Computational and Structural Biotechnology Journal 21 (2023) 4469–4477

Contents lists available at ScienceDirect

Computational and Structural Biotechnology Journal

journal homepage: www.elsevier.com/locate/csbj

Research Article

DBTRG: De Bruijn Trim rotation graph encoding for reliable DNA storage ☆

Yunzhu Zhao [a,1], Ben Cao [b,1], Penghao Wang [a], Kun Wang [a], Bin Wang [a,*]

[a] The Key Laboratory of Advanced Design and Intelligent Computing, Ministry of Education, School of Software Engineering, Dalian University, Dalian, Liaoning 116622, China

[b] School of Computer Science and Technology, Dalian University of Technology, Dalian, Liaoning 116024, China

ARTICLE INFO

Dataset link: https://github.com/TalentZ111/DNA-Storage_DBTRG

Keywords:
DNA storage
De Bruijn Trim graph
Dynamic binary sequence

ABSTRACT

DNA is a high-density, long-term stable, and scalable storage medium that can meet the increased demands on storage media resulting from the exponential growth of data. The existing DNA storage encoding schemes tend to achieve high-density storage but do not fully consider the local and global stability of DNA sequences and the read and write accuracy of the stored information. To address these problems, this article presents a graph-based De Bruijn Trim Rotation Graph (DBTRG) encoding scheme. Through XOR between the proposed dynamic binary sequence and the original binary sequence, k-mers can be divided into the De Bruijn Trim graph, and the stored information can be compressed according to the overlapping relationship. The simulated experimental results show that DBTRG ensures base balance and diversity, reduces the likelihood of undesired motifs, and improves the stability of DNA storage and data recovery. Furthermore, the maintenance of an encoding rate of 1.92 while storing 510 KB images and the introduction of novel approaches and concepts for DNA storage encoding methods are achieved.

1. Introduction

By 2025, the total amount of data generated annually in the world will reach 175 ZB, growing at a compound annual rate of 61%, as predicted by the International Data Corporation [1]. Although big data has immense value, it also creates problems with data storage. To address the urgent need for a new storage medium and accommodate large amounts of data, DNA, a high-storage and high-density storage medium, has emerged as an answer [2,3]. Encoding, synthesizing, sequencing, and decoding are the four processes of information storage using DNA molecules [4]. Utilizing proper encoding and error correction is essential for the long-term preservation and accurate reading of digital information in DNA storage [5].

After digital information has been encoded, DNA data storage stores artificial DNA sequences [6]. Information may be kept for thousands of years in DNA storage because it is highly dense, stable in the long term, and able to store substantial information in a small amount of space [7,8]. Ceze et al. [9] used DNA as a digital information storage medium and proposed a novel method to encode digital data into DNA sequences. They also discussed how to achieve fast reading, random access, and error handling. Their research results showed that DNA storage systems are high-density, long-term stable, and large-capacity digital storage solutions with broad application prospects. Goldman et al. [10] suggested an affordable approach to converting information files into DNA sequences using Shannon's information encoding and file segregation techniques, which can help store more information without affecting data integrity and readability while providing high data density. Banal et al. [11] suggested a brand-new technique for encoding file data into DNA sequences and storing them in silicon capsules. Their method labels single-stranded DNA barcodes on the surface to represent file metadata. Users can simply choose the file set they desire by selecting a specific barcode without having to enlarge it. The study also made use of fluorescent sorting technologies to access massive molecular databases with great sensitivity and selectivity. The highest storage

☆ This work is supported by 111 Project (No. D23006), the National Natural Science Foundation of China (Nos. 62272079, 61972266, 61802040), Liaoning Revitalization Talents Program (No. XLYC2008017), Natural Science Foundation of Liaoning Province (Nos. 2021-MS-344, 2021-KF-11-03, 2022-KF-12-14), the Postgraduate Education Reform Project of Liaoning Province (No. LNYJG2022493), the Dalian Outstanding Young Science and Technology Talent Support Program (No. 2022RJ08), the Innovation and Entrepreneurship Team of Dalian University (No. XQN202008).

* Corresponding author.
E-mail addresses: zhaoyunzhu7@gmail.com (Y. Zhao), bencaocs@gmail.com (B. Cao), penghaowang926@gmail.com (P. Wang), 17862833837@163.com (K. Wang), wangbinpaper@gmail.com (B. Wang).
[1] These authors are joint first authors.

https://doi.org/10.1016/j.csbj.2023.09.004
Received 20 June 2023; Received in revised form 4 September 2023; Accepted 5 September 2023
Available online 11 September 2023

capacity of DNA-based storage currently is less than 1 GB, which is significantly less than most electronic data-storage systems, despite DNA having theoretically better storage densities, longer storage durations, and cheaper maintenance costs [12]. Furthermore, existing synthesis and sequencing technologies for DNA necessarily result in a variety of storage uncertainties [13]. Consequently, there remain several challenges in large-scale implementations of DNA storage. This includes finding ways to efficiently synthesize and sequence DNA, enhance the security and dependability of the storage, and address technical issues related to storing and retrieving massive amounts of data [14]. These challenges need to be gradually addressed through in-depth research and technological innovation to achieve widespread application of DNA storage. Adopting an appropriate and efficient encoding strategy can not only improve base utilization but also decrease the probability of errors during the DNA synthesis and sequencing [15,16].

Efficient DNA storage encoding schemes and error-correcting algorithms improve the stability and accuracy of DNA storage systems, thus effectively ensuring the long-term preservation of digital information [17–19]. Erlich et al. [20] proposed a fountain encoding scheme that segments binary files into droplets, and then they performed Luby transformation and screening to obtain DNA sequences that met certain conditions. Wang et al. [21] proposed an encoding scheme that reduces redundancy by hiding addressing, thereby reducing the number of base pairs required for storage. This encoding scheme has good adaptability to indicators with low self-similarity and stable local GC content, and it reduces the error rate in DNA synthesis and sequencing while improving base utilization and optimizing the reliability and efficiency of DNA storage. Ping et al. [22] proposed a yin-yang codec system (YYC), which encodes two binary bits into a base and uses yin-yang rules for encoding. This encoding method has good robustness and is highly compatible with synthesis and sequencing techniques, thus helping obtain more reliable DNA sequences. Qu et al. [23] proposed an efficient Clover clustering algorithm that uses a tree data structure to search for specific intervals between DNA sequences and can quickly cluster large amounts of DNA sequences into fewer groups, which promotes the application and implementation of DNA storage technology. DNA encoding was an essential innovation in DNA storage; it facilitates the preservation of data with as few base sequence errors as feasible under restrictions [24]. However, DNA data storage still faces many challenges related to technology and cost, such as errors in sequence synthesis, sequencing, storage, and processing, which need to be overcome to better apply DNA storage to practical scenarios [25]. Meiser et al. [26] proposed a method of using DNA sequences for information storage and retrieval and pointed out that molecular errors also occurred in DNA storage, mainly due to synthesis, sequencing, storage, and processing. To solve these problems, researchers have proposed various methods, such as the Marine Predator algorithm [27] and the maximum independent set method [28], to design DNA sequences that meet specific constraints. Low-density parity-check (LDPC) codes and Reed-Solomon (RS) codes have been used to improve error correction. The emergence of new technologies such as nanopore sequencers and assembly techniques is expected to help achieve more convenient and efficient reading of and access to DNA data storage, which can promote the development and application of DNA storage technology [29,30]. Yin et al. [27] proposed the Marine Predator algorithm, which constructs sequences that meet constraints by raising the lower bounds of the encoding sets. Luncasu et al. [28] used the frequency matrix game graph to design sequences that satisfied combinatorial constraints, which helped improve the reliability and efficiency of DNA storage. Cao et al. [31] combined the frequency matrix game graph with the maximum independent set to satisfy constraint conditions and achieve a more efficient DNA sequence design. As DNA storage technology continues to develop, error-correcting capabilities are also constantly improving. Chen et al. [32] encoded two images and a video clip into an artificial chromosome, used LDPC codes for error correction, and evaluated and compared the error-correcting capabilities of the LDPC and RS codes through computer simulations. Welzel et al. [33] proposed an encoding scheme that supports variable-sized encoding sequences that can correct substitution, insertion, loss, and entire DNA chain errors. Compared with other DNA storage technologies, DNA-Aeon was reported to have better error-correcting capabilities at similar redundancy levels and reduced DNA synthesis costs.

