Design of Capacity-Approaching Constrained Codes for DNA-Based Storage Systems

Kees A. Schouhamer Immink, *Fellow, IEEE*, and Kui Cai, *Senior Member, IEEE*

***Abstract*— We consider coding techniques that limit the lengths of homopolymer runs in strands of nucleotides used in DNA-based mass data storage systems. We compute the maximum number of user bits that can be stored per nucleotide when a maximum homopolymer runlength constraint is imposed. We describe simple and efficient implementations of coding techniques that avoid the occurrence of long homopolymers, and the rates of the constructed codes are close to the theoretical maximum. The proposed sequence replacement method for k-constrained q-ary data yields a significant improvement in coding redundancy than the prior art sequence replacement method for the k-constrained binary data. Using a simple transformation, standard binary maximum runlength limited sequences can be transformed into maximum runlength limited q-ary sequences which opens the door to applying the vast prior art binary code constructions to DNA-based storage.**

***Index Terms*— Data storage systems, channel coding.**

I. Introduction

THE first large-scale archival DNA-based storage architecture was implemented by Church *et al.* [1] in 2012. Naturally occurring DNA consists of four types of *nucleotides* (nt): adenine (A), cytosine (C), guanine (G), and thymine (T). A DNA strand (or string) is a linear sequence of these nucleotides, and hence is essentially a q-ary sequence with $q = 4$. Binary source, or user, data is translated into a strand of nucleotides, for example, by mapping two binary source bits into a single nucleotide. Repetitions of the same nucleotide, a homopolymer run, may significantly increase the chance of sequencing errors [2], [10]. From [10, Fig. 5], a long homopolymer run (e.g. more than 4 nt) would result in a significant increase of insertion and deletion errors, so that such long runs should be avoided.

In this letter, we focus on constrained coding techniques that avoid the occurrence of long homopolymer runs. That is, we will study the generation of sequences of q-ary symbols, $\ldots, x_{i-1}, x_i, x_{i+1}, \ldots$, $x_i \in \mathcal{Q} = \{0, \ldots, q-1\}$, where the occurrence of vexatious substrings is disallowed. Note that we prefer for the DNA case, $q = 4$, the usage of the alphabet $\mathcal{Q} = \{0, 1, 2, 3\}$ instead of the set of four nucleotide types $\{A, C, G, T\}$ as it allows the introduction of arithmetic operations on the symbols.

Manuscript received October 3, 2017; revised November 1, 2017 and November 15, 2017; accepted November 15, 2017. Date of publication November 20, 2017; date of current version February 9, 2018. This work was supported by the SUTD-MIT International Design Center (IDC) research grant. The associate editor coordinating the review of this letter and approving it for publication was A. Noel. *(Corresponding author: Kees A. Schouhamer Immink.)*

K. A. Schouhamer Immink is with Turing Machines Inc., 3016 DK Rotterdam, The Netherlands (e-mail: immink@turing-machines.com).

K. Cai is with the Singapore University of Technology and Design, Singapore 487372 (e-mail: cai_kui@sutd.edu.sg).

Digital Object Identifier 10.1109/LCOMM.2017.2775608

Constrained sequences have been applied in a great number of mass data storage systems such as optical and magnetic data recording systems [3]. Constrained codes based on *runlength limited* (RLL) sequences have found almost universal application in recording practice, and most of the codes are binary with $q = 2$. The number of repetitions of the same consecutive symbol (nucleotide) is usually called *runlength* [4]. A maximum runlength constraint is characterized by the integer $(k + 1)$, $k \geq 0$, which stipulates the maximum runlength. We focus on sequences, where the 'zero' runlength lies between d and k. Such a sequence is often called a dk-constrained sequence, and in case $d = 0$, it is called a k-constrained sequence. A k-constrained sequence is converted into an RLL sequence whose maximum runlength equals $k+1$, using *precoding*, a modulo-q integration step [3]. The notation k versus $k+1$ for a k-constrained sequence versus a $k+1$ RLL sequence is inconvenient, but the term is generally used in data recording practice, and is a heritage rooted in the 1960s [5]. We use the notation $m = k+1$ to denote a maximum homopolymer run of m nt.

