Iterative Coding Scheme Satisfying GC Balance and Run-Length Constraints for DNA Storage With Robustness to Error Propagation

Seong-Joon Park, Yongwoo Lee, and Jong-Seon No

***Abstract*—In this paper, we propose a novel iterative encoding algorithm for DNA storage to satisfy both the GC balance and run-length constraints using a greedy algorithm. DNA strands with run-length more than three and the GC balance ratio far from 50% are known to be prone to errors. The proposed encoding algorithm stores data with high flexibility of run-length at most m and GC balance between $0.5 \pm \alpha$ for arbitrary m and α. More importantly, we propose a novel mapping method to reduce the average bit error compared to the randomly generated mapping method. By using the proposed method, the average bit error caused by the one base error is 2.3455 bits, which is reduced by 20.5%, compared to the randomized mapping. Also, it is robust to error propagation since the input sequence is partitioned into small blocks during the mapping step. The proposed algorithm is implemented through iterative encoding, consisting of three main steps: randomization, M-ary mapping, and verification. It has an information density of 1.833 bits/nt in the case of $m = 3$ and $\alpha = 0.05$.**

***Index Terms*—Bioinformatics, constrained coding, DNA storage, error propagation, greedy algorithm, iterative algorithm.**

I. Introduction

RECENTLY, there is a massive amount of data being produced every day. In 2025, nearly 175 zettabytes of data are expected to be created [1]. To handle and store all this information, the need for a new archival storage system has arisen. There are three main important aspects of new archival systems: density, durability, and energy cost. However, current storage such as magnetic tape, hard disk drive (HDD), and solid-state drive (SSD) cannot store exponentially growing data. Therefore, new archival storages that satisfy these constraints and store a huge amount of data have been researched.

Among several candidates, deoxyribonucleic acid (DNA) emerges as a suitable medium for a new storage system [2], which is called DNA storage. The main idea is to map the data to four nucleotides of DNA, adenine, cytosine, guanine, and thymine, denoted by 'A', 'C', 'G', and 'T', respectively. Since the size of nucleotides is extremely small, DNA storage can theoretically store up to one exabyte of data per cubic millimeter. It is already proved by the experiment that in one gram of DNA, nearly 215 petabytes of data can be stored [3]. Also, data can be stored for more than centuries in DNA storage, and also has low energy costs for storing the data. These are the reasons why DNA storage is more suitable for future archival storage compared to other devices, such as flash memory, HDD, and magnetic tape [4]. Because of these reasons, DNA storage is currently an active area of research in storage systems.

Despite these many advantages, DNA storage has several shortcomings that should be overcome. DNA storage has a relatively high synthesis cost. This problem requires an efficient encoding algorithm to store a large amount of data in less number of nucleotides. Also, the error rate of DNA storage is greatly influenced by the biochemical structure of DNA. The following two biochemical constraints should be met because these can cause a high error rate in both the synthesizing and sequencing processes.

- *GC balance ratio*: GC balance ratio is defined by the ratio of the number of G and C nucleotides to the whole sequence. This ratio needs to be near 0.5 because high or low GC balance ratio causes high drop out rates and polymerase chain reaction (PCR) errors [3], [5], [6]. Balancing this ratio leads to a lower error rate during both synthesis and sequencing processes.
- *Maximum run-length*: Maximum run-length is the maximum number of consecutive identical nucleotides in the DNA strand. It is known that substitution and deletion error rates increase if the maximum run-length is longer than six [5]-[6]. This would also cause a high error rate during the DNA storage processes.

Therefore, these two biochemical constraints should be met to have better performance in DNA storage.

There are many studies to preserve these two constraints. Goldman *et al.* [7] compressed the raw data using Huffman coding and preserved the maximum run-length limit. However, they did not preserve the GC balance ratio. Xue *et al.* [8] did not preserve the maximum run-length limit, but made GC balance ratio exactly 0.5 with deletion/insertion/mutation error correction. Some other researches preserved both the GC balance ratio and the maximum run-length limit. Erlich and Zielinski [3] used Luby transform and screening method to preserve these two constraints and proposed DNA Fountain code which stores the data with high physical density. Lu *et al.* [9] also applied DNA Fountain encoding scheme to preserve both constraints and proposed a new log-likelihood ratio calculation for low-density parity-check codes. Yazdi *et*

Manuscript received July 20, 2021; revised December 26, 2021; approved for publication by Alberto Tarable, Division I Editor, February 18, 2022.

S.-J. Park, Y. Lee, and J.-S. No are with the Department of Electrical and Computer Engineering, INMC, Seoul National University, Seoul 08826, Korea, email: sjpark@ccl.snu.ac.kr, yongwool@ccl.snu.ac.kr, jsno@snu.ac.kr.
Y. Lee is the corresponding author.

Digital Object Identifier: 10.23919/JCN.2022.000008

1229-2370/22/$10.00 © 2022 KICS

al. [10] preserved the GC balance ratio by partitioning data into eight bases and using specialized constrained coding. Also, they limited the maximum run-length by using homopolymer check codes. Mishra *et al.* [11] used the minimum Huffman variance tree and its complementary tree to compress the data and limit run-length at most 1. Also, they limited the GC balance ratio to nearly 0.5. Immink and Cai [12], [16] proposed mathematical DNA coding that preserves both the GC balance ratio and maximum run-length limit. Wang *et al.* [17] proposed DNA coding with high information density of 1.92 bit/nt, which converts 23 bits into 11 nt DNA sequence. They preserved both the GC balance ratio between 40% to 60% and the maximum run-length at most three by using a finite state transition diagram. Also, they conducted a practical DNA storage experiment using their constrained coding method [19]. Deng *et al.* [20] proposed a hybrid coding architecture consists of modified variable-length run-length limited constrained codes and protograph low-density parity-check code. Nguyen *et al.* proposed a [18] proposed constrained coding with information density of 1.92 bit/nt in error-free channel. They used run-length limited code to preserve the maximum run-length at most three and Knuth's balancing technique to preserve the GC balance ratio between 40% to 60%. Also, they applied error-correcting code to handle a single substitution/insertion/deletion error.