The existing DNA storage encoding schemes tend to achieve high-density storage and neglect the local and global stability of DNA sequences and the read and write accuracy of stored information, which raises the possibility of data damage and errors during DNA synthesis and sequencing. To address these issues, this article proposes the DBTRG encoding scheme based on the graph: the image is first converted into binary sequences, then split into two halves (C1 and C2) and subjected to an XOR operation. The binary sequence obtained after XOR is encoded using the De Bruijn Trim Graph algorithm, and the dynamic binary sequence and C2 are encoded using the Rotating Tree algorithm. Characterization of encode generation rules and RS correction are addressing data corruption in DNA storage. In Fig. 1, the overall process of the encoding scheme is shown, including the De Bruijn Trim Graph algorithm and the Rotating Tree algorithm. The overall encoding-decoding process is shown in detail (Supplementary Figure S1). According to simulated experimental findings, the encoded DNA sequences ensure baseline diversity and balance while also preventing the emergence of undesired motifs. Therefore, the exhibited stability and robust error-correcting capability of this approach can facilitate the extensive adoption of graph-based encoding within the realm of DNA storage.

2. Methods

Numerous DNA storage encoding schemes have been proposed that have demonstrated practical high-density storage capabilities. However, these schemes have limitations, such as insufficient consideration of DNA sequence stability and read/write accuracy. Further research is needed to investigate the stability and information storage mechanisms of DNA sequences and to develop more robust DNA storage encoding schemes that can meet future needs for DNA data processing and storage. The DBTRG encoding scheme proposed in this article includes the De Bruijn Trim Graph algorithm and the Rotating Tree algorithm. In the De Bruijn Trim Graph algorithm, the binary sequence obtains the dynamic binary sequence according to the local constraint rule and conducts an XOR operation with the dynamic binary sequence. The XOR result is mapped into the corresponding base sequence to obtain the De Bruijn sequence, and the relationship between nodes is used to realize the compression function of the sequence and improve the base utilization. In the Rotating Tree algorithm, the information needed in the decoding process is stored by building an index tree and a matrix. During data recovery, both parts of the encoded information can be decoded simultaneously to obtain the data stored in the DNA sequence.

2.1. De Bruijn Trim graph algorithm

2.1.1. Dynamic binary sequence

To obtain the De Bruijn sequence, the article introduces a dynamic binary sequence, obtained by performing XOR operations between the binary sequence and the dynamic binary sequence. The dynamic binary sequence is generated dynamically by the binary sequence according to the local constraint rules, which can ensure that the DNA sequence satisfies the local and global GC content of 49%–51%, and the homopolymer length of two. Base balance and diversity are ensured to obtain high-quality DNA sequences. When encoding is performed, constraints must be observed to ensure the generation of high-quality and reliable DNA sequences [34].

When generating the dynamic binary sequence, the binary sequence is divided into several sub-sequences of six bits each. Each sub-sequence is XORed with the dynamic binary sequence to obtain a new binary sequence. This new binary sequence is then mapped to a base sequence,

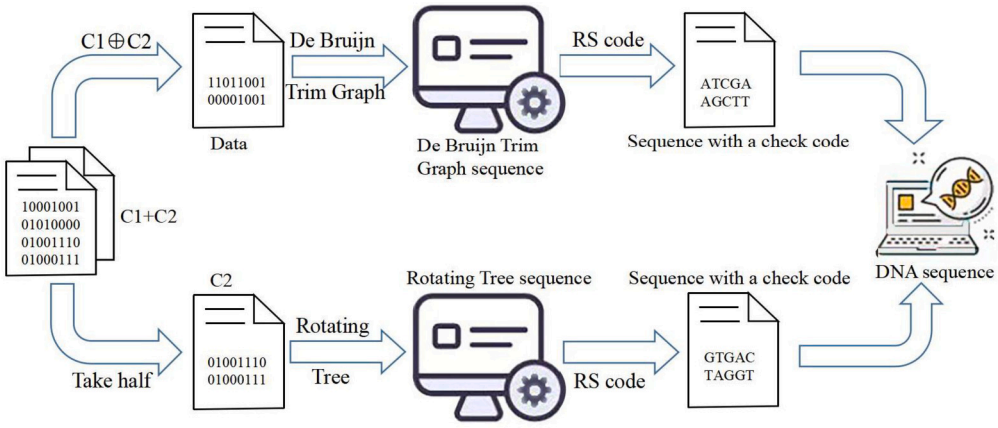

Fig. 1. Overall process of the De Bruijn Trim Rotation Graph encoding scheme.

with the last two bases of each group being identical to the first two bases of the adjacent group, resulting in a De Bruijn sequence. The required dynamic binary sequence that satisfies the local and global constraints for the generated De Bruijn sequence is shown in Algorithm 1. The process of generating the dynamic binary sequence is as follows:

Step 1: For the first group of bases, the first six binary sequences and the corresponding base sequences after the dynamic binary sequence XOR meet the constraints and non-undesired motifs, that is, TGA.

For example, 110100, the original sequence XORed with the dynamic binary sequence, obtains 000011 ⊕ 110111 = 110100. According to the rules of 00, 01, 10, and 11 converted to A, G, C, and T, respectively, every two bits of the binary correspond to one base, and the resulting base sequence is TGA.

Step 2: Starting with the second group, ensure that the first two bases of the group coincide with the last two bases of the previous group; that is, it is necessary to ensure that the second set of sequences' XOR becomes 0100xx. The first four binary sequences of the second group are XORed by 0011 and 0111 to obtain 0100. The last base of each group needs to be determined according to the previous GC content, undesired motifs, and the balance of bases at odd and even positions.

According to this rule, it can be determined that the last base in the second group is an A or a T. If it is A, the last two bits of the binary XOR are followed by 11. After XORing the last two bits of the second set of original sequences, which is 11, with the binary corresponding to A, which is 00, the resulting sequence is 00. Therefore, the second group of the dynamic sequence would be 010010. The XOR of this sequence would result in 011000, and the corresponding base sequence would be GAC.

