This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/LCOMM.2019.2912572, IEEE Communications Letters

Construction of Bio-constrained Code for DNA Data Storage

Yixin Wang, Md. Noor-A-Rahim, Erry Gunawan, Yong Liang Guan, Chueh Loo Poh

***Abstract*—With extremely high density and durable preservation, DNA data storage has become one of the most cutting-edge techniques for long-term data storage. Similar to traditional storage which impose restrictions on the form of encoded data, data stored in DNA storage systems are also subject to two biochemical constraints, i.e., maximum homopolymer run limit and balanced GC content limit. Previous studies used successive process to satisfy these two constraints. As a result, the process suffers low efficiency and high complexity. In this paper, we propose a novel content-balanced run-length limited (C-RLL) code with an efficient code construction method, which generates short DNA sequences that satisfy both constraints at one time. Besides, we develop an encoding method to map binary data into long DNA sequences for DNA data storage, which ensures both local and global stability in terms of satisfying the biochemical constraints. The proposed encoding method has high effective code rate of 1.917 bits per nucleotide and low coding complexity.**

***Keywords*—DNA data storage, constrained code, Run-length limited code, long term data storage.**

I. Introduction

With four molecular units named *nucleotide* (nt), including Adenine ('A'), Thymine('T'), Cytosine('C') and Guanine('G'), DNA as a storage medium provides a twice larger capacity than binary systems, while with an information density in magnitude of petabyte. Meanwhile, DNA-based data storage can be preserved for many years under favorable conditions [1]. Attracted by these features, several proof-of-principle DNA storage schemes have been implemented [2–10].

In DNA data storage, data is stored in the form of *oligo* which represents a string of nucleotides of length around 200nt. It has been found that oligos with high/low GC content or long homopolymer runs are prone to sequencing errors [11]. We thus consider that the DNA sequence has global stability (i.e., robust against errors in the processes of data storage), if it has no homopolymer run larger than 3 and GC content of around 50% and has local stability, if its short component blocks (sub-strings) also satisfy the constraints.

In [8], two bits were directly mapped to one nucleotide with code rate close to the theoretical channel capacity, while the iterative post-processing impedes the approach as it may cause severe error propagation in decoding [12]. The authors in [13] designed capacity-approaching constrained codes that avoid long homopolymer runs using a sequence replacement method, but neglected the GC content constraint. [12] presents a scheme to concatenate short codes that only satisfy homopolymer run constraint into long DNA sequences that satisfy both constraints, achieving 1.9 bits/nt code rate. However, the encoding method suffers high complexity, and the encoded long DNA sequences cannot guarantee a balanced GC content in the short component blocks.

In this work, we have devised a new bio-constrained code called content-balanced run-length limited (C-RLL) code. A key advantage of the proposed method is that it constructs codes satisfying both biochemical constraints at one time. We constructed C-RLL codes of lengths from 8 to 12. Based on these codes, we developed an encoding scheme to map binary data to long DNA sequences. As a result, not only each encoded long DNA sequence but also its component sub-strings satisfy the two biochemical constraints (global and local stability). Compared to previous works, our proposed encoding method achieves a high effective code rate of 1.917 bits/nt with lower coding complexity. In addition, the encoded DNA sequences have both global and local stability.

II. Coding with Homopolymer Run Constraint

Homopolymer run constraint in DNA data storage restricts the maximum number of consecutively repetitive nucledotides in the encoded DNA sequences. Similar constraints exist in traditional recordings, in which the stored data are required to avoid certain patterns of repetitions (run-lengths) [14]. Codes that satisfy the run-length limits are known as run-length limited (RLL) codes [15]. $(M, d+1, k+1)$ RLL codes that define M-ary codes with at least $d+1$ run-length and at most $k+1$ run-length can be generated from (M, d, k) constrained codes that define M-ary codes with at least d and at most k zeros between consecutive non-zeros. (M, d, k) codes can be constructed by a state transition diagram, in which the state s_i, where $i \in \{0, 1, ..., k\}$, records k consecutively repetitive zeros in the output sequence. Codewords in the constructed (M, d, k) code can be regarded as transition sequences which are then fed to an M-mod precoder to generate the RLL code via $y_i = y_{i-1} + x_i \ (mod\ M)$, where y_i is the current precoding symbol, y_{i-1} is the last precoded symbol and x_i is the current transition symbol in the transition sequence.

