Constructions and Properties of Efficient DNA Synthesis Codes

Schouhamer Immink, Kees A. ; Cai, Kui; Nguyen, Tuan Thanh; Weber, Jos H.

DOI
10.1109/TMBMC.2024.3401583

Publication date
2024

Document Version
Final published version

Published in
IEEE Transactions on Molecular, Biological, and Multi-Scale Communications

Constructions and Properties of Efficient DNA Synthesis Codes

Kees A. Schouhamer Immink, *Life Fellow, IEEE*, Kui Cai, *Senior Member, IEEE*, Tuan Thanh Nguyen, *Member, IEEE*, and Jos H. Weber, *Senior Member, IEEE*

***Abstract*—We report on coding methods for efficiently synthesizing deoxyribonucleic acid (DNA) for massive data storage, where a plurality of DNA strands are synthesized in parallel. We examine the trade-offs between the information contents, redundancy, and the average or maximum number of cycles required for synthesizing a plurality of parallel DNA strands. We analyze coding methods such as guided scrambling and constrained codes for minimizing the cycle count.**

***Index Terms*—Code design, DNA synthesis, guided scrambling, multiple strands, nibble replacement algorithm.**

I. Introduction

CHURCH et al. described pioneering experiments with DNA based storage systems [1]. Naturally occurring DNA consists of four types of nucleotides: adenine ('A'), cytosine ('C'), guanine ('G'), and thymine ('T'). A DNA oligo is a sequence of these four nucleotides that are composed by DNA synthesizers. In DNA-based storage systems, binary source data are translated into strings (called **strands**) composed of four types of nucleotides, for example, by mapping two binary source symbols into a single nucleotide.

A substantial body of literature has emerged focusing on coding techniques for data storage in DNA. Studies have delved into a spectrum of critical aspects, including error correcting codes for restoring various kinds of defects in DNA, such as substitution, insertions and deletions and so on [2], [3], [4]. Also prior art work has been published on constrained codes that, for example, avoid long substrings of the same nucleotide, called homopolymer runs, unbalance between GC and AT content [5], [6], secondary structures [7], [8], or various combinations of constraints [9]. Due to the high cost of synthesizing strands, more recent research has focused on low-cost synthesis of DNA [10], [11], [12], [13].

In a common type of synthesis process, a plurality of DNA strands are synthesized in parallel. Basically, the synthesizer scans a fixed (predetermined) super sequence of DNA nucleotides and appends a nucleotide at a time to the corresponding DNA strands. As such, in each cycle of synthesis, the machine only appends one nucleotide to a subset of the DNA strands that require that particular nucleotide [10], [11], [14]. Figure 1 depicts an example of the synthesis process of $k = 3$ parallel strands, called x_1, x_2, and x_3, exploiting the fixed super sequence $ACGTACGT\ldots$. The total number of synthesis cycles in this example is 12. Lenz et al. and Elishco and Huleihel [10], [13] showed that the super sequence that maximizes the information rate is the alternating quaternary sequence. The above synthesis of parallel strands can be seen as a queuing problem, where ticket windows named 'A', 'C', 'G', and 'T' are open to serve customers with the same name on a periodic basis.

Our goal in this work is to find coding techniques for translating k source sequences, x_i, into more suitable sequences that limit or minimize the average number of cycles. After an introduction in Section II, we present coding techniques for efficiently synthesizing DNA in Section III. Section IV concludes our contributions.

II. Efficient Synthesis of DNA Nucleotides, Basics

In this section, we start by introducing the main parameters of our paper, namely a) information rate, b) redundancy, and c) the distribution function of the cycle count. We present properties of low-weight codes, which makes it possible to compute the distribution function of the cycle count of a single strand, information rate and redundancy in Sections II-A1 and II-A2. In Section II-B, we compute the cycle count of multiple parallel strands.

We assume that we do not have four types of nucleotides, but q, $q > 1$, types, denoted by $0, 1, 2, \ldots, q-1$. We further assume that k parallel n-symbol q-ary source streams, x_i, $1 \leq i \leq k$, are synthesized using a super sequence which is a cyclic repetition of $0, 1, 2, \ldots, q-1$, for example $012301230123\ldots$ for the case $q = 4$.

Let $T_{\mathbf{x_{s_1}, x_2, \ldots, x_k}}$ denote the number of cycles required to synthesize the k parallel source streams x_i, $i = 1, \ldots, k$. We

Manuscript received 11 December 2023; revised 15 March 2024; accepted 8 May 2024. Date of publication 15 May 2024; date of current version 17 June 2024. The work of Kui Cai and Tuan Thanh Nguyen was supported by the SUTD Kickstarter Initiative (SKI) under Grant 2021_04_05, and in part by the Singapore Ministry of Education Academic Research Fund Tier 2 under Grant T2EP50221-0036. The associate editor coordinating the review of this article and approving it for publication was H. M. Kiah. *(Corresponding author: Kees A. Schouhamer Immink.)*

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

Kui Cai and Tuan Thanh Nguyen are with the Science, Mathematics and Technology Cluster, Singapore University of Technology and Design, Singapore 487372 (e-mail: cai_kui@sutd.edu.sg; TuanThanh_Nguyen@sutd.edu.sg).

Jos H. Weber is with the Applied Mathematics Department, Delft University of Technology, 2628 CD Delft, The Netherlands (e-mail: j.h.weber@tudelft.nl).