In this paper, we propose a new iterative DNA coding algorithm that satisfies two constraints mentioned above with high information density. The proposed method guarantees high flexibility for various constraints. The DNA strand with the GC balance ratio between $0.5\pm\alpha$ and maximum run-length of m can be obtained. One can flexibly set the desired values α and m for the proposed iterative encoding algorithm. Not only these two constraints but also other desired constraints such as avoiding specific patterns (for example, primers) of DNA can be set. There are two main contributions in this paper: i) reduction in the number of average bit errors, ii) robustness to error propagation. By using the proposed mapping table, the number of average bit errors is reduced compared to the randomly generated mapping table. The average bit error caused by one base error is 2.3455 bits, which is reduced by 20.5%, compared to the randomized mapping method. Also, the proposed work uses a 6nt-long DNA sequence, which is very short and makes robustness to error propagation. Although the length of the encoded sequence is very short, it has a high information density. These two features show that the proposed method is robust to error propagation. The proposed method is implemented through iterative encoding, which consists of three steps: randomization, M-ary mapping, and verification. Since the GC balance ratio and maximum run-length limit are two main constraints for DNA storage, this paper mostly focuses on preserving these two constraints, especially for the typical DNA data storage of $m = 3$ and $\alpha = 0.05$ [3], [6].

This paper is organized as follows. Section II contains definitions and preliminaries to understand concepts in DNA storage. In Section III, we describe the proposed iterative coding algorithm consisting of three steps that compress the raw data with two constraints. In Section IV, we calculate the information density by using well-known raw data files and compare them with existing works. Section V concludes the paper.

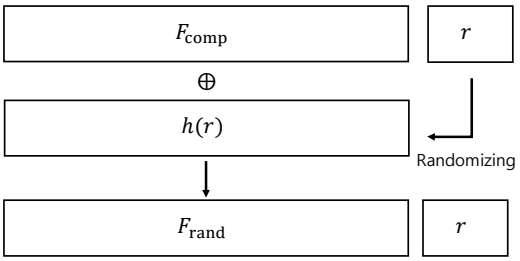

Fig. 1. Randomization algorithm.

II. Definitions and Preliminaries

In this paper, we define four types of nucleotides as quaternary numbers: $A = 0$, $C = 1$, $G = 2$, and $T = 3$.

Definition 1. *Let* $\mathbf{v} = (v_1, \cdots, v_n)$, $v_i \in \{0, 1, 2, 3\}$, *denote a DNA strand of length n with quaternary elements. Define the number of '1' and '2' in* $\mathbf{v}$ *as* $\eta(G, C)$. *Then GC balance ratio is defined as*

$$r_{GC} = \frac{\eta(G, C)}{n}.$$

In this paper, we say that DNA strand is balanced if $0.5-\alpha \leq r_{GC} \leq 0.5 + \alpha$ *for small* α.

Definition 2. *Let m be the maximum number of consecutively repeated identical nucleotides in the DNA strand, and we call it maximum run-length. In this paper, we say that the DNA strand satisfies the maximum run-length limit m.*

III. Proposed Iterative Encoding Algorithm

In this section, we propose a new encoding algorithm of DNA coding with two constraints in detail. It is implemented through iterative encoding, which consists of three steps: randomization, M-ary mapping, and verification. DNA sequences created by the proposed encoding algorithm satisfy given constraints, the GC balance ratio, and the maximum run-length. It is worth noting that, depending on the application, any efficient source coding can be applied before the proposed encoding algorithm. Since raw data files are used in our experiment, we apply the source coding step before the proposed encoding algorithm. Here, we perform binary source coding, considering each divided block of length k as a source symbol. Then the compressed output F_{comp}, as a binary form, is obtained.

A. Randomization

After obtaining the compressed output F_{comp}, we perform randomization to make the GC balance ratio between $0.5 \pm \alpha$. In the coding theory, there is a scheme called guided scrambling [15], which is similar to the randomization step. It is

TABLE I
COMPARISON OF SUBSTITUTION ERRORS ACROSS DIFFERENT BASES AND AVERAGE BIT ERROR.

Base to base	G to A	G to T	C to A	C to T	T to C	A to G	T to A	A to T	T to G	G to C	A to C	C to G
Sub. error prob.(%)	14.133	13.773	8.894	7.842	7.142	7.067	7.050	7.046	6.948	6.889	6.826	6.387
Average bit error	2	2.357	2.214	2.357	2.357	2	2.5	2.5	2.357	2.857	2.214	2.857

applied with the verification step to satisfy the GC balance ratio. Fig. 1 is the randomization process. In this process, the input value r is used to generate a randomized sequence using a random number generator function such as a hash function to obtain the output $h(r)$. After that, we perform bitwise XOR operation of F_{comp} and $h(r)$ to obtain F_{rand}, which is randomized binary data of F_{comp}. Generally, the length of F_{comp} is longer than $h(r)$, and thus F_{comp} is partitioned into the block of the same length as $h(r)$. Since $h(r)$ is randomized sequence, it also guarantees that F_{rand} is randomized. Also, the randomization step is performed iteratively, and it is explained more specifically in Section III-C.

