting insertion errors using HMSA. (d) Recovery rate of decoding after correcting deletion errors using HMSA.

with Reads, firstly, the error correction ability of both MSA and HMSA improves with the increase of sequencing times. The comparison shows that our improvement effect on MSA is very significant, especially when the number of sequencing is only 5, HMSA can still reach a 100% recovery rate when dealing with Reads with a 25% error rate, while MSA can only deal with Reads with 20% error rate under the premise that the number of sequencing is 5 and the recovery rate is 100%. During the simulation experiments, we found that HMSA is almost 0% effective in correcting Reads with insertion and deletion errors when the error rate is at 8%. To collect more experimental data, we changed the variable from error rate to the number of errors, which is also in line with the actual situation of the simulated sequencing results. The performance of HMSA in correcting Reads with insertion errors is shown in Fig. 7c. From the figure, it can be seen that 2 insertion errors can be perfectly corrected when the number of sequencing times is 5. However, when facing

Reads with deletion errors, as shown in Fig. 7d, the performance of HMSA for error correction is not satisfactory when the number of sequencing times is very small.

## D. Simulation Analysis

In terms of simulated DNA sequencing results, although we did not do wet experiments in this paper, we found through

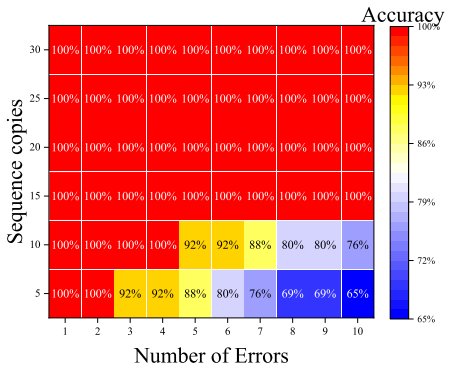


Fig. 8. Error correction performance of HMSA when correcting discontinuous type errors.

reading a large amount of literature that, in fact, among all the sequencing read lengths, excluding DNA sequences with large deletions, the lengths of about 90% of the sequencing reads are within $\pm 3$ of the length of the correct Reads [15]. So in the simulation experiments, for individual simulated sequencing results, the number of errors is more realistic compared to the error rate. In the next simulation experiments, the simulated sequencing Reads are composed of random groups of three types of errors, The results of each simulation experiment in the figure below are the mode of recovery rates obtained from our simulation of 20 simulation experiments. The reason for choosing mode is that the mode of the recovery rate represents the results with a higher likelihood of occurrence, which is more descriptive of the performance of the error correction of the HMSA in general.

*1) Error Correction Performance of HMSA Under Non-Continuous Errors (Three Error Types):* Fig. 8 shows the error correction performance of HMSA when there are three error groups in Reads. As the number of sequencing times increases, the error correction performance of HMSA improves, and in the course of continuous simulation experiments, we find that HMSA performs better in error correction when the number of Reads in the substitution error group accounts for a large percentage of the total number of Reads. This is consistent with the HMSA performance analysis in the previous section, where HMSA is highly tolerant of errors when correcting the substitution error group.

*2) Error Correction Performance of HMSA Under Continuous Errors (Three Error Types):* Previous studies have shown that consecutive insertion and deletion errors occur frequently in sequencing Reads, and that the longer the length of consecutive errors, the more difficult error correction becomes. For