Digital Object Identifier 10.1109/TMBMC.2024.3401583

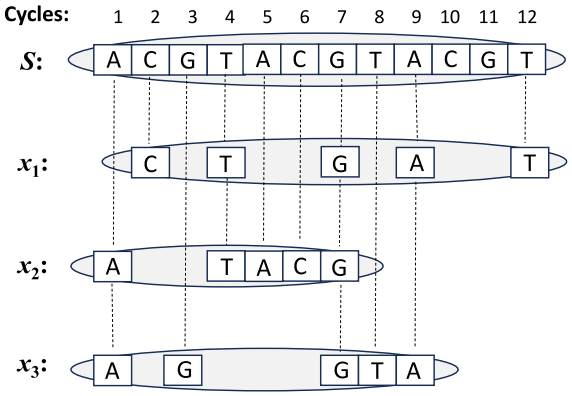


Fig. 1. Basic diagram of DNA synthesis, where $k = 3$ parallel source data streams, x_1, x_2, and x_3, are synthesized using a super sequence S. The total number of synthesis cycles in this example is twelve.

now concentrate on the computation of the cycle number of synthesizing a single strand, $k = 1$. To that end, denote $x = x_1$ and $T = T_{\mathbf{xs}_1}$, where we omitted the subscript for clerical convenience. Then, following [10]

$$T = n + \sum_{i=1}^{n} y_i, \tag{1}$$

where

$$y_i = (x_i - x_{i-1} - 1) \mod q, \tag{2}$$

and $x_0 = q - 1$. Clearly, $n \le T \le qn$. From (2) we infer

$$x_i = \left(x_{i-1} + y_i'\right) \mod q, \tag{3}$$

where $y_i' = y_i + 1$. In the context of magnetic and optical recording, the above 'modulo q integration' operation (3) is called **precoding** [15]. For the binary case, $q = 2$, (3) simplifies into

$$x_i = x_{i-1} \oplus \tilde{y}_i, \tag{4}$$

where $\tilde{y}_i = 1 - y_i$ and $\oplus$ denotes the exclusive-or (xor) operation.

A. Low-Weight Codes

The symbol sum

$$w = \sum_{i=1}^{n} y_i \tag{5}$$

is often called the **weight** of $\boldsymbol{y}$. We may translate a source word into a low-weight codeword $\boldsymbol{y}$ and transform it into a DNA sequence $\boldsymbol{x}$ using (3). Note that the map from $\boldsymbol{x}$ to $\boldsymbol{y}$ is bijective using the mapping (2) or (3).

A fixed-weight code, S_w, of codeword length n consists of codewords with symbol sum, w, that is,

$$S_w = \left\{ \mathbf{y} \in \{\mathbf{0}, \ldots, \mathbf{q-1}\}^{\mathbf{n}} \mathbf{:} \sum_{\mathbf{i=1}}^{\mathbf{n}} \mathbf{y_i} = \mathbf{w} \right\}. \tag{6}$$

The cardinality of S_w, denoted by $|S_w|$, is found as the coefficient of z^w of the **generating function**

$$\left(\sum_{i=0}^{q-1} z^i \right)^n. \tag{7}$$

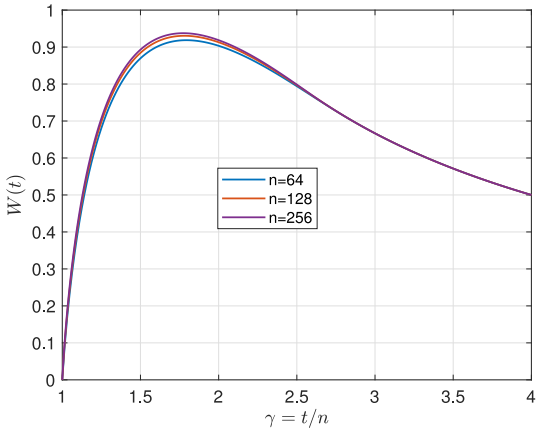


Fig. 2. Information rate $W(t)$ versus $\gamma = t/n$, for $n = 64$, 128, 256 and $q = 4$.

Let the low-weight code, $S(t) = \cup_{w=0}^{t-n} S_w$ denote the union of the sets of codewords of weight $w \le t - n$, $n \le t \le qn$, where the integer t denotes the maximum cycle count. Let

$$f(i) = \frac{|S_{i-n}|}{q^n}, \ n \le i \le qn, \tag{8}$$

and

$$F(i) = \sum_{j=n}^{i} f(j), \ n \le i \le qn, \tag{9}$$

denote the distribution and cumulative cycle count distribution, respectively.

1) Information Rate: A first basic coding parameter in this context was described by Lenz et al. [10], who defined the **information rate** of a low-weight code, $S(t)$, denoted by $W(t)$, as the amount of information (measured in bits) per synthesis cycle, that is,

$$W(t) = \frac{1}{t} \log_2 \sum_{i=n}^{t} |S_{i-n}| = \frac{\log_2 q^n F(t)}{t}. \tag{10}$$

Figure 2 shows the information rate, $W(t)$, of low-weight codes, $S(t)$, versus $\gamma = t/n$ for $n = 64$, 128, 256, and alphabet size $q = 4$. Note that $W(qn) = \log_2(q)/q$, and thus $W(qn) = 1/2$ for $q = 2, 4$.

Since

$$F\left(\frac{q+1}{2} n\right) \approx \frac{1}{2}, \ n \gg 1, \tag{11}$$

we have

$$W\left(\frac{q+1}{2} n\right) \approx 2 \frac{n \log_2 q - 1}{n(q+1)} \approx \frac{2 \log_2 q}{q+1}, \ n \gg 1. \tag{12}$$

2) Redundancy: A second basic coding parameter is the **redundancy** of $S(t)$, denoted by $r(t)$, defined by

$$r(t) = -\log_2 \frac{\sum_{i=0}^{t-n} |S_i|}{q^n} = -\log_2 F(t). \tag{13}$$

Clearly, $r(n) = n \log_2(q)$, $r(qn) = 0$, and, see (11), $r(\frac{q+1}{2} n) \approx 1$. Figure 3 shows the (relative) redundancy, $r(t)/n$, of low-weight codes versus (relative) maximum cycle count

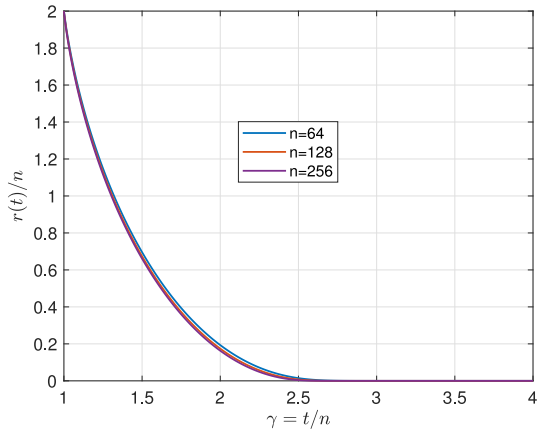


Fig. 3. Relative redundancy, *r(t)*/n, of low-weight codes versus relative maximum cycle count, $\gamma = t/n$, for $n = 64$, 128, 256 and $q = 4$.