B. M-ary Mapping

The next step is to map the binary sequence to the DNA sequence. In this step, the binary sequence is converted into M-ary symbols ($M = 3 \cdot 4^{m-1}$) and mapped to the DNA sequence by using the mapping table to satisfy the maximum run-length m. There are two important features in this step. The first feature is that we propose the mapping table to reduce the average bit error when one base error in the DNA sequence occurs. The mapping table is constructed using a greedy algorithm according to the substitution error probability between bases based on our experiment. The next feature is that the proposed method is robust to error propagation. While mapping the binary sequence to the DNA sequence, the binary sequence is partitioned into blocks of 11 bits. Then 11 bits are converted to a decimal number and a decimal number to two-digit 48-ary symbols in the case of $m = 3$. Since the binary sequence is partitioned into small blocks, the proposed method is robust to error propagation.

1) The Mapping Method Using a Greedy Algorithm: In the proposed method, the binary sequence is converted into M-ary symbols and mapped to the DNA sequence using a mapping table. To ensure that the maximum run-length m is satisfied after the mapping step, we should have $M = 4^{m-1} \cdot 3$. Therefore, in the typical DNA data storage of $m = 3$, 48-ary mapping table is required in the mapping step.

Lemma 1. *Let* $\mathbf{x}$ *be the vector of length* m *and* $\mathbf{X}_i$ *be the set of vectors defined as*

$$\mathbf{X}_i = \{\mathbf{x} = (x_1, \cdots, x_m) \in \{0,1,2,3\}^m \mid x_i \neq x_{i+1}\},$$

for an integer i.

For example, $\mathbf{X}_2 = \{(0,0,1), (0,0,2), \cdots, (3,3,1), (3,3,2)\}$, *where* $m = 3$ *and* $i = 2$ *and then,* $x_2 \neq x_3$ *for all* $\mathbf{x} = (x_1, x_2, x_3) \in \mathbf{X}_2$. *If any vectors in* $\mathbf{X}_i$ *are appended together, they have run-length less than or equal to* m.

Proof. Let $\mathbf{u}, \mathbf{v} \in \mathbf{X}_i$, that is, $u_i \neq u_{i+1}$ and $v_i \neq v_{i+1}$. Then there are two cases to be considered.

TABLE II
GRAY CODE OF 48-ARY SYMBOLS.

Binary	Symbol	Binary	Symbol	Binary	Symbol
000000	0	011011	27	101000	40
000001	1	011010	26	101010	42
000011	3	011110	30	101011	43
000010	2	011111	31	101001	41
000110	6	011101	29	101101	45
000111	7	011100	28	101111	47
000101	5	010100	20	101110	46
000100	4	010101	21	101100	44
001100	12	010111	23	100100	36
001101	13	010110	22	100101	37
001111	15	010010	18	100111	39
001110	14	010011	19	100110	38
001010	10	010001	17	100010	34
001011	11	010000	16	100011	35
001001	9	011000	24	100001	33
011001	25	001000	8	100000	32

(i) When $u_{i+1} \neq v_i$, the maximum run-length is $m-1$ when $\mathbf{u}$ and $\mathbf{v}$ are appended together as $(\mathbf{u} \mid \mathbf{v})$.
(ii) When $u_{i+1} = v_i$, the maximum run-length is m when $\mathbf{u}$ and $\mathbf{v}$ are appended together as $(\mathbf{u} \mid \mathbf{v})$.

For both cases, appending vectors in a different way such as $(\mathbf{v} \mid \mathbf{u})$, also has the same result. Therefore, vectors formed by appending any vectors in $\mathbf{X}_i$ have run-length less than or equal to m. □

Theorem 1. *Let* $\mathbf{V}$ *be the set of vectors consist of* $\mathbf{v} = (v_1, \cdots, v_m)$, $v_i \in \{0,1,2,3\}$. *In* $\mathbf{V}$, *there exist at least* $3 \cdot 4^{m-1}$ *vectors, whose combination of any of these elements have run-length at most* m.

Proof. It is easy to check that in Lemma 1, the vector set $\mathbf{X}_i$ has $3 \cdot 4^{m-1}$ different vectors. Vectors formed by appending any vectors in $\mathbf{X}_i$ have the run-length at most m. Therefore, this ensures that in vector set $\mathbf{V}$, there are at least $3 \cdot 4^{m-1}$ of vectors, in which combination of any of these vectors have maximum run-length m. □

According to Theorem 1, there are $3 \cdot 4^{m-1}$ different vectors in $\mathbf{V}$, and each vector can be mapped to $3 \cdot 4^{m-1}$-ary symbols. Let $M = 3 \cdot 4^{m-1}$. In the typical DNA storage of $m = 3$, we should have $M = 48$ and the binary sequence is converted into 48-ary symbol. After that, each 48-ary symbol is mapped to a 3nt-long DNA sequence using 48-ary mapping table. Rather than forming the randomized mapping table, we propose the mapping table to reduce the average bit error when one base

TABLE III
EXAMPLE OF 48-ARY MAPPING TABLE FOR $m = 3$ USING A GREEDY ALGORITHM.

DNA	Symbol	DNA	Symbol	DNA	Symbol
AAC	0	TCT	27	GTG	40
AAT	1	CCT	26	ATG	42
GAT	3	CCA	30	TTG	43
TAT	2	CCG	31	CTG	41
TGT	6	CAG	29	CTA	45
CGT	7	CAT	28	CGA	47
AGT	5	CAC	20	AGA	46
AGC	4	TAC	21	GGA	44
ATC	12	TGC	23	TGA	36
ATA	13	TTC	22	CGC	37
GTA	15	TTA	18	CTC	39
GCA	14	TCA	19	GTC	38
ACA	10	TCG	17	GAC	34
ACG	11	TAG	16	GGC	35
ACT	9	AAG	24	GGT	33
GCT	25	GAG	8	GCG	32

DNA sequence error occurs. Here, the mapping table is constructed using a greedy algorithm according to the substitution error probability between bases. We have an experimental result and obtain the substitution error probability between bases [13]. A more detailed explanation for the experiment is mentioned in Section IV-A. Table I shows the comparison of substitution errors across different bases according to our experiment. A substitution error from G to A has the highest probability, and a substitution error from C to G has the lowest probability.