We consider the DNA data storage as an (M, d, k) constrained system, where $M = 4$, $d = 0$ and $k = 2$ as the maximum homopolymer run is 3nt (which refers to $k+1$) and no limit on the minimum homopolymer run. With $\mathbb{Z}_4 = \{0, 1, 2, 3\}$ to represent 4 transition symbols, e.g., transiting 'A' to 'A', 'C', 'T' and 'G' using 4 different symbols, we generate the finite state transition diagram (FSTD) of the homopolymer-constrained DNA system as illustrated in Fig. 1. Note that transition sequences of any arbitrary length can be

Yixin Wang, Erry Gunawan and Yong Liang Guan are with the School of Electrical and Electronic Engineering (EEE), Nanyang Technological University (NTU), Singapore (E-mail: {ywang065, egunawan, eylguan}@ntu.edu.sg). Md. Noor-A-Rahim is with the School of Computer Science and IT, University College Cork, Ireland (E-mail: m.rahim@cs.ucc.ie). Chueh Loo Poh is with the Department of Biomedical Engineering, National University of Singapore (NUS), Singapore (E-mail: poh.chuehloo@nus.edu.sg). (Corresponding author: Chueh Loo Poh)

This work was supported in part, by the NUS Startup grant and by the EDGE COFUND Marie Sklodowska Curie grant (agreement No. 713567).

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/LCOMM.2019.2912572, IEEE Communications Letters

generated by the labels of the edges of a path in the FSTD and subsequently transformed into the RLL sequences via the precoder. Next, We introduce an approach to ensure the resultant RLL sequences also satisfy the balanced GC content.

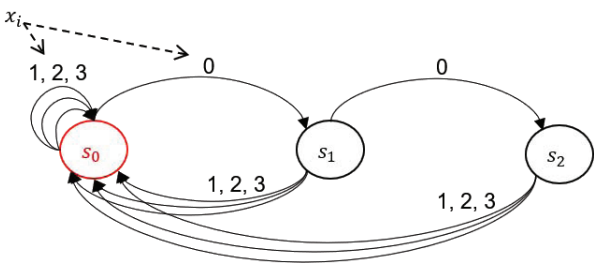

Fig. 1: Finite state transition diagram of (4, 0, 2) constrained DNA storage.

III. Content Balanced RLL Code Construction

The following definitions and notations are used in the proposed code construction:

Content state: Recall that GC content is the ratio of the number of G and C against the total number of symbols in a DNA sequence. We define two *content pairs*, $\{A, T\}$, $\{G, C\}$, of which symbols in the same pair count for the same content, i.e., AT content and GC content. With the mapping $A \mapsto 0, C \mapsto 1, T \mapsto 2, G \mapsto 3$ in the ring $\mathbb{Z}_4$ of integers modulo 4, when adding 0 or 2 to any symbol, the symbol will be transited to itself or its partner in the same content pair, while when adding 1 or 3, it will be transited to symbols in the other content pair. We thus denote $\{0, 1, 2, 3\}$ as a content-based transitions set, in which 0 and 2 transform a symbol to the same content pair, increasing the symbol's corresponding content, while 1 and 3 transform a symbol to the other content pair, reducing the corresponding content. The change of content led by the DNA transition step is represented by the change of *content state*.

Basic transition word set (BTWS): As shown in Fig. 1, one can construct a finite set consisting of words by starting from and ending with state s_0. These words can be arbitrarily concatenated into long transition sequences. We denote this set, $\{1, 2, 3, 01, 02, 03, 001, 002, 003\}$, as the BTWS $\mathbb{X}$.

Content disparity: It is defined by the difference between the number of A & T and the number of G & C of an n-length DNA sequence, denoted as $\Delta(n)$. For simplicity, we assume n is an even number. For 50% GC content, $\Delta(n) = 0$, while for 40%-60% GC content, $-(2\lfloor\frac{3n}{5}\rfloor - n) \leq \Delta(n) \leq (2\lfloor\frac{3n}{5}\rfloor - n)$.