Step 3: The same method can be used to obtain the third group of dynamic binary sequences: 111001. The XOR sequence is 001011, and the corresponding base sequence is ACT.

Algorithm 1: Algorithm for generating dynamic binary sequence.

```
Input: After the original binary sequence has been divided, XOR.
Output: Dynamic binary sequence.
Select a set of adjacent bases in the desirable motifs
Save the newly generated sequence
Adjacent k-mers overlap (k-1)-mer
for indexes and bases do
    if satisfy the GC content and the balance of bases at positions then
        select A or T that satisfies the designed motifs
        else
            select G or C that satisfies the designed motifs
        end
        Convert dynamic base sequence to binary sequence
    end
end
```

2.1.2. De Bruijn Trim graph

The De Bruijn graph (DBG) represents a DNA sequence as a directed graph. Given a string $S = s_1 s_2 \ldots s_n$ of length n, it is divided into a k-mer sequence of length k. The structural characteristics of the k-mer sequence can be represented using the symbol Σ and the graph $G(V, E)$ [35]. Here, V is the set of all substrings of length k, that is, the set of nodes. E represents the connection relationship between adjacent k-mers, that is, the set of edges. V and E are defined as follows:

$$V = \{v \in \sum^{k} | \exists i \in \{1, \ldots, n\} \; such \; that \; v \; is \; a \; substring \; of \; s_i \in S\} \quad (1)$$

$$E = \{(v, v') | if \; the \; suffix \; of \; length \; k-1 \; of \; v \; is \; a \; prefix \; of \; v'\} \quad (2)$$

In the DBG, each node represents a k-mer sequence fragment, and edges represent the repeat relationship between two k-mer sequence fragments of length k. Therefore, by constructing the DBG, the size of sequence data can be reduced to a manageable range while retaining a large amount of information [36]. The DNA sequence is decomposed into all possible k-mers (substrings of length k), and each k-mer is a node in the graph. Adjacent k-mers form an edge by overlapping $k-1$ bases. Fig. 2 can be explained as follows: Let X be a DNA sequence on the symbol Σ = {A, C, G, T}. Given a reference sequence X and an integer k, $DBG(X, k)$ represents the DBG with k-mer length of X. The original sequence of this graph is $TCGAACGATGGATCG$, and $k = 3$, that is, $DBG(X, 3)$. The De Bruijn Trim graph is the opposite. Since edges only exist when there is an overlap of base sequences between nodes, this relationship helps to compress the storage space of the base sequences. To store the same information using fewer bases during encoding, base groups that match the relationship between nodes are first obtained, and redundant base pairs are then removed during graph construction.

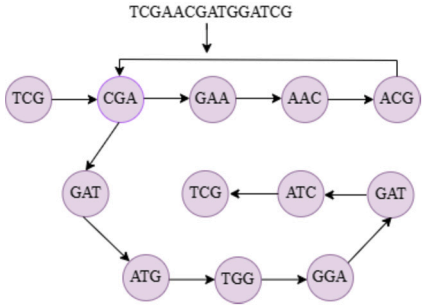

Fig. 2. All k-mers of the sequence have overlapping $(k-1)$-mers and are constructed into a De Bruijn graph.

As the DBG divides a sequence into all possible k-mers, any two adjacent k-mers share a $(k-1)$-mer. Therefore, the proposed De Bruijn Trim graph fully utilizes the overlapping relationship between nodes and satisfies the characteristics of a De Bruijn sequence after mapping the binary sequences to the base sequences. The De Bruijn Trim graph represents a directed graph with overlapping relationships between the sequences and is used to show the overlap between the base sequences. Although the De Bruijn Trim graph also represents a DNA sequence as a directed graph, the k-mer is partitioned in a different way. The De Bruijn Trim graph divides the De Bruijn sequence into groups per k-mer, with the last $k-1$ bases of the previous group being identical to the first $k-1$ bases of the next group. The edges of the adjacent base groups are constructed as a De Bruijn Trim graph, which can clearly represent the overlapping relationship in the base sequence and thus facilitate the removal of duplicated base pairs. Equations (3)–(6) can be used to describe this process.

Assuming $S = s_1 s_2 \ldots s_n$, S is divided into k-mers every three bases, resulting in a sliding window sequence with a step size of three. The element of the sliding window sequence at position k is denoted as w_k, where k ranges from 1 to $n/3$.

$$w_1 = s_1 s_2 s_3, w_2 = s_4 s_5 s_6, \ldots, w_k = s_{n-2} s_{n-1} s_n \tag{3}$$

The last two bases of the current k-mer must be identical to the first two bases of the next adjacent k-mer. Therefore, this constraint can be expressed as follows:

$$(s_{i+1}, s_{i+2}) = (s_{i+3}, s_{i+4}), i = 1, 4, 7, \ldots, n-5 \tag{4}$$

In summary, the constraint of k-mer partitioning can be expressed as follows:

$$w_i = (s_{3i-2} s_{3i-1} s_{3i}) \ \ for \ \ all \ \ i \in [1, n/3] \tag{5}$$

$$(s_{3i+1}, s_{3i+2} = s_{3i+4}, s_{3i+5}) \ \ for \ \ all \ \ i \in [1, n/3-2] \tag{6}$$

Each k-mer of the De Bruijn sequence forms a node, and the connected nodes have overlapping $(k-1)$-mers. Then, one $(k-1)$-mer of one of the nodes is removed, which improves base utilization. Even if there are cases where the constraints are not met during the process of converting the binary code into bases, by compressing the De Bruijn sequence, a base sequence that satisfies the constraints can be created. The De Bruijn Trim graph compresses the De Bruijn sequence to satisfy local and global constraints and reduce storage space, as shown in Fig. 3. For example, ATC and TCG can be represented as ATCG. There will be one redundant base pair every six bases, which is removed as redundancy. The dynamic binary sequence and the original binary sequence are mapped and compressed after XOR, and the compressed sequence complies with the local GC content and homopolymer length constraints. Given a set of base pairs, ATC, the next set must be TCG or TCC, which can be compressed into ATCC or ATCG in the De Bruijn Trim graph.