We apply a greedy algorithm to form the mapping table according to Table I. Starting from (A, A, C), we find the sequence that is most likely to be changed when one base error occurs among 47 candidates, excluding (A, A, C). Table I is used to find the next DNA sequence. When the DNA sequence, which is most likely to be changed when one base error occurs, is chosen, we find the next DNA sequence from the chosen DNA sequence, eliminating the chosen DNA sequence from candidates. In this way, we find the next DNA sequence until all 48 DNA sequences are used. For example, (A, A, C) is most likely to change to (A, A, A) since the substitution error from C to A has the highest probability. However, (A, A, A) is not one of the candidates of 47 sequences since the last two bases are the same. Therefore, the next highest probability is the substitution from C to T, and it is one of the candidates of 47 sequences. Therefore, the next DNA sequence would be (A, A, T). If there is no remaining candidate that differs only one base, the DNA sequence with two nt differences with the highest probability is chosen for the next DNA sequence.

Now all 48 different DNA sequences should be mapped to all 48-ary symbols, respectively. Here, 48-ary symbols, from 0 to 47, can be expressed in 6 bits, from 000000 to 101111. We can reduce the average bit error by constructing the mapping table using a greedy algorithm. In this step, the mapping table can be constructed to have only a one-bit difference for adjacent symbols, such as Gray code. Table II is the Gray code of 48-ary symbols in the binary form, and adjacent symbols have only one-bit difference. By mapping DNA sequences to 48-ary symbols in the order of Table II, less number of bit errors would occur when one base error occurs in the DNA sequence. Table III is the example of 48-ary mapping table for $m = 3$. The greedy algorithm is applied in DNA sequences and DNA sequences are mapped with a Gray code of 48-ary symbols. This mapping method would decrease the average bit error by 20.5%, compared to the one in the randomly generated mapping table.The more detailed explanation is given in Section IV-A.

2) The Mapping Method With Robustness to Error Propagation: The proposed mapping method converts the 11 bit-long binary sequence to a 6nt-long DNA sequence. The binary sequence is converted to the DNA sequence as follows. The first step is to partition the binary sequence into blocks of length 11 bits. Then, 11 bits are converted to the decimal number D, which is in $0 \leq D \leq 2047$. Since $0\times48^1+0\times48^0 \leq D \leq 42\times48^1+31\times48^0$, D can be expressed as two-digit 48-ary symbols. Finally, each 48-ary symbol is converted to a 3nt-long DNA sequence by using the proposed mapping table in Table III. Therefore, every 11 bits are converted to a 6nt-long DNA sequence.

Since the binary sequence is partitioned into small blocks, it is robust to error propagation. Therefore, when an error occurs in the sequence, only the block with the error is corrupted. Also, the proposed mapping method has a complexity $O(n)$, where n is the length of the DNA sequence. In the proposed method, the binary sequence is partitioned and converted to the DNA sequence by using the mapping table. Since the size of the mapping table is related to the maximum homopolymer run, it has no relation to n. Therefore, it would be efficient to implement the proposed mapping method in the DNA storage channel because of its low complexity and robustness to error propagation. The comparison of the complexity with recent works is shown in Section IV-B, Table VI.

In the proposed method, the binary input data is randomized in the randomization step, which means the occurrence from 0 to 2047 for 11 bits is equally likely. However, when 11 bits are converted into two-digit 48-ary symbols from (0,0) to (42,31), each 48-ary symbol does not occur equally likely. However, when the mapping step is combined with the verification and iterative encoding, which is the next step of the proposed encoding algorithm, it is ensured that the GC balance ratio is satisfied. A more detailed explanation and the proof are given in Section III-C. In the proposed method, 48-ary symbols are mapped to DNA sequences according to Table III. Finally, the information density of this case is $11/6 = 1.8333$ bits/nt.

C. Verification and Iterative Encoding

After the mapping step, we partition the data into n-nt long DNA sequences and check whether the constraint is satisfied or not. Since the cost of synthesizing a long DNA sequence is very high, a DNA sequence should be partitioned into shorter sequences, about 150 nt to 300 nt, in the DNA storage [3], [18]. We call the last step of the proposed

coding algorithm, *verification*: to find DNA sequences until the desired constraints are satisfied. This method is not defined in the field of DNA storage, and thus we define the term *verification* as follows.

Definition 3. *For a given DNA sequence and set of constraints that the DNA sequence should meet, the verification is defined by the process of checking whether all constraints are satisfied.*

Algorithm 1 Iterative Encoding

Input: input data X, desired constraint U
Output: valid DNA sequence F_{dna}.
Initialization: $r \leftarrow 0$

```
F_comp ← SourceCoding(X)
while true do
    F_rand ← Randomization(F_comp, r)
    F_mapping ← M-aryMapping(F_rand)
    F_partitioned ← Partitioning(F_mapping)
    if Verification (U, (F_partitioned|r)) succeeds then
        return F_dna ← (F_partitioned|r)
    end if
    r ← r + 1
end while
```

The iterative encoding in Algorithm 1 is an algorithm to obtain the DNA sequence which satisfies the desired constraints. When the constraints are not satisfied, we go back to the randomization step and encode the data again until the constraints are satisfied.