Instead of the single quaternary symbols that are used in the conventional FSTD, we generate a reduced FSTD in Fig. 2 based on elements of the BTWS, where cs_1 and cs_2 are two DNA content states. According to this FSTD, a state transition function is defined to indicate the changes of content states that are led by the new concatenated words,

$$\alpha(S_i) = \begin{cases} \alpha(S_{i-1}) & \hat{s}_i \in \{0, 2\} \\ -\alpha(S_{i-1}) & \hat{s}_i \in \{1, 3\} \end{cases}, \quad \text{for } i > 0 \qquad (1)$$

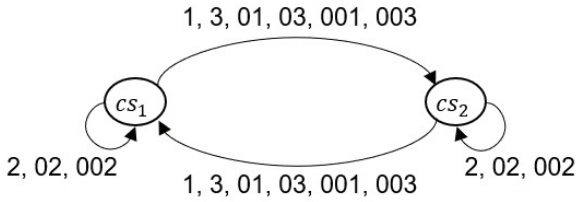

Fig. 2: A reduced finite state transition diagram based on the basic transition word set (BTWS).

where $\hat{s}_i$ represents the last symbol of i^{th} concatenated word S_i $(i > 0)$, and the initial content state $\alpha(S_0)$ is set to be 1.

In the concatenation, for each symbol of the new concatenated transition word from the BTWS, the content state used for computing the content contribution in the resultant C-RLL code remains to be the one that is passed from the last concatenated word. Thus, the content contribution of each concatenated word can be identified as a content weight,

$$W(X) = \sum_{j=1}^{|X|} w(x_j), \ X \in \mathbb{X} \qquad (2)$$

where $w(x)$ equals 1 for symbol $x \in \{0,\ 2\}$ and equals -1 for symbol $x \in \{1,\ 3\}$. $|X|$ and x_j are the length and the j^{th} symbol of the transition word X, respectively. A content weight array $\mathbb{W} = \{-1, 1, -1, 0, 2, 0, 1, 3, 1\}$ is thus obtained, elements of which measure the content contributions of corresponding transition words in the C-RLL code. Afterwards, a content disparity function is used to ensure the encoded code with a balanced GC content,

$$\Delta(n) = \sum_{i=1}^{l} \alpha(S_{i-1}) W(S_i), \ \sum_{i=1}^{l} |S_i| = n, \ S_i \in \mathbb{X} \qquad (3)$$

where n is the length of the codeword, $|S_i|$ is the length of the i^{th} concatenated word S_i, $\alpha(.)$ follows (1), and $W(.)$ follows (2). With a predefined codeword length n and a content disparity parameter $\Delta(n) = \delta$, the resultant C-RLL code $\mathbb{C}(\delta, n)$ can be constructed using the precoding operation on the transition sequence set $\mathbb{S}(\delta, n)$, which consists of transition sequences $\mathcal{S} = S_1 S_2 S_3 ..... S_{l-1} S_l$. This process is denoted by $\mathbb{C}(\delta, n) \Leftarrow \mathbb{S}(\delta, n)$.

We notice that some valid codewords[1] cannot be directly constructed via the above approach due to the following two reasons brought by FSTD. Firstly, all transition sequences generated based on the BTWS end with a non-zero symbol. Thus, all resultant C-RLL codes have different symbols at the last two positions. Second, there is no codeword that begins with three consecutively repetitive initial symbols in the resultant C-RLL codes. This is because no transition word has a number of zeros more than 2 that can be used as the first concatenated word to produce a 3nt repetition of the initial symbols at the beginning of the resultant codewords. To have a complete investigation into the cardinality of the

[1]Codewords that satisfy specific content constraint and RLL limit.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/LCOMM.2019.2912572, IEEE Communications Letters

bio-constrained code, we now explain how to overcome the limitation of FSTD and retrieve the codewords that are missed by the finite-length code construction using FSTD.