The De Bruijn Trim Graph algorithm is divided into three parts: processing of the dynamic binary sequence, construction of the De Bruijn Trim graph, and compression of the De Bruijn sequence. First, the binary sequence is grouped at intervals of six bits, and a new binary sequence is designed with a four-bit prefix and suffix matching. Subsequently, all the matching binary sequences are merged, and the dynamic binary sequence is obtained by XORing with the binary sequence. Next, the binary sequence and the dynamic binary sequence are XOR-mapped to obtain a new binary sequence. Then, based on the rule that one base is represented by two binary bits, the new binary sequence is mapped into the corresponding base sequence, which is the De Bruijn sequence. Finally, the De Bruijn sequence is k-merized for further analysis and processing. K-merization refers to dividing the sequence into strings containing k bases. A window of length k slides along the entire base sequence one base at a time, and each sliding produces a k-sized subsequence. Therefore, a base sequence of length m

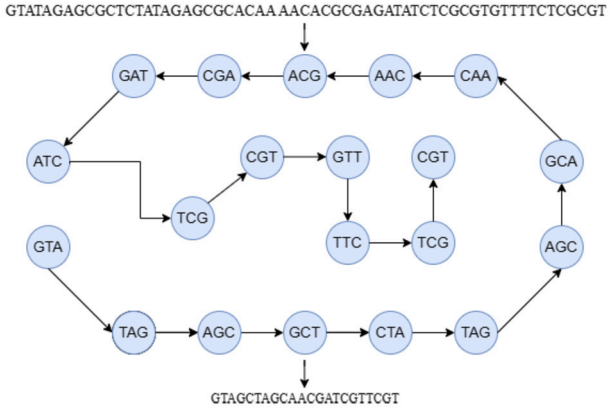

Fig. 3. All adjacent k-mers in the sequence have overlapping $(k-1)$-mers and are constructed into a De Bruijn Trim graph.

can be divided into $m-2$ k-mers. Here, the base sequence is divided into groups of three bases, namely 3-mers. Each 3-mer serves as a node in the graph. Since adjacent nodes have overlapping $(k-1)$-mers, connecting them forms the De Bruijn Trim graph. This graph can show the overlapping relationships and order of the sequences. Since every two base groups in the De Bruijn Trim graph share a common base pair, compression can be performed; that is, the first node takes the entire k-mer, and each subsequent node only takes the final base and combines it with the previous k-1 bases to obtain the compressed DNA sequence. The De Bruijn Trim Graph algorithm flow is shown in Fig. 4. C1 and C2 represent two halves of the binary sequence obtained by converting the images. In this article, C1 corresponds to the first half of the original binary, and C2 corresponds to the second half of the original binary. After applying the XOR operation between C1 and C2, the De Bruijn Trim Graph algorithm is used for encoding. On the other hand, C2 is encoded using the Rotating Tree algorithm. The combination of these two encoding methods can effectively reduce the number of undesired motifs and increase the encoding rate.

2.2. Rotating Tree Algorithm

To improve base utilization, information storage can be compressed during the encoding process [37]. The Huffman tree encoding with the rotating table does not meet sufficient constraints, which will lead to errors in the procedures of synthesis and sequencing in vitro DNA storage [38,39]. The Rotating Tree algorithm is proposed to obtain the matrix by deleting the same base pair at the same index position in the rotating table. The Rotating Tree algorithm combines the index tree with rotating encoding to achieve compression encoding and modifies the rotating table into a matrix. To fulfill the requirement of local GC content, it is necessary to adjust the position of base pairs in the matrix. Specifically, one can achieve this by cyclically shifting elements, ensuring that each row has an equal number of repeated base pairs. This approach helps maintain the balance of local GC content. Therefore, to avoid the occurrence of homopolymers in this process as much as possible, the continuous occurrence of repeated base pairs should be considered when setting the matrix. And to ensure that the homopolymer length is at most 2, which means that consecutive repeated base pairs (such as AA, GG, CC, and TT) are not allowed, it is essential to prevent these combinations from appearing in the matrix. This can be achieved by placing non-repetitive base pairs at the corresponding positions in the matrix. For example, if “AA”occurs, the next possible selection should not be “AA”as it would result in “AAAA”, which does not meet the requirement of a maximum homopolymer length of 2. If AA, GG, CC, and TT are not present in the matrix, then the homopolymer length of any two base pair combinations is at most 2.

The set x contains four bases: $\{A, C, G, T\}$. Let x_i denote a distinct combination of two bases (with the bases being different) from set x.

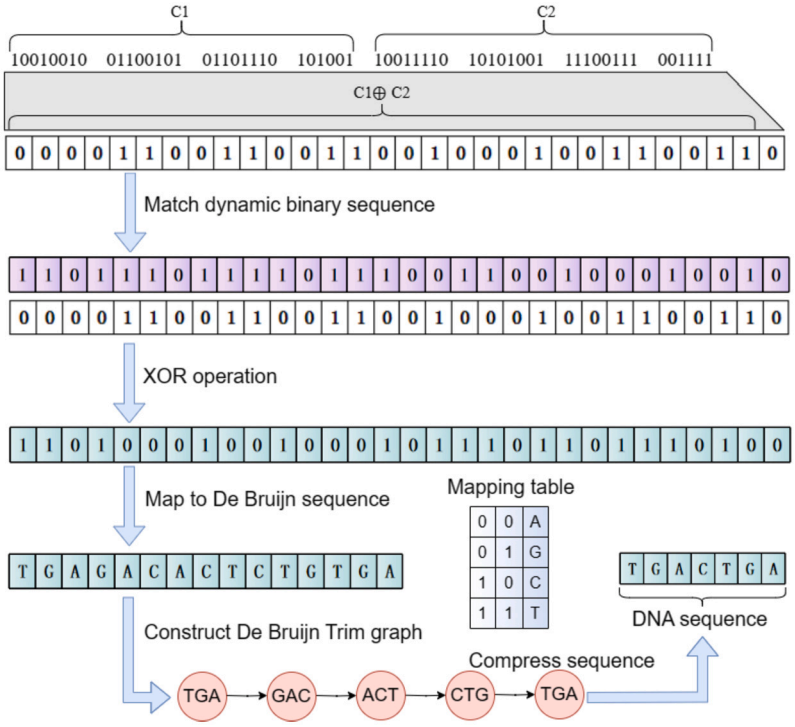

Fig. 4. Schematic illustration of an instance of a DNA storage encoding strategy using the De Bruijn Trim Graph algorithm.

Twelve distinct combinations can be formed, such as $x_1 = AC, x_2 = AG, \ldots, x_{12} = TG$. The i-th row of the matrix can be expressed as follows:

$$(M_{i1}, M_{i2}, ..., M_{i12}) = (x_i, x_{i+1}, ..., x_{12}, x_1, x_2, ..., x_{i-1}) \tag{7}$$

Here, the i-th row of the matrix is represented as $M_{i1}, M_{i2}, \ldots, M_{i12}$, where M_{ij} represents the element in the i-th row and j-th column. $x_1, x_2, \ldots, x_{12}$ represent the 12 base pairs in the first row, and each x_i represents the corresponding base pair. The i-th row represents the base pairs starting from x_i and is cyclically shifted left $i-1$ positions along the column direction. The matrix can be represented by a 12×12 *matrix*, given by

$$Matrix = \begin{bmatrix} x_1 & x_2 & \cdots & x_{12} \\ x_2 & x_3 & \cdots & x_1 \\ \vdots & \vdots & \ddots & \vdots \\ x_{12} & x_1 & \cdots & x_{11} \end{bmatrix} \tag{8}$$

Additionally, it should be emphasized that in the Rotating Tree algorithm, the index tree pertains to a specialized tree structure utilized for the purpose of an index (code), which is built using decimal numbers and their corresponding frequencies. And the index tree is designed such that the leaves of the tree are strictly ordered according to the index values ranging from 0 to 10. While the index tree is being constructed, the code is controlled at $[0, 10]$, which corresponds to the index in the matrix. Each base pair in the matrix corresponds to an index in the range $[0, 10]$. The Rotating Tree algorithm is shown in Fig. 5. First, the binary blocks are converted to decimals and sorted by frequency. Then, the index tree is constructed following the rules of the Huffman tree, with each layer containing the frequency, decimal number, and code. Next, the code corresponding to the original decimal number is found based on the index tree. Finally, the matrix is used to map the corresponding base pairs. During the encoding process, the first base pair after the first code is directly selected, and for the second code, the base pair that follows the code is determined based on the base pair corresponding to the first code. The DNA sequence obtained after matrix mapping has a homopolymer length of 2, which ensures base balance and diversity. The dynamic binary sequence and the second half of the original binary are encoded into DNA sequences by the Rotating Tree algorithm and stored for subsequent decoding work.