As we mentioned in Section III-B2, the binary sequence is randomized after the randomization step. Thus the occurrences of each 48-ary symbol are equally likely. However, since 11 bits are converted into the 6nt-long DNA sequence in the mapping step, the iterative encoding needs to be applied to satisfy the GC balance ratio. In the following theorem, we derive the GC balance ratio $0.5 \pm \alpha$ in terms of the DNA sequence length n, for the case of $m = 3$, within four times of iterations. In the case of $m = 3$, let $\mathbf{v} = (v_1, \cdots, v_n)$, $v_i \in \{0, 1, 2, 3\}$, denote a DNA sequence of length n with quaternary elements. Let p_{GC} be the probability of vector $\mathbf{v}$ to satisfy the GC balance ratio between $0.5 \pm \alpha$ within the required number of iterations, I, defined by

$$p_{GC} = 1 - \{1 - P(0.5 - \alpha \le r_{GC} \le 0.5 + \alpha)\}^I \ge 1 - \epsilon, \quad (1)$$

where r_{GC} denotes the GC balance ratio of $\mathbf{v}$. Let $\mathbf{X}_q$ be the random variable representing the number of occurrences of G or C in the qth six tuple $(v_{6q+1}, v_{6q+2}, v_{6q+3}, v_{6q+4}, v_{6q+5}, v_{6q+6})$, for $0 \le q \le \frac{n}{6} - 1$. We assume that $\mathbf{X}_q$'s are statistically independent. Let p_l be the probability of $\mathbf{X}_q = l$, $0 \le l \le 6$. Then, we have

$$(p_0 + p_1 x + p_2 x^2 + \cdots + p_6 x^6)^{\frac{n}{6}} = \sum_{j=0}^{n} a_j x^j, \quad (2)$$

where a_j denotes the probability of the number of occurrences of G or C being j and let

$$p(\alpha, n) = \sum_{(0.5-\alpha)n \le j \le (0.5+\alpha)n} a_j, \quad (3)$$

TABLE IV
THE MINIMUM VALUE OF α FOR $I = 4, 8$ AND VARIOUS n WITH $\epsilon = 10^{-4}$.

I	n (nt)				
	100	150	200	250	300
4	0.07	0.06	0.05	0.048	0.044
8	0.04	0.04	0.03	0.028	0.027

which means $P(0.5 - \alpha \le r_{GC} \le 0.5 + \alpha)$ in (1).

Theorem 2. *Let the GC balance ratio between* $0.5 \pm \alpha$ *is satisfied in the proposed iterative encoding algorithm, for* $m = 3$. *Then, for* $n = 200$ *and* $I = 4$, $\alpha \ge 0.05$ *is achieved.*

Proof. Let Y be a random variable representing the number of the base G and C in a vector $\mathbf{v}$. Since every 11 bits are converted into six-nt DNA sequence, $\mathbf{X}_q$, for $0 \le q \le \frac{n}{6} - 1$, are independent and each $\mathbf{X}_q$ has the same $p_0, \ldots, p_6$ in (2). Therefore, the probability of $Y = j$ in $\mathbf{v}$ can be expressed as a_j as in (2). In other words, $a_j = P[Y = j]$. To satisfy balancing constraint for $I = 4$ iterations, (1) can be rewritten as

$$p_{GC} = 1 - \{1 - p(\alpha, n)\}^4 \ge 1 - \epsilon.$$

Since, $p(\alpha, n)$ is a probability, $p(\alpha, n)$ can be written as

$$1 \ge p(\alpha, n) \ge 1 - \epsilon^{\frac{1}{4}}.$$

Therefore, for $\epsilon = 10^{-4}$, we say the probability p_{GC} approximates to 1 and for $n = 200$ and $I = 4$,

$$\sum_{200(0.5-\alpha) \le j \le 200(0.5+\alpha)} a_j \ge 1 - \epsilon^{\frac{1}{4}}. \quad (4)$$

In the typical DNA storage with $m = 3$, when 11 bits are converted into 6nt-long DNA sequence using 48-ary mapping in Table III, we can express (2) as

$$\left(\frac{16}{2048} + \frac{148}{2048}x + \frac{487}{2048}x^2 + \frac{724}{2048}x^3 + \frac{505}{2048}x^4 + \frac{152}{2048}x^5 + \frac{16}{2048}x^6\right)^{\frac{n}{6}} = \sum_{j=0}^{n} a_j x^j,$$

where a_j and $p(\alpha, n)$ can be obtained. Therefore, for $n = 200$ and $I = 4$, $\alpha \ge 0.05$ can be achieved according to 4. □

Table IV shows the minimum value of α for $I = 4, 8$ and various n, using Theorem 2 for $\epsilon = 10^{-4}$. For the longer length and larger iteration number, the tighter GC balance ratio can be satisfied. Therefore, for the longer DNA sequences, the proposed iterative encoding algorithm can satisfy the tighter GC balance ratio for the fixed iteration number of $I = 4$. In addition, since the maximum DNA synthesis length becomes longer recently, the tighter GC balance ratio can be achieved in the proposed iterative encoding algorithm.

We chose $I = 4$ in Theorem 2 due to the following reasons: it is good to use a multiple-of-four I because I should be converted into quaternary DNA symbols, and in practice, $I = 4$ is enough to satisfy the GC balance ratio in the proposed iterative encoding algorithm. Since randomization input value r must be stored to obtain randomized output $h(r)$ in the decoding procedure, the iteration number at most four can be

expressed as one quaternary symbol. Therefore, a multiple-of-four I is adequate for the iteration number, but only $I = 4$ is enough to satisfy the GC balance ratio in the proposed iterative encoding algorithm. Starting from $r = 0$, as iteration number increases, r increases together until the GC balance ratio is satisfied. In other words, $r = I+1$. The input value r from 0 to 3 can be expressed as one quaternary symbol: $0 \rightarrow A$, $1 \rightarrow C$, $2 \rightarrow G$, $3 \rightarrow T$. Therefore, only one redundancy is appended on the front of the partitioned DNA sequence of F_{mapping}, which is $F_{\text{partitioned}}$, for satisfying the desired constraints. If the constraint is satisfied after appending the input value r, we obtain the final DNA sequence F_{dna}. According to Theorem 2 and Table IV, the GC balance ratio between 0.5 ± 0.05 can be satisfied for $n = 200$ within four times of iterations. Also, since only one-nt long DNA sequence is required for r, this occupies a very small part compared to the data.