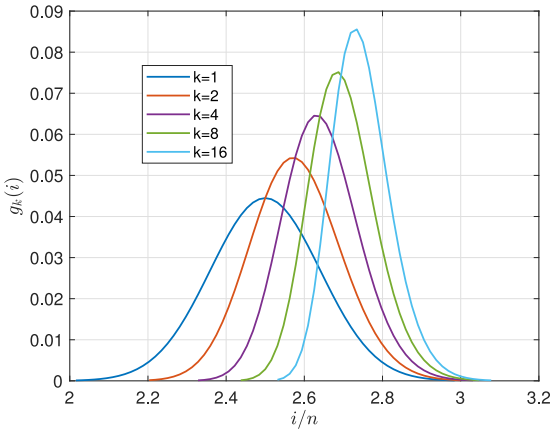


Fig. 4. Distribution, $g_k(i)$, of the cycle count of k multiple strand synthesis of uncoded data versus *i/n* for $n = 64$, $q = 4$, and number of parallel strands, $k = 1$, 2, 4, 8 and 16.

$\gamma = t/n$ for $n = 64$, 128, 256 and $q = 4$. Note that the redundancy, *r*(*t*), is less than 1 bit for $\gamma \geq 5/2$, but the redundancy is rapidly increasing for decreasing values of γ. The increase of *W*(*t*), see Figure 2, from around 0.8 at $\gamma = 5/2$ to its maximum of about 0.94 at $\gamma \approx 1.75$ will come at the cost of a redundancy *r(t)/n* of around 0.4, see Figure 3. It is the province of the system designer to weigh the benefit of a smaller cycle count versus the extra cost and volume of the stored DNA material.

We may write down an approximation of *r*(*t*) by noting that for large n the cycle count distribution $f(t)$ can be usefully approximated by a Gaussian distribution (Central Limit Theorem),

$$f(x) \approx \varphi\left(x; \mu, \sigma^2\right) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}, \tag{14}$$

$$F(t) = \int_{x=-\infty}^{t} \varphi\left(x; \mu, \sigma^2\right) dx, \tag{15}$$

where the cycle count average and variance, denoted by $\mu_o(n)$ and $\sigma_o^2(n)$, respectively, are

$$\mu_o(n) = \sum_{i=n}^{qn} i f(i) = \frac{q+1}{2} n \tag{16}$$

and

$$\sigma_o^2(n) = \sum_{i=n}^{qn} (i - \mu_o(n))^2 f(i) = \frac{q^2-1}{12} n. \tag{17}$$

After the evaluation of a Taylor series of *r(t)* at $t = \mu_o(n)$, using the Gaussian approximation (14), we obtain

$$r(t) \approx 1 + c_1(t - \mu_o(n)) + c_2(t - \mu_o(n))^2, \ n \gg 1, \tag{18}$$

where

$$c_1 = -\frac{1}{\ln(2)}\sqrt{\frac{2}{\pi\sigma^2}} \text{and} c_2 = \frac{1}{\ln(2)}\frac{1}{\pi\sigma^2}.$$

For $q = 4$, we find that $c_1 \approx -1.03/\sqrt{n}$ and $c_2 \approx 0.367/n$.

B. Multiple Parallel Strand Synthesis

We assume that the elements of the sequence $\boldsymbol{y}$ are independent and identically distributed (i.i.d.) random variables, so that the cycle count of $\boldsymbol{y}$ is a stochastic variable. Let T_k denote the largest cycle count value of k random samples drawn from a population with distribution $f(i)$. The distribution of T_k, denoted by $g_k(T_k = i)$, is

$$g_k(i) = F(i)^k - F(i-1)^k, \ n \leq i \leq qn. \tag{19}$$

Results of computations of $g_k(i)$ versus *i/n* are shown in Figure 4 for $n = 64$, $q = 4$, the number of parallel strands is $k = 1$, 2, 4, 8 and 16. We notice that the peak (modus) of the distribution shifts to larger values with increasing values of the number of parallel strands k. In the next section, we show that with a coding step of low redundancy, as small as one bit, we can shift the distribution to much lower values of the cycle count, and significantly reduce the probability of occurrence of a maximum (worst case) cycle count.

III. Coding Techniques for Efficiently Synthesizing DNA

In order to minimize the average, or maximum, number of synthesis cycles, we consider the usage of a constrained code for translating the source sequence into a more suitable version having a bearing on the average or maximum number of required synthesis cycles.

For limiting the maximum number of cycles, an encoder translates source data into a low-weight codeword $\boldsymbol{y}$ with limited word sum and transforms it, using the precoding operation (3), into the sequence $\boldsymbol{x}$ to be translated into a DNA strand. Schalkwijk and Cover presented enumerative coding schemes for generating q-ary fixed-weight codes [16], [17]. Cover [17, Example 3], presented a binary scheme for enumerating codewords whose weight lies in a prescribed interval. His scheme can directly be applied to the situation at hand where codewords with limited word sum are generated. Recently more enumerative schemes (LOCO codes) have been presented by Hareedy and Calderbank [18].

Binary fixed-weight codes with equal numbers of 1's and 0's can be efficiently generated using Knuth's procedure [19], [20]. The cycle count of such sequences, after precoding (3), equals $\frac{3}{2}n$, while the redundancy is $\log_2(n)$. Efficient coding schemes for generating q-ary sequences of weight $n(q-1)/2$ have been presented in [21], [22]. The redundancy of their scheme is approximately $\log_q n$, $n \gg 1$. Binary coding schemes for generating constant weight sets are presented in [23].

The enumerative coding methods are efficient in terms of redundancy, but its practical application is often compounded by their complexity that does not scale linearly with word length n. The other schemes listed above, such as q-ary alternatives of Knuth's celebrated scheme, cannot easily be redesigned to generate sum constrained q-ary words.