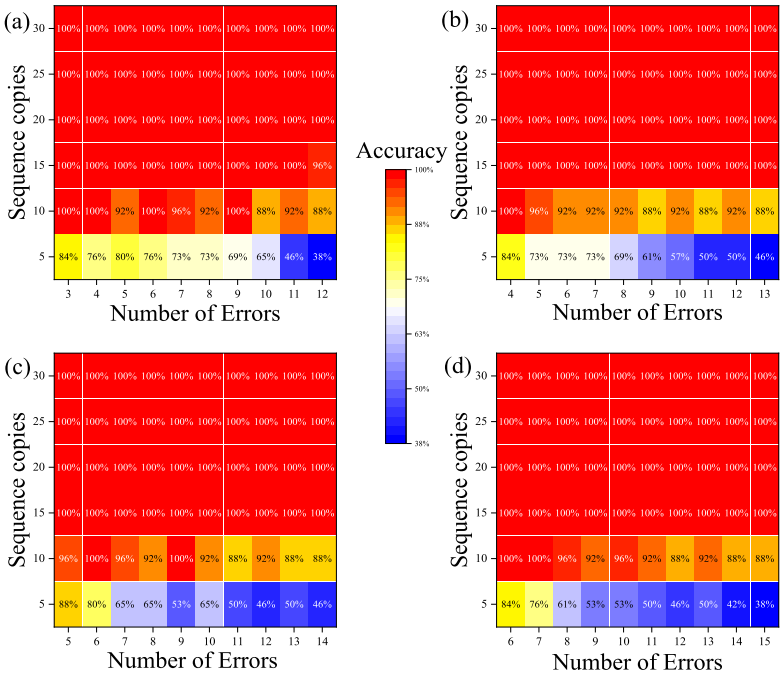


Fig. 9. Error correction performance of HMSA when correcting consecutive types of errors (substitution, insertion, or deletion). (a) three consecutive errors, (b) four consecutive errors, (c) five consecutive errors, (d) six consecutive errors.

TABLE IV
COMPARISON OF THIS PAPER WITH PREVIOUS RESEARCH METHODS

| Study | Methods | Correct methods | Indels correction | Storage file type | Information density(bits/nt) |
|---|---|---|---|---|---|
| Church et al. | A or C for 0, G or T for 1 | No | No | All | 1 |
| Goldman et al. | Ternary rotation coding | Repetition | No | All | 1.58 |
| Grass et al. | Triplet binding of GF (47) and bases | RS | No | All | 1.78 |
| Erlich et al. | Fountain | Fountain | No | All | 1.98 |
| Press et al. | HEDGES | HEDGES、RS | Yes | All | 1 |
| Xue et al[32] | DNA-XL | Levenshtein code | Yes | All | 1.32 |
| Chen et al | YYC | RS | No | All | 1.95 |
| Zan et al | Encoding table | Three-level error correction | Yes | Txt | 1.06 |
| This work | Encoding table | HMSA | Yes | All | 1.4or1.6 |

TABLE V
CORRECTED RECOVERY RATE AT DEL = INS

| | 10 | 20 | 30 |
|---|---|---|---|
| Ins=Del=1nt | 80% | 88% | 92% |
| Ins=Del=2nt | 46% | 74% | 42% |
| Ins=Del=3nt | 30% | 38% | 46% |

example, HEDGES [31] can handle consecutive deletions but cannot tolerate more than 2 consecutive insertions. Therefore, we further simulated to explore the impact of consecutive errors on the decoding performance of HMSA for different lengths.

Fig. 9 shows the error correction performance of HMSA when the number of consecutive errors is 3, 4, 5, and 6, respectively. It is easy to find that HMSA's ability to correct continuous errors is increasing with the number of sequencing times when HMSA is faced with continuous errors. With a sequencing number of 15, HMSA can make the recovery rate reach about 100%.

In most previous studies, due to the inability to solve the sequence synchronization problem caused by insertion and deletion errors, non-correct length Reads are usually discarded, which may lead to a non-negligible loss of DNA storage in terms of cost and time. However, the use of the HMSA error correction method can significantly alleviate this problem, and the error correction performance of HMSA is also very reliable on continuous errors.

*3) Performance of HMSA Error Correction Under Mixed Errors:* When insertion errors are mixed with substitution errors and deletion errors are mixed with substitution errors, the error correction ability of HMSA decreases with the increasing number of substitution errors under the same number of simulated sequencing.

Here we mainly analyze the three cases when insertion errors are mixed with deletion errors:

TABLE VI
CORRECTED RECOVERY RATE AT DEL > INS

| | 10 | 20 | 30 |
|---|---|---|---|
| Del=2nt , Ins=1nt | 53% | 69% | 84% |
| Del=3nt , Ins=1nt | 30% | 57% | 80% |
| Del=4nt , Ins=1nt | 23% | 27% | 42% |