2.3. Data recovery

During data recovery, the DNA sequence can be divided into the De Bruijn Trim Graph sequence and the Rotating Tree sequence by removing the check bits after RS error correction. The process of transforming the DNA sequence into the original binary is a two-step process that involves first transforming the Rotating Tree sequence and then the De Bruijn Trim Graph sequence. The Rotating Tree decoding divides every two bases into a base pair, and each base pair can be found in the matrix. Each base pair corresponds to a code, and the code corresponding to the first set of base pairs defaults to the index corresponding to that base pair in the matrix. The first base pair is in the first row of the matrix, and then the second base pair is in the matrix. The index obtained by the intersection of the two base pairs is the second code. The third code is found by the second base pair and the third base pair, and the process is iterated until the codes corresponding to all base pairs are found. According to the code and index tree, we find the corresponding decimal of the code and convert it into binary to obtain the dynamic binary sequence and C2. The De Bruijn Trim Graph decoding first requires the extension of the De Bruijn Trim Graph sequence into a De Bruijn sequence following the procedure used to construct the DBG. The De Bruijn sequence is converted to binary and XORed with the dynamic binary sequence to obtain C1. The data recovery is completed by merging C1 and C2 (Fig. 6).

3. Results

To demonstrate the performance of DBTRG in a DNA storage system, constraints, encoding rate, error-correcting capability, and other aspects were analyzed in detail. “Random Access in Large-Scale DNA Data Storage”proved that undesired motifs in practical experiments did lead to a high error rate in DNA sequences [2,42]. The results of the simulated experiments demonstrate that DBTRG can meet local and global constraints while ensuring the encoding rate, local base balance, diversity,

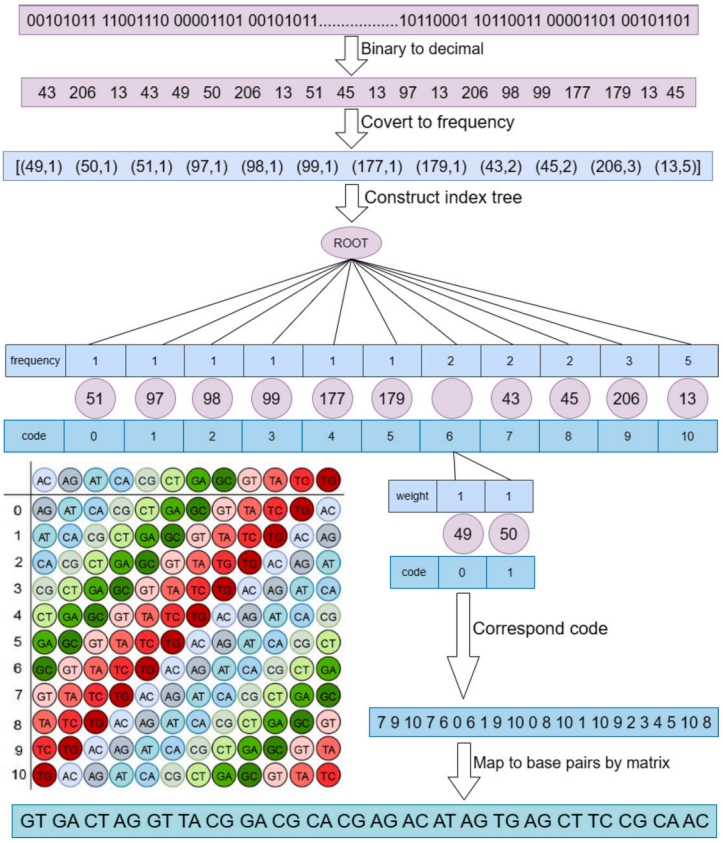

Fig. 5. Schematic illustration of an instance of a DNA storage encoding strategy using the Rotating Tree algorithm.

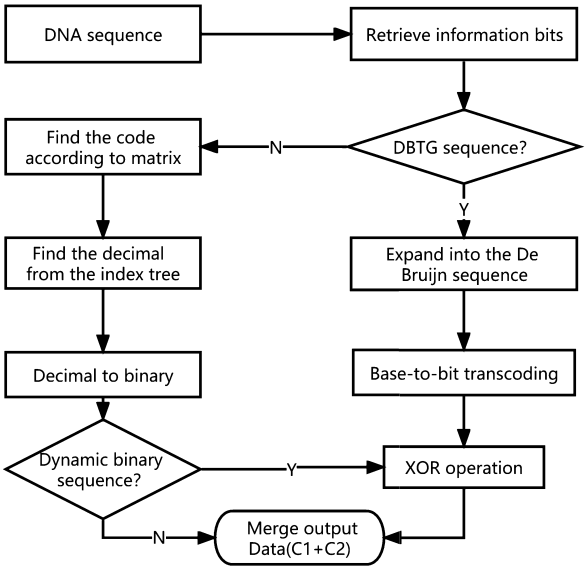

Fig. 6. Procedure for transforming binary digits from DNA sequences.

and lower probability of undesired motifs. The evaluation results show that this approach improves the quality and stability of DNA sequences.

3.1. Encoding performance

To validate the performance of the DBTRG encoding scheme in DNA storage systems, DBTRG was compared with previous representative works in terms of encode rate, local constraints, and error correction. The comparison results are shown in Table 1. The analysis of the table data shows that one of the advantages of the DBTRG encoding scheme is its ability to control the local GC content within the range of 49%-51%, which is better than other representative works. In addition, the DBTRG encoding scheme has strict control over the length of homopolymers, thus ensuring high base-balance diversity. Therefore, the DBTRG encoding scheme exhibits good performance in terms of local GC content and homopolymer length. In the process of encoding, an XOR mapping between the dynamic binary sequence and its original binary sequence is proposed, and the resulting De Bruijn sequence is compressed. After the compression is finished, DNA sequences that satisfy the local GC content constraints can be generated while ensuring the homopolymer length of two. The DNA sequence that satisfies the above constraint conditions improves the reliability and stability of DNA storage [40]. The encode rate after adding check bits for verification and error correction in the DNA sequence is 1.92.