In the next subsections, we take a look at alternative schemes that have small complexity and redundancy. The Polarity Invert (PI) scheme [10] requires a single bit for minimizing the cycle count. We introduce the PI scheme with multiple subwords, and show that the probability of occurrence of a maximum cycle count is much smaller than that of the basic PI scheme introduced in [10]. In guided scrambling [24] (GS), a set of pseudo-random representations of the source data is generated, and the encoder selects the representation with the least cycle count. The Nibble Replacement (NR) algorithm [25], [26] is a method for encoding/decoding low-weight codes with small complexity and low redundancy.

A. Polarity Invert (PI) Scheme

In [10], a useful technique was presented for minimizing the cycle count. We coin the name *Polarity Invert* (PI) for this scheme. Define the *inverse* of the symbol y_i by $\tilde{y}_i = q-1-y_i$, $1 \le i \le n$. We simply find, using (1), that

$$T_y + T_{\tilde{y}} = (q+1)n, \tag{20}$$

where T_y and $T_{\tilde{y}}$ denote the cycle count of y and $\tilde{y}$, respectively. Then, as a result, either T_y or $T_{\tilde{y}}$ is less or equal to $\mu_o(n)$. The encoder sends the vector requiring the least cycle count, which requires a redundancy of one bit to signal its choice.

The distribution of the cycle count is found by 'folding' the values of $f(w)$, $w > \mu_o(n)$, to $f((q+1)n - w)$. Let $\hat{f}(w)$ denote the cycle count distribution of the PI scheme, then we obtain for reasons of symmetry, $n \le i \le qn$,

$$\hat{f}(i) = \begin{cases} 2f(i) & i < \mu_o(n) \\ f(\mu_o(n)) & i = \mu_o(n) \\ 0 & i > \mu_o(n). \end{cases} \tag{21}$$

The average of the distribution $\hat{f}(i)$, denoted by $\mu_1(n) = \sum_i i\hat{f}(i)$, can, using the Gaussian distribution (14), be approximated by

$$\begin{aligned} \mu_1(n) &\approx \int_{x=-\infty}^{\mu_o(n)} x\varphi\Big(x; \mu_o(n), \sigma_o^2(n)\Big)\,dx \\ &\approx \mu_o(n) - \sigma_o(n)\sqrt{\frac{2}{\pi}}. \end{aligned} \tag{22}$$

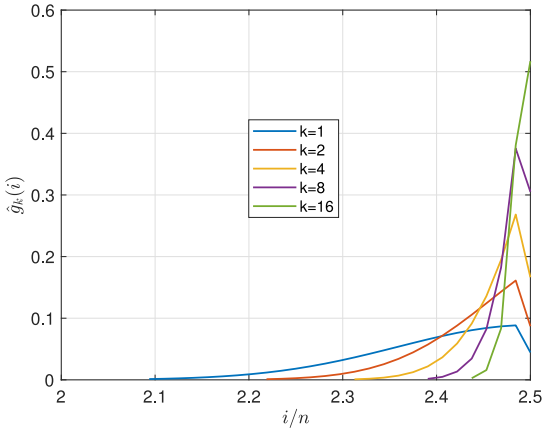


Fig. 5. Distribution, $\hat{g}_k(i)$, of the cycle count of the polarity invert scheme of multiple strand synthesis versus *i/n*, for strand length $n = 64$, $q = 4$, and number of parallel strands $k = 1$, 2, 4, 8, and 16.

In a similar vein we may compute the variance of the cycle count of the PI scheme, denoted by $\sigma_1^2(n)$,

$$\sigma_1^2(n) \approx \frac{\pi - 2}{\pi}\sigma_o^2(n). \tag{23}$$

The cumulative cycle count distribution of the PI scheme is denoted by $\hat{F}(i)$. The distribution of the cycle count of k multiple streams generated by the PI scheme, denoted by $\hat{g}_k(i)$, is

$$\hat{g}_k(i) = \hat{F}(i)^k - \hat{F}(i-1)^k, \ n \le i \le qn. \tag{24}$$

Results of computations of the distribution $\hat{g}_k(i)$ are shown in Figure 5 for word length $n = 64$, $q = 4$ and $k = 1$, 2, 4, 8, and 16 parallel streams. We notice that the cycle count distribution of PI-generated strands concentrates more and more just below the uncoded average, $\mu_o(n)$, with mounting number of parallel strands k.

B. PI Scheme With Multiple Subwords

We may trade redundancy versus cycle count by devising a code format where an n-symbol word is divided into ℓ m-symbol subwords. Clearly, $n = \ell m$. We apply the PI scheme to each m-symbol subword, so that the overall redundance of the n-symbol word equals ℓ bit. Define the discrete *convolution* of the distributions $f_x(i)$ and $f_y(i)$ by

$$f_x * f_y(j) \ = \sum_i f_x(i) f_y(j-i). \tag{25}$$

Let $\hat{f}_{[m]}(i)$ denote the cycle count distribution of the m-symbol PI scheme (the subscript $[m]$ was added to distinguish it from the distribution $\hat{f}(i)$ of an n symbol word). The cycle count distribution of n-symbol words composed of m-symbol subwords, denoted by $\tilde{h}(i)$, is the ℓ-fold convolution of $\hat{f}_{[m]}(i)$, or

$$\tilde{h}(i) = \hat{f}_{[m]} * \hat{f}_{[m]} * \cdots * \hat{f}_{[m]}(i). \tag{26}$$

Figure 6 shows the distribution of the cycle count, $\tilde{h}(i)$, of multiple subword PI encoded single strand synthesis versus *i/n*, for $n = 64$, $q = 4$ and subword length $m = 64$, 32, 16

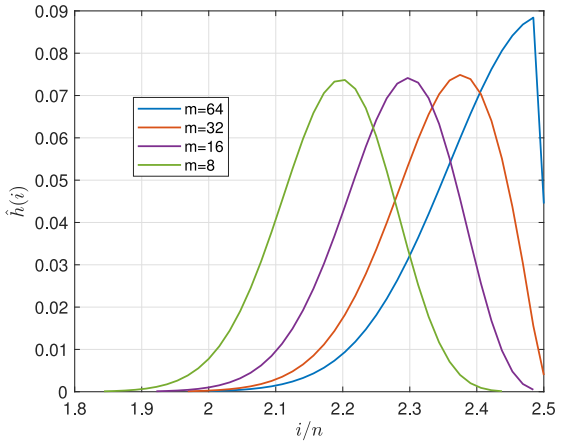


Fig. 6. Distribution of the cycle count, $\tilde{h}(i)$, of multiple subword PI of single strand synthesis versus *i/n*, for $n = 64$, $q = 4$ and subword length $m = 64$, 32, 16 and 8.