Bornholt *et al.* [2] presented a coding method that avoids the occurrence of repetitions of the same nucleotide for DNA-based storage. They convert binary user data into a $(k = 0)$-constrained ternary data stream using a base-change converter, where the generated ternary data are taken from the alphabet $\{1, 2, 3\}$. The ternary data so obtained are translated using modulo-4 integration precoding into a strand of nucleotides, where homopolymers are avoided, $m = 1$, that is substrings '00', '11', '22, '33' (or in nucleotide language: 'AA', 'CC', 'GG', and 'TT') are not generated. The relative loss of information capacity due to the proposed 3-base code, instead of the full 4-base, equals $1 - \log_2(3)/2 \sim 0.208$. The additional loss of the proposed binary-based source word to ternary-based codeword conversion using the proposed fixed-to-variable-length Huffman code is ignored. The more than 20 percent loss of information capacity is significant, and therefore alternative coding methods with less overhead are desirable.

In this letter, we propose alternative, more efficient, coding techniques that avoid the occurrence of long homopolymer runs. In particular, in Section II, we compute the information capacity of q-ary, k-constrained channels, which follows directly from Shannon's noiseless input restricted channel [6]. Then, in Section III, we present the main contribution of this work, algorithms for translating arbitrary binary source data into k-constrained q-ary data. Among the three code design methods we describe, the second method removes forbidden substrings of q-ary sequences by using a recursive, 'sequence replacement', method yielding a significant improvement

TABLE I

CAPACITY, C_k, VERSUS k AND $m = k + 1$ FOR $q = 4$.

k	m	C_k(bit/nt)
0	1	$1.5850(=\log_2 3)$
1	2	1.9227
2	3	1.9824
3	4	1.9957
4	5	1.9989
5	6	1.9997

TABLE II

MAXIMUM LENGTHS n FOR WHICH A RATE $(n-1)/n$, k-CONSTRAINED 4-ARY CODE CAN BE CONSTRUCTED

k	$m = k + 1$	$n_{\max}$	n_A	n_B
1	2	25	22	11
2	3	113	106	39
3	4	467	445	148
4	5	1885	1848	581

in coding redundancy than the prior art binary sequence replacement method [9]. In the third method, standard binary maximum runlength limited sequences are transformed into maximum runlength limited q-ary sequences using two simple steps of precoding, which opens the door to using the vast prior art binary code constructions to DNA-based storage. Section IV concludes our letter.

II. INFORMATION CAPACITY

Strands of nucleotides with (long) repetitions of the same nucleotide are prone to error, and DNA sequences with more than $m = k + 1$ consecutive nucleotides of the same type must be avoided. Each k-constrained sequence of symbols starting with a non-zero symbol can be seen to be composed of substrings taken from the set $\{a_0, a_1 0, a_2 0^2, \ldots, a_k 0^k\}$, where 0^j stands for a string of j consecutive '0's, and the integer $a_i \in \{1, \ldots, q-1\}$. Let $N_k(n)$ denote the number of k-constrained sequences of q-ary symbols starting with a non-zero symbol, then we may write down, following Shannon's approach [6], the recurrent relationship

$$N_k(n) = (q-1)\sum_{i=1}^{k+1} N_k(n-i), \quad n > k. \tag{1}$$

For large n, the number of sequences $N_k(n)$ grows exponentially, that is

$$N_k(n) \sim c\lambda_k^n, \quad n \gg 1, \tag{2}$$

where $c \sim 1$ is a constant, and the *growth factor,* λ_k, is the largest real root of

$$\lambda_k^{k+2} - q\lambda_k^{k+1} + q - 1 = 0. \tag{3}$$

The maximum number of user bits that can be stored per nucleotide (nt), called (information) capacity, denoted by C_k, is defined by

$$C_k = \lim_{n\to\infty} \frac{1}{n} \log_2 \ N_k(n) = \log_2 \lambda_k \ \text{(bit/nt)}. \tag{4}$$