Another important feature of the proposed encoding algorithm is flexibility. The desired constraints could be not only GC balance ratio but also other specific constraints. For example, avoiding particular patterns in the DNA strand could be possible. In future DNA storage, one pool might contain multiple files. Also, each file needs a primer, which is a pattern of DNA that is used for the polymerase chain reaction (PCR) or sequencing process, and different files have a different pattern of primer [14]. To not be confused with primer and payload, each file should avoid the primers of other files' patterns in its payload. We can set undesired patterns and encode them iteratively using verification until that pattern does not appear in the payload.

In addition to flexibility, as mentioned in Section III-B2, the proposed algorithm is robust to error propagation. In the proposed mapping method, by using the error probability obtained from the DNA storage experiment [13], we reduced the average bit error when one base error occurs during the mapping step. As a result, the average bit error for the proposed encoding algorithm is reduced by 20.5% compared to the randomly generated mapping table. Also, as mentioned in Section III-B2, the binary sequence is converted into the DNA sequence by partitioning it into blocks. These two features make the proposed encoding algorithm robust to error propagation. It is very important because, like other channels, the DNA storage channel also has several types of errors, such as substitution, insertion, and deletion errors. Therefore, in practical DNA storage, both constrained coding and channel coding should be applied. There are many works that apply error-correcting code in the DNA storage [3], [8], [10], [13], [14], [18]. However, there is a possibility that errors may occur even after the channel error correction. In this case, since the proposed method is robust to errors, only 11 bits in which the error occurred are replaced, and the rest of the data can be maintained. This is a strong point of the proposed method.

D. Decoding Algorithm

The decoding algorithm is performed through the inverse operation of the encoding algorithm. The decoding of the DNA strand is done by the following step:

1) Separate the value r from F_{dna}.
2) Map F_{mapping} to M-ary symbols and obtain $F_{M\text{-}ary}$.
3) Obtain F_{rand} by converting $F_{M\text{-}ary}$ into binary sequence.
4) XOR $h(r)$ with F_{rand} to obtain F_{comp}.
5) Obtain the raw input data F by source decoding if source coding is applied.

By using this algorithm, F_{dna} can be uniquely decoded, and the raw data input file F would be obtained.

IV. Results and Analysis

In this section, we show two performance results of the proposed encoding algorithm through the simulation. First, we show a reduction of the average bit error rate when one base error occurs in the DNA sequence, compared to the randomly generated mapping table. Second, results for information density and iteration numbers are obtained when the proposed encoding algorithm is applied in raw data files. Here, we use the proposed encoding algorithm for one text file and two image files.

A. The Comparison of the Average Bit Error Rate

As we mentioned in Section III-B1, before creating a mapping table, we obtain the substitution error probability between bases by conducting experiments [13]. In this experiment, 18000 oligo sequences of length 152 nt with 300 nanogram DNA oligo pools are synthesized by Twist Bioscience. We use the Illumina Miseq Reagent v3 kit (600 cycles) for sequencing and obtain 151 nt run for both forward and reverse reads. In this experiment, the substitution error probability across different bases are obtained as in Table I.

Using these error probabilities, we create the mapping table, Table III, and find the average bit error for each substitution error probability between bases. Table I shows the average bit error for each case of substitution errors when one base error occurs in the DNA sequence. Therefore, the average number of bit errors when one base error occurs is 2.3455 bits in the proposed encoding algorithm. Every 48-ary symbol could be changed to any other 48-ary symbol except itself when one base error occurs in the randomly generated mapping table. Therefore, the average number of bit errors can be obtained by considering all cases of converting one symbol into the other, from 0 to 1, 0 to 2, $\cdots$, 47 to 48, and we calculate the average bit errors caused by all cases. Then, the average number of bit errors is 2.9504 bits. In other words, the proposed mapping method could reduce the bit error by 0.6049 bits, which corresponds to 20.5% reduction, compared to the randomly generating mapping method. Therefore, in this step, the proposed mapping method reduces the bit error 20.5% in the DNA sequence while satisfying the maximum run-length at most three. In constrained coding, even one base error could cause a high bit error rate because of its high error propagation. However, the proposed algorithm prevents a single DNA error from causing a large number of bit errors.

TABLE V
COMPARISON OF INFORMATION DENSITY (BITS/NT) WITH SOURCE CODING.

Method	Mishra ($k = 16$) [11]			Proposed method ($k = 16$)		
	Text file of 3920 bits	Image files in pixels		Text file of 3920 bits	Image files in pixels	
		256×256	512×512		256×256	512×512
Information density	2.41	2.09	2.31	4.39	2.63	2.37

TABLE VI
COMPARISON OF INFORMATION DENSITY (BITS/NT) AND COMPLEXITY WITHOUT SOURCE CODING.

Method	Proposed method	[16]	[17]
Information density	1.833	1.834 ($n = 6$)	1.625 ($n = 8$)
		1.900 ($n = 10$)	1.917 ($n = 12$)
Complexity	$O(n)$	$O(4^n)$	$O(n)$

TABLE VII
THE NUMBER OF DNA SEQUENCES WITH $n = 200$ THAT SATISFY THE GC BALANCE RATIO BETWEEN $[0.45, 0.55]$ AT EACH ITERATION.