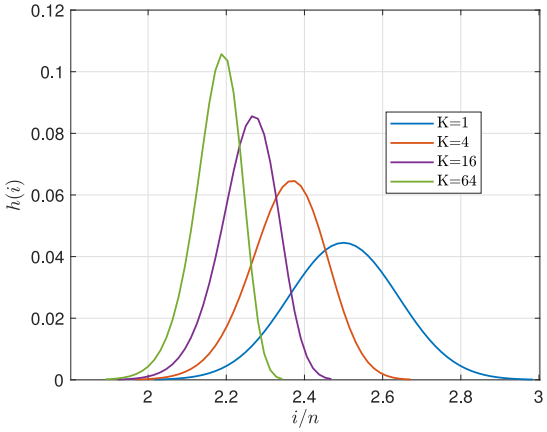


Fig. 7. Distribution of the cycle count, *h(i)*, of GS of single strand synthesis for $n = 64$, $q = 4$, using a selection set of $K = 1$, 4, 1 and 64 pseudo-random words.

and 8. The code redundancy is 1, 2, 4, and 8 bits, respectively. Note that the maximum cycle count of the multiple *m*-symbol subword scheme is the same as that of the original scheme, namely $(q + 1)n/2$. The probability of occurrence of a maximum cycle count is much smaller than that of the original scheme.

The average and variance of the distribution $\tilde{h}(i)$, denoted by $\mu_\ell(n)$ and $\sigma^2_\ell(n)$, respectively, can be approximated by

$$\mu_\ell(n) = \sum_i i\tilde{h}(i) = \ell \sum_{i \le \mu_o(m)} i\hat{f}_{[m]}(i)$$
$$\approx \mu_o(n) - \sigma_o(n)\sqrt{\frac{2}{\pi}\ell} \tag{27}$$

and

$$\sigma^2_\ell(n) \approx \frac{\pi - 2}{\pi}\sigma^2_o(n), \tag{28}$$

which is independent of the number of subwords ℓ. In the next subsection, we analyze the performance of guided scrambling.

C. Guided Scrambling (GS)

Guided scrambling is an efficient high-rate code, where the encoder sends the most suitable word taken from a selection set of *K* pseudo-random representations of the source word [24]. In a typical embodiment of GS, a representation is generated by adding mod q a pseudo-random sequence to the source data, where the pseudo-random sequence is taken from the set of *K* predefined sequences, known to both sender and receiver. The selected random word added to the source word is identified by appending a tag to the sent word, so that the receiver can uniquely undo the 'randomization'. Clearly, $K \le q^p$, where p is the number of redundant symbols of the identification tag. We concentrate here on the generation of codewords with a small cycle count; it should be noted that GS is a versatile technique so that other constraints such as GC-balance or homopolymer-run restrictions [28] can easily be embedded in the selection criteria.

The distribution of the cycle count of GS of a single strand, where a selection set of *K* pseudo-randomly generated words is used, denoted by *h*(*i*), is

$$h(i) = (1 - F(i-1))^K - (1 - F(i))^K,\ n \le i \le qn. \tag{29}$$

Examples of distributions are shown in Figure 7 for $n = 64$, $q = 4$, using a selection set of $K = 1$, 4, 16 and 64 pseudo-random words.

The encoder is not able to guarantee that it can generate a codeword in *S*(*t*), that is, with a cycle count $T \le t$, $n \le t \le qn$. The probability of *encoder failure*, defined as the probability that the encoder does not produce an allowed codeword in *S*(*t*), is denoted by $P_r(t)$, and given by

$$P_r(t) = \sum_{i=t+1}^{qn} h(i) = (1 - F(t))^K. \tag{30}$$

In case we design an encoder with an encoder failure rate $P_r(t) = \epsilon$, $\epsilon \ll 1$, we infer that $K \ge K_\epsilon$, where

$$K_\epsilon = \frac{\log_2 \epsilon}{\log_2(1 - F(t))}. \tag{31}$$

The redundancy of the GS method at an encoder failure rate ϵ, denoted by $D_\epsilon(t)$, is defined by

$$D_\epsilon(t) = \log_2 K_\epsilon. \tag{32}$$

The (relative) redundancy is plotted in Figure 8 for $\epsilon = 10^{-4}$, $q = 4$, and selected values of *n*.

1) Redundancy Estimate: Using the Gaussian approximation (15), we may write down a Taylor series of (31) and (32) at $t = \mu_o(n)$, and obtain

$$D_\epsilon(t) \approx c_3 + c_4(t - \mu_o(n)), \tag{33}$$

where

$$c_3 = \log_2(-\log_2(\epsilon)) \text{ and } c_4 = -\frac{1}{\ln(2)^2}\sqrt{\frac{2}{\pi\sigma^2}}.$$

For $\epsilon = 10^{-4}$ and $q = 4$, we have $c_3 \approx 3.732$ and $c_4 \approx -1.485/\sqrt{n}$. Note that the coefficients of the linear terms in (33) and (18) differ by a factor of $1/\ln(2) \approx 1.443$.

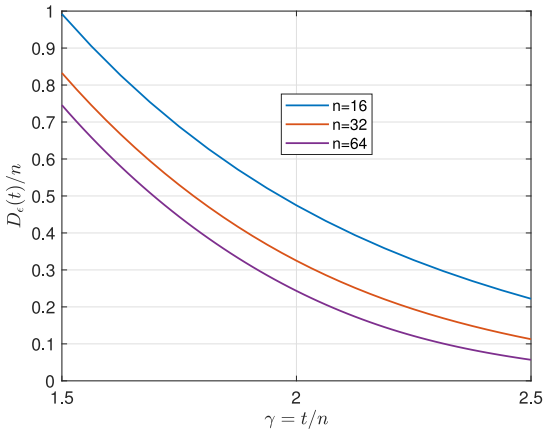


Fig. 8. Relative redundancy, $D_\epsilon(t)/n$, of guided scrambling versus $\gamma = t/n$ for $\epsilon = 10^{-4}$ and word length $n = 16$, 32, and 64, $q = 4$.