Table I shows the capacity of the k-constrained channel, C_k, versus k and the maximum homopolymer run $m = k + 1$, for the DNA case, $q = 4$. For asymptotically large k, we obtain

$$\lambda_k \sim q\left(1 - \frac{q-1}{q^{k+2}}\right), \quad k \gg 1, \tag{5}$$

so that

$$C_k \sim \log_2 \ q - \frac{1}{\ln 2}\frac{q-1}{q^2} q^{-k}. \tag{6}$$

It is immediate from Table I that a relaxation of the maximum homopolymer runlength constraint from $m = 1$, a value proposed in [2], to a higher value may significantly increase the maximum code rate. We are interested in k-constrained code constructions of rate $(n-1)/n$, where n is the codeword length. Define the integer $n_{\max}$ as the largest n for which a rate $(n-1)/n$, k-constrained code can be constructed. We simply find that

$$\frac{1}{n_{\max}} \geq 1 - \log_q \lambda_k, \tag{7}$$

or

$$n_{\max} = \left\lfloor \frac{1}{\log_q \frac{q}{\lambda_k}} \right\rfloor. \tag{8}$$

Results of computations are collected in Table II. We may notice that it is possible, in theory, to construct a code with a redundancy of around half a percent, where homopolymers runs have a length at most $m = 4$. A maximum homopolymer run, $m = 3$, costs less than two percent redundancy. In the next section, we investigate properties and constructions of practical codes that translate arbitrary source data into k-constrained q-ary sequences.

III. MAXIMUM RUNLENGTH CONSTRAINED CODES

We detail three methods for generating maximum runlength limited q-ary sequences. In the second method, forbidden substrings of q-ary sequences are removed by a recursive, 'sequence replacement', method. We assume that the binary source data have been translated into q-ary data, which is accomplished by an efficient base converter. In the third method, standard binary maximum runlength limited sequences are transformed into maximum runlength limited q-ary sequences using a simple transformation.

A. Cascadable Block Codes, Method A

Let n be the length of a k-constrained q-ary word that ends with at most r 'zero's and starts with at most l 'zero's. In case $l + r \leq k$ we may cascade the n-words without violating the k constraint at the word boundaries. Blake [7] and Freiman and Wyner [5] showed for the binary case, $q = 2$, that the number of such constrained words, denoted by $N_{klr}(n)$, is maximized by choosing $l = \lfloor k/2 \rfloor$ and $r = k - l$. Their arguments can be generalized to q-ary words, and we denote the number of k-constrained q-ary words by $N_{k,l_0,r_0}(n)$, where $l_0 = \lfloor k/2 \rfloor$ and $r_0 = k - l$. Using generating functions and an algebraic computer program, we can compute $N_{k,l_0,r_0}(n)$ as a function of k and n. As we are interested in the construction of code of maximum rate $1 - 1/n$, we computed the maximum n, denoted by n_A, for which a rate $1 - 1/n$, k-constrained code using Method A, is possible. Results of computations are collected in Table II. For small n it is practically possible to directly

implement RLL codes using look-up tables. For larger codes, and smaller redundancy, we must resort to alternative coding methods. Kautz [8], for example, presented an *enumeration* algorithm for encoding and decoding k-constrained binary sequences. His enumeration algorithm can be rewritten for the q-ary case at hand. The space complexity of his algorithm is $O(n^2)$, which makes it less attractive than the replacement techniques discussed in the next section.