3.2. Encode fragment performance analysis

To verify the local encoding performance of DBTRG, this section compares the local GC content of the sequences after DBTRG encoding and YYC encoding. The DNA sequences generated by the two encoding methods are taken as 600 bases each and divided into small fragments of seven bases each, and then the GC content of each small fragment is calculated. This processing method can better compare the differences in the local GC content of the two encoding methods and help evaluate their impact on DNA storage performance. As shown in Fig. 7, the red line represents the results obtained from the DNA sequence generated by the DBTRG encoding, and the black line represents the results obtained from the DNA sequence generated by the YYC encoding. The changes in the curve clearly show that the DNA sequence obtained by the graph-based DBTRG encoding scheme in this article has a better local GC content balance; this balance can ensure that the generated DNA sequences have good base balance and diversity, thereby improving the stability and reliability of DNA storage [43]. When the DNA sequence

Table 1
Comparison of various DNA-based data storage methods.

Scheme	Goldman [10]	Grass [41]	Anavy [42]	Erlich [20]	Ping [22]	This work
Input data (Mbytes)	0.75	0.08	6.4	2.15	1.40	0.50
Encoding potential (bits/nt)	1.58	1.78	1.93	1.98	1.95	1.92
Error-correcting algorithms	Repetition	RS	RS	Fountain	RA	RS
Robustness against excessive errors	Yes	Yes	Yes	No	Yes	Yes
GC content (%) of sequences	22.5–82.5	12.5–100	—	40–60	40–60	49–51
Maximum homopolymer length (nt)	1	3	—	4	4	2

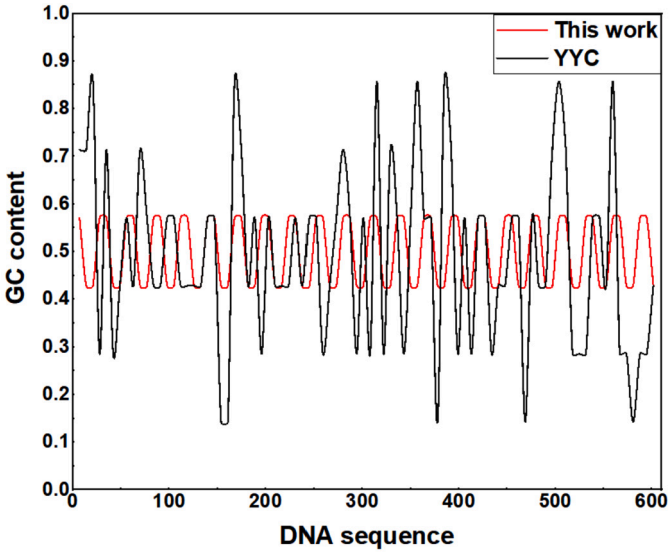

Fig. 7. Comparison of the two DNA storage systems' partial local GC base content.

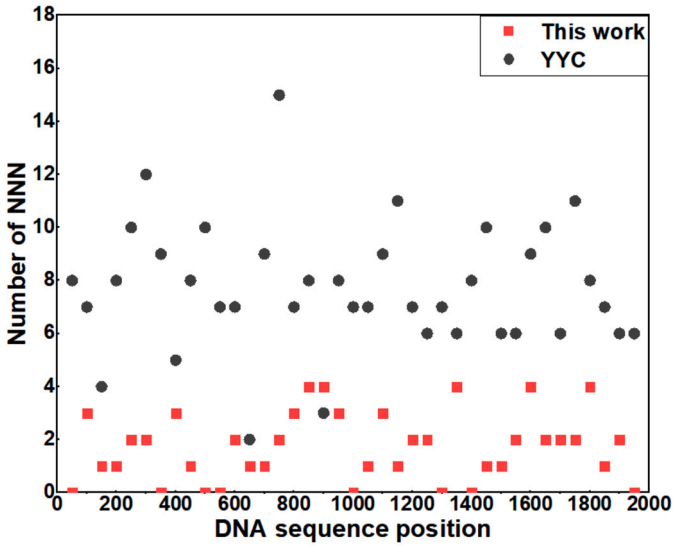

Fig. 8. Comparison of the occurrence of undesired motifs in the two DNA storage systems.

is long, the change in the local GC content has a greater influence on the reliability of DNA storage. Therefore, to improve the efficiency and reliability of DNA storage, special attention should be paid to balancing the local GC content. The De Bruijn sequence obtained by the dynamic binary sequence and the DNA sequence obtained after compressing the stored information ensure local base balance and diversity, thus improving the accuracy of reading and writing stored information and effectively reducing the error rate during the DNA synthesis and sequencing.

3.3. Undesired motifs

Undesired motifs are inappropriate combinations of adjacent bases in DNA sequences that can result in interactions between adjacent bases and increase the rate of synthesis or sequencing errors, thereby affecting the stability and reliability of DNA storage. Compared with other bases, G and T are more prone to random errors. Thus, even identical bases can have inconsistent error rates because their neighbors are different. Among all undesired motifs, the ones with the highest error rate are TGC, CGC, GTC, and GTG, followed by GAC, CAC, GCG, AGA, and ATA. The following is the error rate for each combination of undesired motifs: "GTC": 0.009, "TGC": 0.0085, "CGC": 0.008, "GTG": 0.008, "GAC": 0.0068, "GCG": 0.0067, "AGA": 0.0065, "ATA": 0.0067, "CAC": 0.006, "ACT": 0.0055, "TCT": 0.0052, "TAT": 0.0044 [44]. To verify the performance of DNA sequences encoded by DBTRG, the number of undesired motifs in 2000 base sequences was locally counted, and the same was done for the sequences encoded by YYC. The results of the comparison of the two encoding methods are shown in Fig. 8. It indicates that the number of undesired motifs in the DNA sequence encoded by YYC is higher than that in the DNA sequence encoded by DBTRG. Therefore, encoding by DBTRG reduces the occurrence of undesired motifs and improves the reliability and stability of the encoded DNA sequence.

3.4. Error correction performance

Errors are inevitable during the synthesis and sequencing of DNA. To reduce the error rate of DNA sequences during storage, error-correcting algorithms can be used to detect and correct errors. In this article, indels are converted into substitutions in the De Bruijn Trim Graph encoding DNA sequence according to the nature of the encoding itself. If the indels do not conform to the odd position of A/T and the even position of G/C, the base at the corresponding position is deleted or added, which is converted into substitution errors. Then RS error correction is added to improve the reliability and stability of DNA storage. Fountain encoding typically uses a random distribution method to write data blocks into DNA sequences, which requires sufficient redundant sequences to recover the data, whereas RS codes divide data into several equally sized blocks, encode each block, and then write them into DNA sequences. This division method makes RS codes more powerful in error correction for each data block because each data block is separately encoded with error correction. The simulation experiment is conducted under the following conditions: an indel rate of 2.04×10^{-3} and a substitution rate of 4.5×10^{-3} [44]. We conducted a comprehensive assessment of the combined DBTRG approach, and the experimental results are presented in the Fig. 9, 10, and 11. In handling indel errors, Fountain had a better data recovery rate, while DBTRG had a slightly better data recovery performance when the error rate was higher. However, when handling substitution errors, DBTRG's data recovery performance was significantly better than Fountain's. By comparing the error correction performance of DBTRG and Fountain encoding with the same redundant bases, it can be seen that before the crossing point, Fountain encoding has better data recovery ability, but after the crossing point, DBTRG