(1) When the number of insertion errors is equal to the number of deletion errors, HMSA corrects them by categorizing them into the substitution error group. The simulation data are shown in TABLE V. From the simulation experiments, we find that the recovery rate is increasing with the increase of sequencing times. However, there are anomalies in the table, which are because the number of erroneous bases in our simulated sequencing results is specific rather than random, and the proportion of erroneous bases at the same position maybe larger after the correction is completed.
(2) When the number of insertion errors is greater than the number of deletion errors, HMSA corrects them by categorizing them into the insertion error group. During the correction process, specific bases need to be deleted while ensuring that the length of the sequence is correct. However, it is not possible to guarantee the correct length of the sequence during the constant deletion iterations, so there is a contradiction on this point, resulting in a mixed error where HMSA is unable to correct the situation where the number of insertion errors is greater than the number of deletion errors. Thus, it is easy to see here that the deletion of bases has a greater impact on error correction.
(3) When the number of deletion errors is greater than the number of insertion errors, HMSA corrects them by categorizing them into the deletion error group. The simulation data are shown in TABLE VI. From the simulation experiments, we found that the recovery rate increases as the number of sequencing times increases. In addition, the correction of the deletion group requires the insertion of specific bases, which has less impact on the error correction than the case where the number of insertion errors is larger than the number of deletion errors.

### E. Comparisons With Other Methods

TABLE IV compares the encoding schemes, error correction methods, types of errors corrected, types of files stored, and encoding densities of this paper with those of other studies. Compared to many other studies, this study encodes all types of files while satisfying the GC content and homopolymer constraints and maintaining the encoding density. Also, in terms of error correction HMSA can correct insertion and deletion errors.

## IV. CONCLUSION

DNA has a density of 6-7 orders of magnitude and a hundred times longer lifetime compared to current media, and the error problem that exists in DNA storage is the most difficult to solve. To solve the error problem, the encoding scheme proposed in this paper introduces the ASCII-DNA encoding table, which corresponds to ASCII code one-to-one, and satisfies the GC content constraint and homopolymer constraint, thereby reducing the occurrence of erroneous bases. Simultaneously, it can encode all types of files (text, image, audio, and video) and standardize the coding system, with a coding density of 1.4 bits/nt. In particular, for text files, direct mapping is possible, eliminating the need to convert text files to binary during encoding and decoding, resulting in a coding density of 1.6 bits/nt. Regarding the error correction method, the proposed HMSA error correction method offers several advantages compared to other error correction methods. Firstly, it does not require the addition of any redundancy. Secondly, HMSA, as a heuristic algorithm, corrects erroneous bases in the sequence into decodable bases and then corrects errors through sequence comparison. Most importantly, HMSA solves the sequence synchronization problem that cannot be solved by MSA, enabling correction not only for substitution, insertion, and deletion errors but also for some mixed errors. In this way, Reads of different lengths generated during sequencing can also be effectively corrected, which greatly improves the utilization of Reads and thus reduces the cost overhead of the sequencing process. This paper conducts a comprehensive study on the error correction capability of HMSA for DNA storage. Simulation results demonstrate that HMSA can achieve full recovery at low error rates and low sequencing counts. The error correction performance of HMSA is also very reliable at moderate error rates with increasing sequencing counts. Furthermore, HMSA is a generalized method, equally applicable to the coding method of any codebook type, providing a new direction for error correction in the coding methods of all codebook types in the future.

## DATA AVAILABILITY

All data are available in the main text. The datasets generated during and/or analyzed during the current study are available from the corresponding author on reasonable request.

## COMPETING INTERESTS

The authors declare no competing interests.

## REFERENCES

[1] J. F. Gantz, D. Reinsel, and J. Rydning, "The US datasphere: Consumers flocking to cloud," in *Analyze The Future*. America: International Data Corporation, 2019.
[2] S. Ebrahimi, R. Salkhordeh, S. A. Osia, A. Taheri, H. R. Rabiee, and H. Asadi, "RC-RNN: Reconfigurable cache architecture for storage systems using recurrent neural networks," *IEEE Trans. Emerg. Topics Comput.*, vol. 10, no. 3, pp. 1492–1506, Jul. 2022.
[3] 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.
[4] V. Zhirnov, R. M. Zadegan, G. S. Sandhu, G. M. Church, and W. L. Hughes, "Nucleic acid memory," *Nature Mater.*, vol. 15, no. 4, pp. 366–370, Apr. 2016.
[5] G. M. Church, Y. Gao, and S. Kosuri, "Next-generation digital information storage in DNA," *Science*, vol. 337, no. 6102, p. 1628, 2012.

[6] J. Hu, Y. Zhong, and X. Shang, "A versatile and scalable single-cell data integration algorithm based on domain-adversarial and variational approximation," *Briefings Bioinf.*, vol. 23, no. 1, Jan. 2022, Art. no. bbab400.

[7] Y. Cevallos et al., "A brief review on DNA storage, compression, and digitalization," *Nano Commun. Netw.*, vol. 31, Mar. 2022, Art. no. 100391.