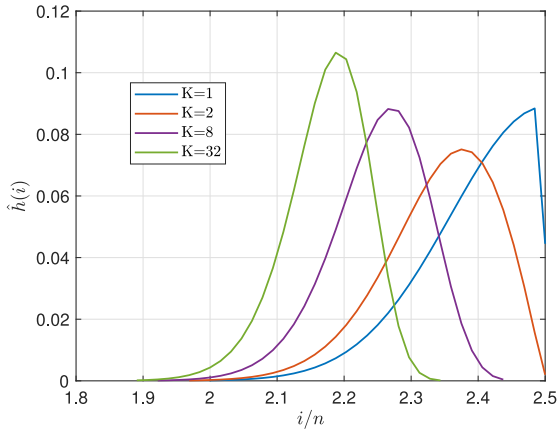


Fig. 9. Distribution of the cycle count, $\hat{h}(i)$, of combined GS and PI of single strand synthesis versus i/n, for $n = 64$, $q = 4$ and a set of $K = 1$, 2, 8 and 32 pseudo-random words. The case $K = 1$ refers to the baseline PI code presented in [10].

D. Combined GS and PI Scheme

We may readily combine the GS and the PI scheme. Again, as above, we define a selection set, $\{y_1, y_2, \ldots, y_K\}$, of K pseudo-random words. An output word with $T \leq (q+1)n/2$ is guaranteed when there is at least one word y_i and its inverse $\tilde{y}_i$ in the selection set. We opt for a slightly different code format that is amenable for analysis of its cycle count distribution. We assume a format where the selection set is $\{y_1, y_2, \ldots, y_K, \tilde{y}_1, \tilde{y}_2, \ldots, \tilde{y}_K\}$. Then, the redundancy of the combined scheme, which is denoted by $\hat{D}_\epsilon(t)$, equals

$$\hat{D}_\epsilon(t) = 1 + \log_2 \frac{\log_2 \epsilon}{\log_2\Big(1 - \hat{F}(t)\Big)}. \tag{34}$$

The cycle count distribution of the combined GS/PI scheme is

$$\hat{h}(i) = \Big(1 - \hat{F}(i-1)\Big)^K - \Big(1 - \hat{F}(i)\Big)^K, \ n \leq i \leq qn. \tag{35}$$

Figure 9 displays examples of distributions of the combined GS/PI scheme for the same parameters as in Figure 7. We have assumed the same code redundancy in the computations of Figures 7 and 9, so that for $K > 1$ the number of pseudo-random evaluations in GS/PI is halve that of the regular GS scheme. The case $K = 1$ refers to the baseline PI code presented in [10] having one bit redundancy. If K is relatively small, we notice a significant difference in performance in Figures 7 and 9, but the difference is diminishing for larger values of K.

E. Nibble Replacement (NR) Algorithm

The generation of low-weight codes that limit the maximum allowed cycle count of an n-symbol q-ary word to t is straightforwardly accomplished with a look-up table that translates the (binary) source data into low-weight n-symbol q-ary words. The decoder requires a look-up table for reversing the encoder operation. The practical difficulty is that for, say, $n > 30$, the required look-up tables for encoding and decoding are much too large.

The nibble replacement (NR) algorithm [25], [27] is an alternative method for encoding/decoding with small complexity and redundancy. As in the PI scheme with multiple subwords, in the NR format, an n-symbol strand is divided into L subwords of length m, so that $n = Lm$. Let t_m be the maximum allowed cycle count of an m-symbol q-ary word, then the overall cycle count of the n-symbol q-ary word is upperbounded by $t = Lt_m$. The number of low-weight m-symbol codewords, denoted by M_{t_m}, equals, see (9),

$$M_{t_m} = F(t_m) q^m. \tag{36}$$

It would be a pleasant coincidence if M_{t_m} is a power of two so that encoding and decoding is a simple and efficient operation. If M_{t_m} is not a power of two, we can translate, using a look-up table, at most $m_l = \lfloor \log_2 M_{t_m} \rfloor$ source bits into an m-symbol q-ary word, and $M_{t_m} - 2^{m_l}$ available words are discarded. This truncation to a power of two can seriously degrade the code efficiency. In general, the NR algorithm may significantly improve the redundancy with respect to the simple look-up table-based method.

Define the integer $m_h = \lceil \log_2 M_{t_m} \rceil$ and let

$$L = \left\lfloor \frac{2^{m_h} - 1}{2^{m_h} - M_{t_m}} \right\rfloor. \tag{37}$$

The NR algorithm translates $Lm_h - 1$ source bits into L m_h-bit words. Each m_h-bit word is translated, using a look-up table, into a q-ary m-symbol word that satisfies the prescribed t_m-cycle count constraint. For details of the NR method we refer to [25]. The NR encoding method requires data storage of L m_h-bit words, the execution of the encoding algorithm, and a look-up table for translating an m_h-bit wide word into a word of m q-ary symbols, so that very large, n-symbol wide, look-up tables are avoided. The overall redundancy of the encoded n-symbol word is

$$r(t) = L(m \log_2 q - m_h) + 1 \tag{38}$$

bit. The information rate $W(t)$ is

$$W(t) = \frac{Lm_h - 1}{t}. \tag{39}$$

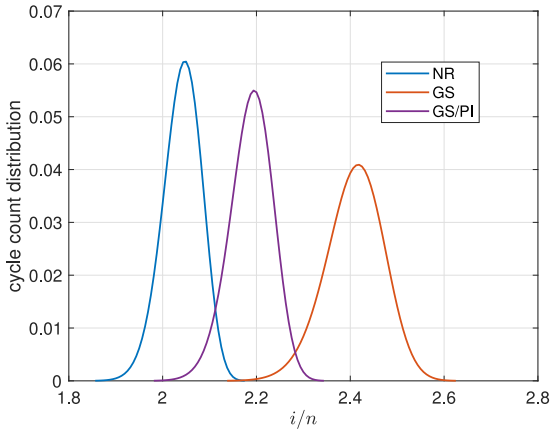


Fig. 10. Cycle count distribution of $n = 16 \times 10 = 160$ 4-ary symbols, $t_m = 22$ and $t = 352$, generated by the NR algorithm assuming i.i.d. source symbols, see Example 1, with a 33 bit overall redundancy. As a comparison, the cycle count distributions of a 16×10 4-ary symbol word using GS and for combined PI/GS both with 32 bit overall redundancy are shown.