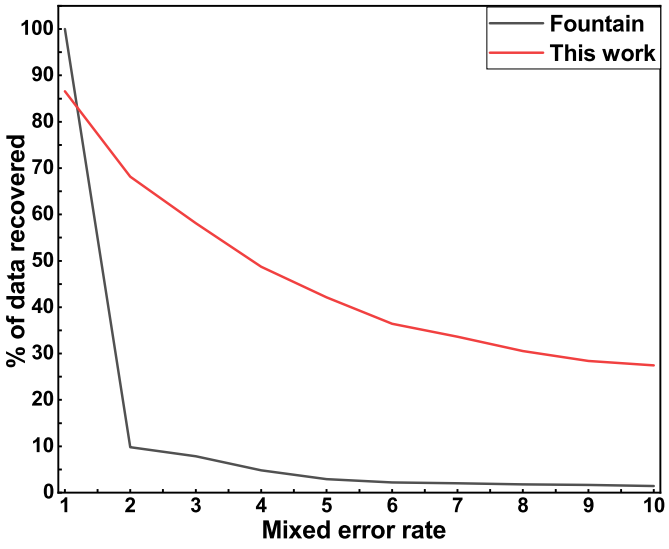

Fig. 9. Data recovery rates of two encoding schemes under mixed errors.

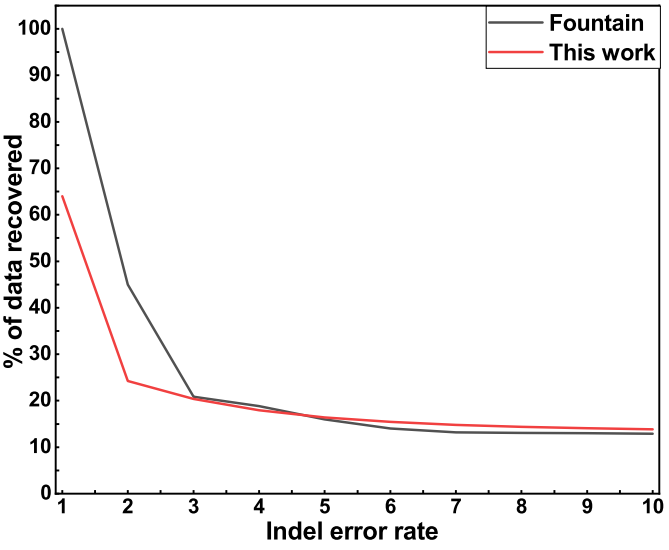

Fig. 10. Error correction analysis for the schemes regarding indel errors.

has significantly better recovery performance. It can be found that the DBTRG encoding scheme has better error correction performance.

4. Conclusion

Previous DNA storage encoding schemes have mainly focused on high storage density but have not fully considered the local and global stability of DNA sequences or the accuracy of information retrieval. Therefore, to ensure the quality and reliability of stored data, it is necessary to further study and optimize encoding schemes with full consideration of the stability and accuracy of DNA sequences. This article describes a graph-based DBTRG encoding scheme that adopts the De Bruijn Trim Graph algorithm and the Rotating Tree algorithm. To construct De Bruijn sequences that satisfy the overlapping relationship between k-mers, this article designs a dynamic binary sequence. By applying XOR and mapping the original binary sequence with the dynamic binary sequence, the De Bruijn sequence is partitioned into k-mers, and a De Bruijn Trim graph is constructed according to the overlapping relationship between bases, thereby achieving high storage density and low error rates. In the Rotating Tree algorithm, the index tree is used to compress the data to obtain the code. Then, the rotating table is improved based on previous work to obtain a matrix that can better meet the constraints, and the code is converted into base pairs by the matrix. Therefore, the constraints of local GC content and homopolymer length can be effectively met, and base balance and diversity can be ensured. To ensure read and write accuracy during DNA storage, DBTRG uses encoding properties and RS codes for verification and correction. The results of simulated experiments indicate that the DBTRG encoding scheme proposed in this article achieves local GC content of 49%-51% and homopolymer length of 2, which ensures base balance and diversity and reduces the probability of undesired motifs. In addition, DBTRG can maintain the encoding rate of 1.92, enhancing the stability and data recovery rate of DNA storage. This work provides valuable insights for the optimization of DNA storage and encoding methods for future research.

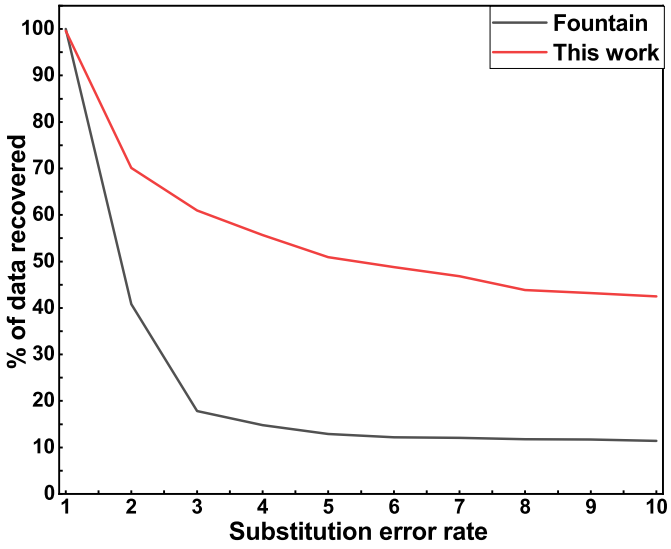

Fig. 11. Error correction analysis for the schemes regarding substitution errors.

Our future work will continue to focus on optimizing the graph-based DBTRG encoding scheme to increase storage density and obtain more stable DNA sequences. We will attempt to solve the encoding problem through graph construction, obtain higher-quality DNA sequences, and reduce redundancy. Furthermore, the application of deep learning techniques in related fields, such as using them for DNA sequence classification, prediction, or exploring patterns within DNA sequences, has the potential to inspire and guide future research on DNA storage and encoding [45,46]. This will ensure that the stored data can be safely kept in a smaller physical space and can be effectively retrieved and searched. This will help DNA storage systems possess greater reliability while achieving high-density storage.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

The data and code are available at the GitHub repository: https://github.com/TalentZ111/DNA-Storage_DBTRG

Appendix A. Supplementary material

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.csbj.2023.09.004.

References

[1] Kirola M, Memoria M, Shuaib M, et al. A referenced framework on new challenges and cutting-edge research trends for big-data processing using machine learning approaches. In: 2023 international conference on smart computing and application (ICSCA). IEEE; 2023. p. 1–5.

[2] Bornholt J, Lopez R, Carmean D, et al. A DNA-based archival storage system. In: Proceedings of the twenty-first international conference on architectural support for programming languages and operating systems; 2016. p. 637–49.

[3] Bencurova E, Akash A, Dobson RCJ, et al. DNA storage—from natural biology to synthetic biology. Comput Struct Biotechnol J 2023.

[4] Doricchi A, Platnich CM, Gimpel A, et al. Emerging approaches to DNA data storage: challenges and prospects. ACS Nano 2022;16(11):17552–71.

[5] Zhirnov V, Zadegan RM, Sandhu GS, et al. Nucleic acid memory. Nat Mater 2016;15(4):366–70.