B. Sequence Replacement Technique, Method B

The three sequence replacement techniques published by Wijngaarden and Immink [9] are recursive methods for removing forbidden substrings from a binary source word. The encoder removes the forbidden substrings, and the positions of the forbidden substrings are encoded as binary pointer words, and subsequently inserted at predefined positions of the codeword. The sequence replacement techniques are attractive as the complexity of encoder and decoder is very low, and the methods are very efficient in terms of rate-capacity quotient. In software or hardware, it would require a counter, a comparator, and a few memory elements. The methods are also suited for high speed implementation, as several steps in the encoding and decoding procedure can be performed simultaneously.

We assume that both the source and encoded channel data are represented in the same q-ary base. Let $X = (x_1, \ldots, x_{n-1})$, $x_i \in Q$, be an $(n-1)$symbol source word, which has to be translated into an n-symbol code word $Y = (y_1, \ldots, y_n)$, $y_i \in Q$. Obviously, the rate of the code is $(n-1)/n$. The task of the encoder is to translate the source word into a k-constrained word.

The encoder simply starts by appending a '1' to the $(n-1)$-symbol source word, yielding the n-symbol word, denoted by $X1$. The encoder scans (from right to left, i.e. from LSB to MSB) the word X1, and if this word does not have the forbidden substring 0^{k+1}, the q-ary codeword $Y = X1$ is transmitted. If, on the other hand, the first occurrence of substring 0^{k+1} is found, we invoke the following replacement procedure.

Replacement Procedure: Let the source word be denoted by $X_2 0^{k+1} X_1 1$, where, by assumption, the tail $X1$ has no forbidden substring. The forbidden substring 0^{k+1} is removed, yielding the $(n-k-1)$-symbol sequence $X_2 X_1 1$. Let the forbidden substring, 0^{k+1}, start at position p_1, $1 \le p_1 \le n-k-1$. The position p_1 is represented by the $(k+2)$-symbol q-ary pointer word, $p = v_1 A v_2 0$, where $v_1, v_2 \in Q \setminus \{0\}$ and A is any q-ary word of $k-1$ symbols. Note that the number of unique combinations of pointer p equals $(q-1)^2\ q^{k-1}$. Subsequently, the tail symbol, '1', of $X_2 X_1 1$ is replaced by the $(k+2)$-symbol q-ary string, p, obtaining the sequence $X_2 X_1 p$.

Note that the sequence $X_2 X_1 p$ is of length n (as the starting sequence $X1$). If, after the replacement, the sequence $X_2 X_1 p$ is free of other occurrences of the forbidden substring 0^{k+1} then the codeword $Y = X_2 X_1 p$ is sent. Otherwise, the encoder repeats the above sequence replacement procedure for the string $X_2 X_1 p$ etc., until all forbidden substrings have been removed. The decoder can uniquely undo the various replacements and shifts made by the encoder. The space complexity of the encoder and decoder is mainly the look-up table for translating the position p_1, $1 \le p_1 \le n-k-1$, into the $(k+2)$-symbol q-ary pointer and vice versa, which amounts to $O(n)$.

As the pointer p_1 is in the range $1 \le p_1 \le n-k-1$, and the number of distinct combinations of p_1 equals $(q-1)^2\ q^{k-1}$, we conclude that the codeword length n is upperbounded by

$$n \le (q-1)^2\ q^{k-1} + k + 1, \quad k \ge 2. \tag{9}$$

The code uses one redundant q-ary symbol, so that we conclude that the redundancy of the sequence replacement code is approximately

$$\frac{\log_2 q}{n} \sim \frac{q}{(q-1)^2} q^{-k}, \quad k \gg 1. \tag{10}$$

From (6), we infer that the redundancy of k-constrained q-ary sequences is at least

$$\log_2 q - C_k \sim \frac{1}{\ln 2} \frac{q-1}{q^2} q^{-k}, \quad k \gg 1. \tag{11}$$

The redundancy of the sequence replacement method is a factor of

$$\left(\frac{q}{q-1}\right)^3 \ln q$$

larger than optimal for $k \gg 1$. For DNA-based storage, $q = 4$, the factor is around 3.29. The above sequence replacement method is efficient in terms of redundancy and space/time hardware requirements as no large look-up tables are needed. For example, for a maximum homopolymer run $m = 4$, we are able to construct a code of length $n = 148$ that needs only one redundant nucleotide.