I	Text file of 3920 bits	Image files in pixels	
		256×256	512×512
1	4	899	3921
2	0	85	429
3	0	6	42
4	0	0	4
success rate	100%	100%	100%

B. Simulation Results for Information Density and Iteration Numbers

In our simulation, to compare with the recent work [11], we used the same text file, and two image files as in [11]. The text file is a poem *Where the Mind is Without Fear* by *Rabindranath Tagore*, which consists of 3,920 bits with 490 characters. The second file is a grayscale image of an airplane with a size of 256×256, where each pixel consists of 8 bits. The last file is a grayscale image of *'Lena'* with a size of 512×512, where each pixel consists of 8 bits. Each image file consists of 2,097,512 bits, 524,288 bits, respectively.

The experiment is held in the case of typical DNA storage, whose output has $m = 3$ and $0.45 \leq r_{GC} \leq 0.55$. In the case of source coding, any efficient scheme can be applied flexibly, but to compare with the work in [11], the minimum variance Huffman tree code has been used. Here, we define a block of length k as one symbol and we used the value $k = 16$ for the comparison with [11]. Obtaining compressed data F_{comp}, we use the SHA-3 algorithm for randomization to balance the data. We initialize $r = 0$ and the output length of $h(r)$ is 512 bits. Then we perform XOR of F_{comp} and $h(r)$. Next, we use a 48-ary converting table to obtain the DNA sequence F_{mapping} since we allow up to three runs. Finally, we first partitioned F_{mapping} into 200 nt long DNA sequences. Then in the verification step, we check whether the final output is balanced or not, and for the balanced DNA sequences, we append the input value r in the form of DNA. If the DNA sequence that does not satisfy the constraint remains, we start again from the randomization step increasing the value r by 1. Repeat this process until all DNA sequences are balanced. As mentioned in Section III-C, the iteration number of $I = 4$ is enough for satisfying the GC balance ratio for all text and image files in our simulation. A more detailed explanation is in the last paragraph of this section.

For the text file, 3920 bits are compressed to 1618 bits using Huffman code ($k = 16$), whose compression rate is 2.4227. Then we randomize the binary text file and convert the binary sequence into 48-ary symbols. In this step, 1618 bits are converted into 296 48-ary symbols. Then, each 48-ary symbol is converted into 3nt-long DNA sequence using Table III, which has the information density of 1.833 bits/nt. Finally, we append r by converting it into the DNA sequence using Table III, and then 3920 bits are converted into 888 nt DNA sequence. Table V shows that the proposed encoding algorithm has improved the information density compared to the recent work [11], for all files. As shown in Table V, the final information density using the proposed mapping method is 4.39 bits/nt, which means one DNA nucleotide contains 4.39 bits. For image files of sizes 256×256 and 512×512, information density is 2.63 bits/nt and 2.37 bits/nt, for $k = 16$, respectively.

Also, Table VI shows the comparison of information density and complexity with recent works [16], [17]. An information density of recent works [2] and [3] has a high information density of 1.900 bits/nt and 1.917 bits/nt. However, the length of the encoded sequence is 10nt and 12nt, which have a higher error propagation compared to the proposed method. In other words, when a channel error occurs, an error propagates to the whole 10nt and 12nt-long DNA sequence. Also, for the similar encoded length of the DNA sequence, the proposed method has an information density of 1.833 bits/nt, which is similar to [16] and higher than [17]. In terms of the complexity, the proposed method and the method in [17] have the same complexity of $O(n)$. However, the complexity of the method in [16] is $O(4^n)$, which is higher than the proposed work. Therefore, according to Table VI, the proposed work is robust to error propagation with high information density and low complexity.

For another experiment, we find the number of DNA

sequences of length 200 nt that satisfy the GC balance ratio between $[0.45, 0.55]$ at each iteration. According to Theorem 2 and Table IV, the GC balance ratio between $[0.45, 0.55]$ can be satisfied within four times of iterations. Table VII shows the experimental results for Theorem 2 and Table IV. As shown in Table VII, All DNA sequences satisfy the desired constraint within four times of iterations, and therefore only one redundancy is appended for each DNA sequence.

V. Conclusion

In the proposed DNA encoding algorithm, we use the iterative encoding algorithm for constrained coding to satisfy the desired GC balance ratio of $0.5 \pm \alpha$ and the maximum run-length of m. There are two main contributions to the proposed method. The first contribution is in the reduction of the number of average bit errors during constrained coding. By using the proposed mapping table, there is 20.5% of the average bit error reduction compared to the randomly generated mapping method. The second contribution is the robustness to error propagation. In the proposed method, each short binary sequence is converted into the DNA sequence with a high information density and low complexity of $O(n)$. These two features show that the proposed encoding algorithm is efficient and robust to error propagation As a result, the proposed encoding algorithm could reduce the synthesizing cost and error that could be occurred in the DNA storage processes.

References

[1] D. Reisel, J. Gantz, and J. Rydning, "Data age 2025: The digitization of the world: From edge to core," *Int. Data Corporation*, pp. 1–28, 2018.

[2] G. M. Church, Y. Gao, and S. Kosuri, "Next-generation digital information storage in DNA," *Science*, vol. 337, no. 6102, pp. 1628–1628, Sep. 2012.

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

[4] J. Bornholt, R. Lopez, D. M. Carmean, L. Ceze, G. Seelig, and K. Strauss, "Toward a DNA-based archival storage system," *IEEE Micro*, vol. 37, no. 3, pp. 98–104, Jun. 2017.

[5] J. J. Schwartz, C. Lee, and J. Shendure, "Accurate gene synthesis with tag-directed retrieval of sequence-verified DNA molecules," *Nat. Methods*, vol. 9, no. 9, pp. 913–915, Aug. 2012.

[6] M. G. Ross, C. Russ, M. Costello, A. Hollinger, N. J. Lennon, R. Hegarty, C. Nusbaum, and D. B. Jaffe, "Characterizing and measuring bias in sequence data," *Genome Biol.*, vol. 14, no. 5, p. R51, May 2013.