To retrieve the lost valid codewords due to the first reason, in the construction process, we attach one extra zero symbol at the end of the transition sequences of length $n-1$ and two extra zeros at the end of the transition sequences of length $n-2$ before checking the content disparity. The updated content disparity are calculated based on the following function with parameter μ_0 to be $\alpha(S_l)$ and $2\alpha(S_l)$, respectively.

$$\Delta(n) = \sum_{i=1}^{l} \alpha(S_{i-1})W(S_i) + \mu_0 w(0) \tag{4}$$

Similarly, we address the second issue, aiming to retrieve valid transition sequences that have three heading zero symbols. To avoid repeated generations, only the sequences that begin with two zero symbols are used in the following operations. We attach one, two and three extra zero symbols to sequences of length $n-1$, $n-2$ and $n-3$, respectively. For each attaching operation, only one zero is attached at the head, while other zeros are attached at the end. The computations of the content disparity follows (4) with parameters μ_0 to be 1, $1+\alpha(S_l)$ and $1+2\alpha(S_l)$, respectively.

Algorithm 1 is the pseudo code for constructing complete C-RLL codes that satisfy 3nt maximum homopolymer run and 40%-60% GC content. The following notations are used:

- $\vec{S}$: A transition sequence set, of which each sequence consists of words from the BTWS $\mathbb{X}$.
- $\vec{C_s}$: The content state vector of $\vec{S}$.
- $\vec{C_d}$: The content disparity vector of $\vec{S}$.
- $|sx|$: The length of the concatenation of words s and x.

The algorithm includes two steps, one is to generate the transition sequences, and the other is to construct the bio-constrained C-RLL code using precoding and DNA mapping. In every outer loop of Algorithm 1, $\vec{S}$, $\vec{C_d}$ and $\vec{C_s}$ are initially reset empty after passing values to corresponding intermediate vectors. In every inner loop, only $s_i x_j$ that satisfies the conditions in line 14 is added to $\vec{S}$. Meanwhile, the content disparity and content state of the valid $s_i x_j$ that have been computed in line 6 and line 8, respectively, are added to $\vec{C_d}$, $\vec{C_s}$, respectively.

Algorithm 1: Complete C-RLL Code Construction

Input : $\mathbb{X}$, $\mathbb{W}$, n, $\delta = 2\lfloor \frac{3n}{5} \rfloor - n$

1 **Step 1: Generating transition sequences**

2 **Initialization:** $\vec{S} = \mathbb{X}$, $\vec{C_d} = \mathbb{W}$, $count = 1$, $\mathbb{S}(n,\delta) = \emptyset$, $\vec{C_s} = \{-1,1,-1,-1,1,-1,-1,1,-1\}$

3 **while** $count < n$ **do**

4 **for** s_i *in* $\vec{S}$ **do**

5 **for** x_j *in* $\mathbb{X}$ **do**

6 **Compute** the content disparity C_d of $s_i x_j$ (refer to (3))

7 **if** $|s_i x_j| < n$ **then**

8 **Update** the content state C_s (refer to (1))

9 **if** $|s_i x_j| \in \{n-1, n-1, n-3\} \& -\delta \leq$ *updated* $C_d \leq \delta$ **then**

10 **Add** lost valid sequences to $\mathbb{S}(n,\delta)$

11 /* Conditional updates. */

12 $\delta_t = n - |s_i x_j| + \delta$

13 **if** $|s_i x_j| < \frac{n}{2}$ *or* $-\delta_t \leq C_d \leq \delta_t$ **then**