Table II shows results of computations for rate $(n-1)/n$, k-constrained codes for $q = 4$ and various values of k, where n_B denotes the maximum n possible with the sequence replacement method (Method B). Results of computations of $n_{\max}$ have been collected in Table II.

C. Translating Binary k'-Constrained Codes Into Quaternary k-Constrained Sequences, $q = 4$

Very efficient constructions of binary k'-constrained codes that avoid long repetitions of a 'zero' have been published in the literature, see, for example, the survey in [9]. We show that after applying a simple coding step to a k'-constrained binary sequence, we obtain a strand of nucleotides, where the length of a homopolymer run is at most $m = \left\lceil \frac{k'}{2} \right\rceil$.

We start with definitions of two simple operations on symbol sequences and their (unique) inverse. Let $\boldsymbol{x} = (x_1, \ldots, x_n)$, $x_i \in \{0, 1\}$, denote a word of n binary symbols. The first operation is defined as follows. The n-bit sequence, $\boldsymbol{x}$, is translated into a $\frac{n}{2}$-symbol sequence $\boldsymbol{w}$, where two consecutive binary symbols of $\boldsymbol{x}$ are translated into one quaternary symbol $w_i \in \{0, 1, 2, 3\}$, using

$$w_i = 2x_{2i-1} + x_{2i}, \quad 1 \le i \le \frac{n}{2}.$$

The above operation is denoted by the shorthand notation $\boldsymbol{w} = \mathcal{Z}(\boldsymbol{x})$.

The second operation, usually called *precoding*, is defined as follows. The word $\boldsymbol{w} = (w_1, \ldots, w_n)$, $w_i \in \{0, 1\}$ is obtained by modulo 2 *integration* of $\boldsymbol{x}$, that is, by the operation

$$w_i = \bigoplus_{k=1}^{i} x_k = w_{i-1} \oplus x_i, \quad 1 \le i \le n, \tag{12}$$

where the dummy symbol $w_0 = 0$, and the symbol $\oplus$ denotes symbol-wise modulo 2 addition. The above operation is denoted by the shorthand notation $\boldsymbol{w} = I(\boldsymbol{x})$. Note that the original word $\boldsymbol{x}$ can be uniquely restored by a modulo 2 *differentiation* operation, defined by

$$x_i = w_i \oplus w_{i-1}, \quad 1 \le i \le n. \tag{13}$$

The above differentiation operation is denoted by $\boldsymbol{x} = I^{-1}(\boldsymbol{w})$. Clearly,

$$I^{-1}(I(\boldsymbol{x})) = \boldsymbol{x}. \tag{14}$$

Assume that the binary source data have been converted into a binary k'-constrained sequence, $\boldsymbol{x} = (x_1, \ldots, x_n)$, where $x_i \in \{0, 1\}$, using a suitable k'-constrained code. Then, by definition, substrings in $\boldsymbol{x}$ of more than k' consecutive 'zero's, are absent. Note that the operation $\boldsymbol{w} = \mathcal{Z}(\boldsymbol{x})$ will not result in a k-constrained sequence $\boldsymbol{w}$. In order to limit the runlengths of the output word, $\boldsymbol{w}$, we first apply a two-step precoding operation, defined by

$$\boldsymbol{w} = I(I(\boldsymbol{x})). \tag{15}$$

For example, we can easily verify the three operations on the sequence $\boldsymbol{x}$,