TABLE I
RESULTS OF THE NIBBLE REPLACEMENT CODING METHOD FOR SELECTED VALUES OF SUBWORD LENGTH m AND MAXIMUM SUBWORD CYCLE COUNT, t_m, $q = 4$

m	t_m	m_h	L	$r(t)$	$W(t)$
8	17	14	3	7	0.8039
10	22	18	16	33	0.8153
10	20	17	2	7	0.8250
10	19	16	12	49	0.8377
12	25	21	56	169	0.8393
12	22	19	2	11	0.8409
12	21	18	3	19	0.8413
12	20	17	2	15	0.8250
14	31	26	2	5	0.8226
14	29	25	2	7	0.8448
14	28	24	13	53	0.8544

The next numerical example may illustrate the above.

Example 1: Let $q = 4$, $m = 10$, and $t_m = 22$, then, using the generating function (7), we obtain $M_{t_m} = 253.991$. Hence, we have $m_h = 18$ and $L = 16$, so that $n = Lm = 160$ and $t = Lt_m = 352$. Then, using (38) and (39), we obtain $r(t) = 33$ and $W(t) = 287/352 = 0.8153$. Figure 10 shows the distribution of the cycle count of the $n = 160$ 4-ary word for i.i.d. source symbols encoded by the NR algorithm with an overall redundancy of 33 bit. As a comparison we plotted the cycle count distributions of an $n = 160$ 4-ary symbol word divided into 16 subwords using GS or combined PI/GS with a 32-bit overall redundancy. ■

Table I shows more results for $q = 4$ for selected values of m and t_m.

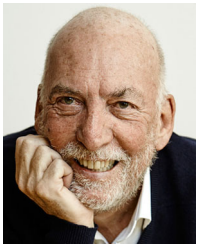
IV. CONCLUSION

We have reported on codes for efficiently synthesizing DNA for data storage, where a plurality of DNA strands are synthesized in parallel. We have analyzed the trade-off between the information contents, redundancy, and average or maximum number of cycles required for synthesizing a plurality of DNA strands in parallel. Assuming random source words, we have computed the cycle number distributions of various code constructions. We have analyzed the performance of polarity invert (PI) codes with and without multiple subwords, guided scrambling (GS), combined GS/PI codes, and constrained codes based on the nibble replacement (NR) algorithm.

REFERENCES

[1] G. M. Church, Y. Gao, and S. Kosuri, "Next-generation digital information storage in DNA," *Science*, vol. 337, no. 6012, p. 1628, 2012, doi: 10.1126/science.1226355.
[2] W. Song, K. Cai, and K. A. S. Immink, "Sequence-subset distance and coding for error control in DNA-based data storage," *IEEE Trans. Inf. Theory*, vol. 66, no. 10, pp. 6048–6065, Oct. 2020, doi: 10.1109/TIT.2020.3002611.
[3] A. Lenz, P. H. Siegel, A. Wachter-Zeh, and E. Yaakobi, "Coding over sets for DNA storage," *IEEE Trans. Inf. Theory*, vol. 66, no. 4, pp. 2331–2351, Apr. 2020, doi: 10.1109/TIT.2019.2961265.
[4] 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 Trans. Inf. Theory*, vol. 67, no. 8, pp. 5602–5613, Aug. 2021, doi: 10.1109/TIT.2021.3066430.
[5] M. Blawat et al., "Forward error correction for DNA data storage," in *Proc. Int. Conf. Comput. Sci. (ICCS)*, vol. 80, 2016, pp. 1011–1022, doi: 10.1016/j.procs.2016.05.398.
[6] K. G. Benerjee and A. Banerjee, "On DNA codes with multiple constraints," *IEEE Commun. Lett.*, vol. 25, no. 2, pp. 365–368, Feb. 2021, doi: 10.1109/LCOMM.2020.3029071.
[7] O. Milenkovic and N. Kashyap, "DNA codes that avoid secondary structures," in *Proc. Int. Symp. Inf. Theory*, 2005, pp. 288–292, doi: 10.1109/ISIT.2005.1523340.
[8] T. T. Nguyen, K. Cai, H. M. Kiah, D. Tu Dao, and K. A. S. Immink, "On the design of codes for DNA computing: Secondary structure avoidance codes," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Taipei, Taiwan, 2023, pp. 573–578, doi: 10.1109/ISIT54713.2023.10206972.
[9] K. G. Benerjee and A. Banerjee, "On homopolymers and secondary structures avoiding, reversible, reversible-complement and GC-balanced DNA codes," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Espoo, Finland, 2022, pp. 204–209, doi: 10.1109/ISIT50566.2022.9834744.
[10] A. Lenz, Y. Liu, C. Rashtchian, P. H. Siegel, A. Wachter-Zeh, and E. Yaakobi, "Coding for efficient DNA synthesis," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Los Angeles, CA, USA, 2020, pp. 2885–2890, doi: 10.1109/ISIT44484.2020.9174272.
[11] K. Makarychev, M. Z. Racz, C. Rashtchian, and S. Yekhanin, "Batch optimization for DNA synthesis," *IEEE Trans. Inf. Theory*, vol. 68, no. 11, pp. 7454–7470, Nov. 2022, doi: 10.1109/TIT.2022.3184903.
[12] M. Abu-Sini, A. Lenz, and E. Yaakobi, "DNA synthesis using shortmers," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Taipei, Taiwan, 2023, pp. 585–590, doi: 10.1109/ISIT54713.2023.10206609.
[13] O. Elishco and W. Huleihel, "Optimal reference for DNA synthesis," *IEEE Trans. Inf. Theory*, vol. 69, no. 11, pp. 6941–6955, Nov. 2023, doi: 10.1109/TIT.2023.3286694.
[14] J. Chrisnata, H. M. Kiah, and V. L. P. Pham, "Deletion correcting codes for efficient DNA synthesis," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Taipei, Taiwan, 2023, pp. 352–357, doi: 10.1109/ISIT54713.2023.10206892.
[15] K. A. S. Immink, "Innovation in constrained codes," *IEEE Commun. Mag.*, vol. 60, no. 10, pp. 20–24, Oct. 2022, doi: 10.1109/MCOM.002.2200249.
[16] J. P. M. Schalkwijk, "An algorithm for source coding," *IEEE Trans. Inf. Theory*, vol. IT-18, no. 3, pp. 395–399, May 1972, doi: 10.1109/TIT.1972.1054832.
[17] T. M. Cover, "Enumerative source coding," *IEEE Trans. Inf. Theory*, vol. IT-19, no. 1, pp. 73–77, Jan. 1973, doi: 10.1109/TIT.1973.1054929.
[18] A. Hareedy and R. Calderbank, "LOCO codes: Lexicographically-ordered constrained codes," *IEEE Trans. Inf. Theory*, vol. 66, no. 6, pp. 3572–3589, Jun. 2020, doi: 10.1109/TIT.2019.2943244.
[19] D. E. Knuth, "Efficient balanced codes," *IEEE Trans. Inf. Theory*, vol. IT-32, no. 1, pp. 51–53, Jan. 1986, doi: 10.1109/TIT.1986.1057136.
[20] J. H. Weber and K. A. S. Immink, "Knuth's balanced codes revisited," *IEEE Trans. Inf. Theory*, vol. IT-56, no. 4, pp. 1673–1679, Apr. 2010, doi: 10.1109/TIT.2010.2040868.
[21] T. G. Swart and J. H. Weber, "Efficient balancing of q-ary sequences with parallel decoding," in *Proc. IEEE Int. Symp. Inf. Theory*, 2009, pp. 1564–1568, doi: 10.1109/ISIT.2009.5205824.