14 **Update** $\vec{S}$, $\vec{C_d}$, $\vec{C_s}$

15 **else if** $|s_i x_j| == n$ **then**

16 **if** $-\delta \leq C_d \leq \delta$ **then**

17 **Add** $s_i x_j$ to $\mathbb{S}(n,\delta)$

18 count = count+1

19 **Step 2: Precoding and mapping to C-RLL code**

20 **Initialization:** $DNA_{set} = \{A, C, T, G\}$

21 **for** $\mathcal{S}$ *in* $\mathbb{S}(n,\delta)$ **do**

22 Perform precoding $\rightarrow$ Y

23 /* Convert quaternary symbols to DNA symbols. */

24 **for** y_i *in* $\mathcal{Y}$ **do**

25 $z_i = DNA_{set}(y_i)$

26 $\mathcal{Z} = z_1 z_2 z_3 ... z_i \in \mathbb{C}(n,\delta)$

Output: Bio-constrained C-RLL Code $\mathbb{C}(n,\delta)$

IV. Concatenated coding for Long DNA Sequence

We construct the long DNA sequence by concatenating the codewords from the C-RLL code. Note that the short C-RLL codewords cannot be arbitrarily concatenated into a long sequence as the resultant sequence may disobey the homopolymer constraint in the conjunctions of two short component codewords. First, we eliminate the codewords ending with 3 repetitive symbols from the candidate component codeword set. Based on the updated candidate set, the concatenated encoding performs in serial, in which each DNA component block is encoded by considering the binary data block B_i and the last two symbols of the previous encoded block $\widetilde{C}_{i-1}$. The encoding function f_e encodes $kN(n)$ bits to kn DNA symbols, denoted as f_e:$\mathbb{Z}_2^{kN(n)} \rightarrow \mathbb{Z}_4^{kn}$. We use $\mathcal{B} = B_1 B_2 ... B_k \in \mathbb{Z}_2^{kN(n)}$ to denote a binary sequence of length $kN(n)$ that consists of k binary blocks of length $N(n)$. As $N(n)$ depends on the maximum number of component codewords that can be arbitrary concatenated to any component codeword, $N(n)$ is computed by

$$N(n) = \lfloor \log_2 min\{Q(\circ\circ, n), Q(\circ\bullet, n)\} \rfloor \tag{5}$$

where $Q(\circ\circ, n)$ and $Q(\circ\bullet, n)$ represent the numbers of component codewords that can concatenate with codewords ending with the pattern of two repetitive symbols and two varied symbols without disobeying the homopolymer run constraint, respectively. Afterward, the $f_e(\mathcal{B}) = C_1 C_2 ... C_k$ is defined such that,

$$C_i = \begin{cases} \Gamma_o(B_i) & i = 1 \\ \Gamma_{\widetilde{C}_{i-1}}(B_i) & i = 2, 3, ..., k \end{cases}$$

where $\widetilde{C}_{i-1} = c_{i-1}^{n-1} c_{i-1}^{n} \in \mathbb{Z}_4^2$, is the last two symbols of the codwords $C_{i-1} = c_{i-1}^1 c_{i-1}^2 ... c_{i-1}^{n-1} c_{i-1}^n$, and $\Gamma_\Theta(B_i) : \mathbb{Z}_2^{N(n)} \rightarrow \mathbb{Z}_4^n$. For simplicity, the $\Gamma_\Theta(.)$ can be the lexicographic encoding, which directly constructs an one-to-one mapping between each $N(n)$-bit binary sequence B_i to each n-nt valid DNA sequence C_i

V. Results

With the constraints of 3nt maximum homopolymer run and 40%-60% GC content, we construct the proposed C-RLL code

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/LCOMM.2019.2912572, IEEE Communications Letters

TABLE I: Short C-RLL code

n	Q	R
8	17,608	1.763
9	124,816	1.881
10	653,896	1.932
11	1,792,992	1.889
12	9,562,368	1.932

TABLE II: Concatenated long bio-constrained DNA storage code

n	Q	$Q(\circ\circ)$	$Q(\circ\bullet)$	R	N	R_e
8	17,056	16,172	16,928	1.748	13	1.625
10	628,456	593,488	622,380	1.918	19	1.900
12	9,178,232	8,662,056	9,086,470	1.921	23	**1.917**

of lengths from 8 to 12 based on the construction approach discussed in Section III. The code cardinality Q and code rate R of the short C-RLL code are shown in Table. I.