[7] N. Goldman, P. Bertone, S. Chen, C. Dessimoz, E. M. LeProust, B. Sipos, and E. Birney, "Towards practical, high-capacity, low-maintenance information storage in synthesized DNA," *Nature*, vol. 494, no. 7435, pp. 77–80, Jan. 2013.

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

[9] Xiaozhou Lu, J. Jeong, J. W. Kim, J. S. No, H. Park, A. No, and S. Kim, "Error rate-based log-likelihood ratio processing for low-density parity-check codes in DNA storage," *IEEE Access*, vol. 8, pp. 162892–162902, Sep. 2020.

[10] S. H. T. Yazdi, R. Gabrys, and O. Milenkovic, "Portable and error-free DNA-based data storage," *Sci. Rep.*, vol. 7, Art. no. 5011, 2017.

[11] P. Mishra, C. Bhaya, A. K. Pal, and A. K. Singh, "Compressed DNA coding using minimum variance Huffman tree," *IEEE Commun. Lett.*, vol. 24, no. 8, pp. 1602–1606, Aug. 2020.

[12] K. A. S. Immink and K. Cai, "Efficient balanced and maximum homopolymer-run restricted block codes for DNA-based data storage," *IEEE Commun. Lett.*, vol. 23, no. 10, pp. 1676–1679, Oct. 2019.

[13] J. Jeong, S.-J. Park, J.-W. Kim, J.-S. No, H. Jeon, J. W. Lee, A. No, S. Kim, and H. Park, "Cooperative sequence clustering and decoding for DNA storage system with fountain codes," *Bioinformatics*, vol. 37, no. 19, pp. 3136–3143, Oct. 2021.

[14] S. Yazdi, Y. Yuan, J. Ma, H. Zhao, and O. Milenkovic, "A rewritable, random-access DNA-based storage system," *Sci. Rep.*, vol. 5, no. 14138, 2015.

[15] K. A. S. Immink and L. Patrovics, "Performance assessment of dc-free multimode codes," *IEEE Trans. Commun.*, vol. 45, pp. 293-299, Mar. 1997.

[16] K. A. S. Immink and K. Cai, "Properties and constructions of constrained codes for DNA-based data storage," *IEEE Access*, vol. 8, pp. 49523–49531, Mar. 2020.

[17] Y. Wang, M. Noor-A-Rahim, E. Gunawan, Y. L. Guan, and C. L. Poh, "Construction of bio-constrained code for DNA data storage," *IEEE Commun. Lett.*, vol. 23, no. 6, pp. 963–966, Jun. 2019.

[18] T. T. Nguyen, K. Cai, K. A. S. Immink, and H. M. Kiah, "Capacity-Approaching Constrained Codes with Error Correction for DNA-Based Data Storage," *IEEE Inform. Theory*, vol. 67, no. 8, pp. 5602–5613, Aug. 2021.

[19] Y. Wang, M. Noor-A-Rahim, J. Zhang, E. Gunawan, Y. L. Guan, and C. L. Poh, "High capacity DNA data storage with variable-length oligonucleotides using repeat accumulate code and hybrid mapping," *J. Biol. Eng.*, vol. 13, no. 1, p. 89, Dec. 2019.

[20] L. Deng, Y. Wang, M. Noor-A-Rahim, Y. L. Guan, Z. Shi, E. Gunawan, C. L. Poh, "Optimized code design for constrained DNA data storage with asymmetric errors," *IEEE Access*, vol. 7, pp. 84107–84121, Jun. 2019.

Seong-Joon Park received the B.S. degree in Electronics Engineering from Sogang University, Seoul, Korea, in 2019. He is currently pursuing the Ph.D. degree in Electrical and Computer Engineering from Seoul National University, Seoul, Korea. His current research interests include error-correcting codes and DNA storage.

Yongwoo Lee received the B.S. degree in Electrical Engineering and Computer Science from Gwangju Institute of Science and Technology, Gwangju, Korea, in 2015, and M.S. and Ph.D. degree in Electrical and Computer Engineering from Seoul National University, Korea, in 2017 and 2021, respectively. He is currently working as a Staff Researcher at the Samsung Advanced Institute of Technology. His current research interests include homomorphic encryption, code-based cryptography, and DNA storage.

Jong-Seon No received the B.S. and M.S.E.E. degrees in Electronics Engineering from Seoul National University, Seoul, Korea, in 1981 and 1984, respectively, and the Ph.D. degree in electrical engineering from the University of Southern California, Los Angeles, CA, USA, in 1988. He was a Senior MTS with Hughes Network Systems from 1988 to 1990. He was an Associate Professor with the Department of Electronic Engineering, Konkuk University, Seoul, from 1990 to 1999. He joined the faculty of the Department of Electrical and Computer Engineering, Seoul National University, in 1999, where he is currently a Professor. He became an IEEE Fellow through the IEEE Information Theory Society in 2012. He was a Recipient of the IEEE Information Theory Society Chapter of the Year Award in 2007. From 1996 to 2008, he served as a Founding Chair of the Seoul Chapter of the IEEE Information Theory Society. He was a General Chair of Sequence and Their Applications 2004 (SETA2004), Seoul. He also served as a General Co-Chair of the International Symposium on Information Theory and Its Applications 2006 (ISITA2006) and the International Symposium on Information Theory 2009 (ISIT2009), Seoul. He served as a Co-Editor-in-Chief of the IEEE JOURNAL OF COMMUNICATIONS AND NETWORKS from 2012 to 2013. He became a Member of the National Academy of Engineering of Korea (NAEK), 2015, where he served as the Division Chair of Electrical, Electronic, and Information Engineering, from 2019 to 2020. His area of research interests includes error-correcting codes, cryptography, sequences, LDPC codes, interference alignment, and wireless communication systems.