[6] Ping Z, Ma D, Huang X, et al. Carbon-based archiving: current progress and future prospects of DNA-based data storage. GigaScience 2019;8(6). giz075.

[7] Dong Y, Sun F, Ping Z, et al. DNA storage: research landscape and future prospects. Nat Sci Rev 2020;7(6):1092–107.

[8] Cao B, Wang B, Zhang Q. GCNSA: DNA storage encoding with a graph convolutional network and self-attention. iScience 2023;26(3).

[9] Ceze L, Nivala J, Strauss K. Molecular digital data storage using DNA. Nat Rev Genet 2019;20(8):456–66.

[10] Goldman N, Bertone P, Chen S, et al. Towards practical, high-capacity, low-maintenance information storage in synthesized DNA. Nature 2013;494(7435):77–80.

[11] Banal JL, Shepherd TR, Berleant J, et al. Random access DNA memory using Boolean search in an archival file storage system. Nat Mater 2021;20(9):1272–80.

[12] Xu C, Zhao C, Ma B, et al. Uncertainties in synthetic DNA-based data storage. Nucleic Acids Res 2021;49(10):5451–69.

[13] Antkowiak PL, Lietard J, Darestani MZ, et al. Low cost DNA data storage using photolithographic synthesis and advanced information reconstruction and error correction. Nat Commun 2020;11(1):5345.

[14] Zhang Y, Ren Y, Liu Y, et al. Preservation and encryption in DNA digital data storage. ChemPlusChem 2022;87(9):e202200183.

[15] Ezekannagha C, Welzel M, Heider D, et al. DNAsmart: multiple attribute ranking tool for DNA data storage systems. Comput Struct Biotechnol J 2023;21:1448–60.

[16] Mu Z, Cao B, Wang P, et al. RBS: a rotational coding based on blocking strategy for DNA storage. IEEE Trans Nanobiosci 2023.

[17] Li X, Chen M, Wu H. Multiple errors correction for position-limited DNA sequences with GC balance and no homopolymer for DNA-based data storage. Brief Bioinform 2023;24(1). bbac484.

[18] Rasool A, Jiang Q, Wang Y, et al. Evolutionary approach to construct robust codes for DNA-based data storage. Front Genet 2023;14:415.

[19] Rasool A, Qu Q, Wang Y, et al. Bio-constrained codes with neural network for density-based DNA data storage. Mathematics 2022;10(5):845.

[20] Erlich Y, Zielinski D. DNA fountain enables a robust and efficient storage architecture. Science 2017;355(6328):950–4.

[21] Penghao W, Mu Z, Sun L, et al. Hidden addressing encoding for DNA storage. Front Bioeng Biotechnol 2022:1220.

[22] Ping Z, Chen S, Zhou G, et al. Towards practical and robust DNA-based data archiving using the Yin–Yang codec system. Nat Comput Sci 2022;2(4):234–42.

[23] Qu G, Yan Z, Wu H. Clover: tree structure-based efficient DNA clustering for DNA-based data storage. Brief Bioinform 2022;23(5):bbac336.

[24] Zheng Y, Cao B, Wu J, et al. High net information density DNA data storage by the MOPE encoding algorithm. IEEE/ACM Trans Comput Biol Bioinform 2023.

[25] Meiser LC, Nguyen BH, Chen JY, et al. Synthetic DNA applications in information technology. Nat Commun 2022;13(1):352.

[26] Meiser LC, Antkowiak PL, Koch J, et al. Reading and writing digital data in DNA. Nat Protoc 2020;15(1):86–101.

[27] Yin Q, Zheng Y, Wang B, et al. Design of constraint coding sets for archive DNA storage. IEEE/ACM Trans Comput Biol Bioinform 2021;19(6):3384–94.

[28] Luncasu V, Raschip M. A graph-based approach for the DNA word design problem. IEEE/ACM Trans Comput Biol Bioinform 2020;18(6):2747–52.

[29] Song L, Geng F, Gong ZY, et al. Robust data storage in DNA by de Bruijn graph-based de novo strand assembly. Nat Commun 2022;13(1):5361.

[30] Park SJ, Park H, Kwak HY, et al. BIC codes: bit insertion-based constrained codes with error correction for DNA storage. IEEE Trans Emerg Top Comput 2023.

[31] Cao B, Shi P, Zheng Y, et al. FMG: an observable DNA storage coding method based on frequency matrix game graphs. Comput Biol Med 2022;151:106269.

[32] Chen W, Han M, Zhou J, et al. An artificial chromosome for data storage. Nat Sci Rev 2021;8(5):nwab028.

[33] Welzel M, Schwarz PM, Löchel HF, et al. DNA-Aeon provides flexible arithmetic coding for constraint adherence and error correction in DNA storage. Nat Commun 2023;14(1):628.

[34] Cao B, Zhang X, Cui S, et al. Adaptive coding for DNA storage with high storage density and low coverage. NPJ Syst Biol Appl 2022;8(1):23.

[35] Limasset A, Cazaux B, Rivals E, et al. Read mapping on de Bruijn graphs. BMC Bioinform 2016;17(1):1–12.

[36] Yu C, Mao K, Zhao Y, et al. Stliter: a novel algorithm to iteratively build the compacted de Bruijn graph from many complete genomes. IEEE/ACM Trans Comput Biol Bioinform 2021;19(4):2471–83.

[37] Ren Y, Zhang Y, Liu Y, et al. DNA-based concatenated encoding system for high-reliability and high-density data storage. Small Methods 2022;6(4):2101335.

[38] Mishra P, Bhaya C, Pal AK, et al. Compressed DNA coding using minimum variance Huffman tree. IEEE Commun Lett 2020;24(8):1602–6.

[39] Liu Y, Ren Y, Li J, et al. In vivo processing of digital information molecularly with targeted specificity and robust reliability. Sci Adv 2022;8(31):eabo7415.

[40] Zhang JX, Yordanov B, Gaunt A, et al. A deep learning model for predicting next-generation sequencing depth from DNA sequence. Nat Commun 2021;12(1):4387.

[41] Grass RN, Heckel R, Puddu M, et al. Robust chemical preservation of digital information on DNA in silica with error-correcting codes. Angew Chem, Int Ed 2015;54(8):2552–5.

[42] Anavy L, Vaknin I, Atar O, et al. Data storage in DNA with fewer synthesis cycles using composite DNA letters. Nat Biotechnol 2019;37(10):1229–36.

[43] Löchel HF, Welzel M, Hattab G, et al. Fractal construction of constrained code words for DNA storage systems. Nucleic Acids Res 2022;50(5) e30-e30.

[44] Organick L, Ang SD, Chen YJ, et al. Random access in large-scale DNA data storage. Nat Biotechnol 2018;36(3):242–8.

[45] Lei J, Li J, Liu J, et al. GALFusion: multi-exposure image fusion via a global-local aggregation learning network. IEEE Trans Instrum Meas 2023.

[46] Li X, Han P, Chen W, et al. MARPPI: boosting prediction of protein–protein interactions with multi-scale architecture residual network. Brief Bioinform 2023;24(1):bbac524.