With the C-RLL codes of length $8,\ 10$ and 12, we used them as the component codewords in concatenated encoding for constructing long DNA storage code. In Table. II, Q is the cardinality of the component code $\mathbb{C}_{ele}$ gained after eliminating codewords that end with 3nt repetitions from the constructed C-RLL code. $Q(\circ\circ)$ and $Q(\circ\bullet)$ are counted by excluding codewords that cannot concatenate to any corresponding pattern from $\mathbb{C}_{ele}$. The code rate R of the long concatenated code is calculated by $R = log_2 min\{Q(\circ\circ), Q(\circ\bullet)\}/n$. N is the number of binary bits that can be encoded in each component codeword of length n, which is computed based on (5). $R_e = \frac{N}{n}$ is the effective code rate of the concatenated encoding scheme, which represents the number of binary bits encoded in one nucleotide.

Our effective code rate 1.917 bits/nt is higher than the 1.90 bits/nt reported in [12]. Besides, the encoding method exhibits lower coding complexity, as [12] used the last three symbols for concatenated encoding leading to a complexity $(O(3k))$, while we only use the last two symbols with corresponding complexity order $(O(2k))$. The reduced coding complexity is due to the new observation on the effective component codewords. Moreover, the authors in [12] sorted the codeword set based on the GC content to select codewords with mostly balanced GC content. This sorting resulted in redundant complexity. As the short codes we constructed already satisfied both constraints, our long sequence construction refrains from the sorting complexity and both the long sequence and its component short blocks satisfy two biochemical constraints.

VI. Conclusion

We have proposed a novel bio-constrained C-RLL code, which satisfies both maximum homopolymer run and balanced GC content constraints. The proposed method provides a parallel process to construct the codes efficiently. In addition, we have devised an encoding scheme to map binary data to long bio-constrained DNA sequences based on the constructed short C-RLL codes. The proposed encoding scheme guarantees the encoded long DNA sequences have local and global stability in terms of satisfying the biochemical constraints. The proposed approach achieves a high effective code rate of 1.917 bits/nt with lower encoding complexity compared to previous works.

References

[1] C. Bancroft, T. Bowler, B. Bloom, and C. T. Clelland, "Long-term storage of information in DNA," *Science*, vol. 293, no. 5536, pp. 1763–1765, 2001.

[2] G. M. Church, Y. Gao, and S. Kosuri, "Next-generation digital information storage in DNA," *Science*, p. 1226355, 2012.

[3] 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, 2013.

[4] R. N. Grass, R. Heckel, M. Puddu, D. Paunescu, and W. J. Stark, "Robust chemical preservation of digital information on DNA in silica with error-correcting codes," *Angewandte Chemie International Edition*, vol. 54, no. 8, pp. 2552–2555, 2015.

[5] S. H. T. Yazdi, Y. Yuan, J. Ma, H. Zhao, and O. Milenkovic, "A rewritable, random-access DNA-based storage system," *Scientific reports*, vol. 5, p. 14138, 2015.

[6] M. Blawat, K. Gaedke, I. Huetter, X.-M. Chen, B. Turczyk, S. Inverso, B. W. Pruitt, and G. M. Church, "Forward error correction for DNA data storage," *Procedia Computer Science*, vol. 80, pp. 1011–1022, 2016.

[7] J. Bornholt, R. Lopez, D. M. Carmean, L. Ceze, G. Seelig, and K. Strauss, "A DNA-based archival storage system," *ACM SIGOPS Operating Systems Review*, vol. 50, no. 2, pp. 637–649, 2016.

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

[9] S. H. T. Yazdi, R. Gabrys, and O. Milenkovic, "Portable and error-free DNA-based data storage," *Scientific reports*, vol. 7, no. 1, p. 5011, 2017.

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

[11] 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 biology*, vol. 14, no. 5, p. R51, 2013.

[12] W. Song, K. Cai, M. Zhang, and C. Yuen, "Codes with run-length and gc-content constraints for DNA-based data storage," *IEEE Communications Letters*, vol. 22, no. 10, pp. 2004–2007, 2018.

[13] K. A. S. Immink and K. Cai, "Design of capacity-approaching constrained codes for DNA-based storage systems," *IEEE Communications Letters*, vol. 22, no. 2, pp. 224–227, 2018.

[14] K. A. S. Immink, *Codes for mass data storage systems*. Shannon Foundation Publisher, 2004.

[15] K. S. Immink, "Runlength-limited sequences," *Proceedings of the IEEE*, vol. 78, no. 11, pp. 1745–1759, 1990.