$$\begin{aligned}
\boldsymbol{x} &= 01100001111111111110011110000111\\
I(\boldsymbol{x}) &= 01000001010101010111010111 1010\\
I(I(\boldsymbol{x})) &= 01111110011001100101100 1010011\\
\mathcal{Z}(I(I(\boldsymbol{x}))) &= 133212121121103,
\end{aligned}$$

where $\boldsymbol{x}$ is a $(k' = 4)$-constrained binary sequence. After the first precoding step, the sequence, $I(\boldsymbol{x})$, is a regular runlength limited sequence with a maximum 'zero' and 'one' runlength equal to $k' + 1 (= 5)$. The second precoding step limits the number of consecutive 'one's and 'zero's to $k' + 2$ in $I(I(\boldsymbol{x}))$, and it also limits the number of consecutive '10's bits to $k'+2$, thus prohibiting the generation of long homopolymer runs.

In the above example, the 4-ary output sequence $\mathcal{Z}(I(I(\boldsymbol{x})))$ has a maximum homopolymer run, $m = 2$. In general, it can easily be verified that in case the binary input sequence, $\boldsymbol{x}$, is k'-constrained that the 4-ary sequence, $\mathcal{Z}(I(I(\boldsymbol{x})))$, has a maximum homopolymer run given by

$$m = \left\lceil \frac{k'}{2} \right\rceil, \quad k' > 2. \tag{16}$$

The above method offers a simple translation of binary k'-constrained sequences into a strand of nucleotides with limited homopolymer runs, which creates the opportunity to apply the vast literature on binary runlength limited coding to DNA-based storage.

IV. Conclusions

We have presented coding methods for translating source data into strands of nucleotides with a maximum homopolymer run. We found that the proposed algorithms can be implemented efficiently, and that the information densities of the constructed codes are close to the theoretical maximum. We have proposed sequence replacement method for k-constrained q-ary data, which yields a significant improvement in coding redundancy than the prior art sequence replacement method for the k-constrained binary data. We have shown that, using two simple steps of precoding, it is possible to translate a binary k'-constrained sequence into a strand of nucleotides with a maximum homopolymer run, which creates the opportunity to applying a myriad of prior art binary code constructions to DNA-based storage.

References

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

[2] J. Bornholt, R. Lopez, D. M. Carmean, L. Ceze, and G. Seelig, "A DNA-based archival storage system," *ACM SIGARCH Comput. Archit. News*, vol. 44, no. 2, pp. 637–649, 2016.

[3] K. S. Immink, *Codes for Mass Data Storage Systems*, 2nd ed. Eindhoven, The Netherlands: Shannon Foundation Publishers, 2004.

[4] K. A. Schouhamer Immink, "Runlength-limited sequences," *Proc. IEEE*, vol. 78, no. 11, pp. 1745–1759, Nov. 1990.

[5] C. V. Freiman and A. D. Wyner, "Optimum block codes for noiseless input restricted channels," *Inf. Control*, vol. 7, no. 3, pp. 398–415, 1964.

[6] C. E. Shannon, "A mathematical theory of communication," *Bell Syst. Tech. J.*, vol. 27, no. 3, pp. 379–423, Jul. 1948.

[7] I. F. Blake, "The enumeration of certain run length sequences," *Inf. Control*, vol. 55, nos. 1–3, pp. 222–237, 1982.

[8] W. Kautz, "Fibonacci codes for synchronization control," *IEEE Trans. Inf. Theory*, vol. IT-11, no. 2, pp. 284–292, Apr. 1965.

[9] A. J. Van Wijngaarden and K. A. S. Immink, "Construction of maximum run-length limited codes using sequence replacement techniques," *IEEE J. Sel. Areas Commun.*, vol. 28, no. 2, pp. 200–207, Feb. 2010.

[10] M. G. Ross *et al.*, "Characterizing and measuring bias in sequence data," *Genome Biol.*, vol. 14, no. 5, p. R51, 2013.