[8] S. M. H. T. Yazdi, H. M. Kiah, E. Garcia-Ruiz, J. Ma, H. Zhao, and O. Milenkovic, "DNA-based storage: Trends and methods," *IEEE Trans. Mol., Biol. Multi-Scale Commun.*, vol. 1, no. 3, pp. 230–248, Sep. 2015.

[9] R. P. Feynman, "There's plenty of room at the bottom [data storage]," *J. Microelectromech. Syst.*, vol. 1, no. 1, pp. 60–66, Mar. 1992.

[10] P. L. Antkowiak et al., "Low cost DNA data storage using photolithographic synthesis and advanced information reconstruction and error correction," *Nature Commun.*, vol. 11, no. 1, p. 5345, Oct. 2020.

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

[12] S.-J. Park, Y. Lee, and J.-S. No, "Iterative coding scheme satisfying GC balance and run-length constraints for DNA storage with robustness to error propagation," *J. Commun. Netw.*, vol. 24, no. 3, pp. 283–291, Jun. 2022.

[13] X. Li, S. Zhou, and L. Zou, "Design of DNA storage coding with enhanced constraints," *Entropy*, vol. 24, no. 8, p. 1151, Aug. 2022.

[14] 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.

[15] M. Blawat et al., "Forward error correction for DNA data storage," *Proc. Comput. Sci.*, vol. 80, pp. 1011–1022, 2016.

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

[17] F. Lu, C. H. Foh, J. Cai, and L.-T. Chia, "LT codes decoding: Design and analysis," in *Proc. IEEE Int. Symp. Inf. Theory*, 2009, pp. 2492–2496.

[18] Z. Ping et al., "Towards practical and robust DNA-based data archiving using the yin–yang codec system," *Nature Comput. Sci.*, vol. 2, no. 4, pp. 234–242, Apr. 2022.

[19] 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.

[20] Y. Zhang, Q. Zhang, J. Zhou, and Q. Zou, "A survey on the algorithm and development of multiple sequence alignment," *Briefings Bioinf.*, vol. 23, no. 3, May 2022, Art. no. bbac069.

[21] S. F. Altschul and D. J. Lipman, "Trees, stars, and multiple biological sequence alignment," *SIAM J. Appl. Math.*, vol. 49, no. 1, pp. 197–209, Feb. 1989.

[22] R. C. Edgar, "MUSCLE: Multiple sequence alignment with high accuracy and high throughput," *Nucleic Acids Res.*, vol. 32, no. 5, pp. 1792–1797, Mar. 2004.

[23] Q. Zhan, Y. Fu, Q. Jiang, B. Liu, J. Peng, and Y. Wang, "SpliVert: A protein multiple sequence alignment refinement method based on splitting-splicing vertically," *Protein Peptide Lett.*, vol. 27, no. 4, pp. 295–302, Mar. 2020.

[24] R. Xie, X. Zan, L. Chu, Y. Su, P. Xu, and W. Liu, "Study of the error correction capability of multiple sequence alignment algorithm (MAFFT) in DNA storage," *BMC Bioinf.*, vol. 24, no. 1, pp. 1–11, Mar. 2023.

[25] X. Zan et al., "A hierarchical error correction strategy for text DNA storage," *Interdiscipl. Sci., Comput. Life Sci.*, vol. 14, no. 1, pp. 141–150, Mar. 2022.

[26] J. Bornholt, R. Lopez, D. M. Carmean, L. Ceze, G. Seelig, and K. Strauss, "A DNA-based archival storage system," in *Proc. ASPLOS*, 2016, pp. 637–649.

[27] U. Kumar and B. Umashankar, "Improved Hamming code for error detection and correction," in *Proc. 2nd Int. Symp. Wireless Pervasive Comput.*, 2007, pp. 498–500.

[28] L. C. Meiser et al., "Reading and writing digital data in DNA," *Nature Protocols*, vol. 15, no. 1, pp. 86–101, Jan. 2020.

[29] L. Deng et al., "Optimized code design for constrained DNA data storage with asymmetric errors," *IEEE Access*, vol. 7, pp. 84107–84121, 2019.

[30] 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.

[31] 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.

[32] T. Xue and F. C. Lau, "Construction of GC-balanced DNA with deletion/insertion/mutation error correction for DNA storage system," *IEEE Access*, vol. 8, pp. 140972–140980, 2020.