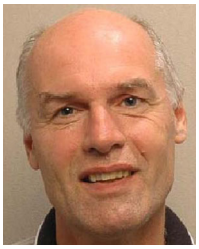
[22] R. M. Capocelli, L. Gargano, and U. Vaccaro, "Efficient q-ary immutable codes," *Discr. Appl. Math.*, vol. 33, pp. 25–41, Nov. 1991, doi: 10.1016/0166-218X(91)90106-7.
[23] V. Skachek and K. A. S. Immink, "Constant weight codes: An approach based on Knuth's balancing method," *IEEE J. Sel. Areas Commun.*, vol. 32, no. 5, pp. 908–918, May 2014.
[24] I. J. Fair, W. D. Gover, W. A. Krzymien, and R. I. MacDonald, "Guided scrambling: A new line coding technique for high bit rate fiber optic transmission systems," *IEEE Trans. Commun.*, vol. COM-39, no. 2, pp. 289–297, Feb. 1991, doi: 10.1109/26.76466.
[25] K. A. S. Immink and K. Cai, "Efficient encoding of constrained block codes," *IEEE Commun. Lett.*, vol. 25, no. 11, pp. 3468–3472, Nov. 2021, doi: 10.1109/LCOMM.2021.3105327.
[26] J. H. Weber, "Asymptotic results on codes for symmetric, unidirectional, and asymmetric error control," *IEEE Trans. Inf. Theory*, vol. 40, no. 6, pp. 2073–2075, Nov. 1994, doi: 10.1109/18.340484.
[27] A. J. van Wijngaarden and K. A. S. Immink, "Combinatorial construction of high rate runlength-limited codes," in *Proc. GLOBECOM*, vol. 1, 1996, pp. 343–347, doi: 10.1109/GLOCOM.1996.594386.
[28] 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, doi: 10.1109/LCOMM.2019.2930970.



Kees A. Schouhamer Immink (Life Fellow, IEEE) founded Turing Machines Inc., an innovative start-up focused on novel signal processing for DNA-based storage, where he is currently the President. Among the accolades received are the Personal Emmy Award in 2004, the 2017 IEEE Medal of Honor, the 1999 AES Gold Medal, the IEEE Masaru Ibuka Consumer Electronics Award, the 2004 SMPTE Progress Medal, and the 2015 IET Faraday Medal. He was inducted into the Consumer Electronics Hall of Fame and the U.S. National Academy of Engineering.

Kui Cai (Senior Member, IEEE) received the B.E. degree in information and control engineering from Shanghai Jiao Tong University, Shanghai, China, and the joint Ph.D. degree in electrical engineering from the Technical University of Eindhoven, The Netherlands, and the National University of Singapore. She is currently an Associate Professor with the Singapore University of Technology and Design. Her main research interests are in the areas of coding theory, information theory, and signal processing for emerging data storage systems and computing. She received the 2008 IEEE Communications Society Best Paper Award in Coding and Signal Processing for Data Storage. She served as the Vice-Chair (Academia) of IEEE Communications Society, Data Storage. She was listed in the 2020 Who's Who in Engineering Singapore.

Tuan Thanh Nguyen (Member, IEEE) received the B.Sc. and Ph.D. degrees in mathematics from Nanyang Technological University, Singapore, in 2014 and 2019, respectively. He is currently a Research Fellow with the Advanced Coding and Signal Processing Laboratory, Singapore University of Technology and Design. He was a Research Fellow with the School of Physical and Mathematical Sciences, Nanyang Technological University, Singapore, from August 2018 to September 2019. His research interests lie in the interplay between applied mathematics and computer science/engineering, particularly, including combinatorics and coding theory. His current research project concentrates on error correction codes and constrained codes for DNA-based data storage.



Jos H. Weber (Senior Member, IEEE) was born in Schiedam, The Netherlands, in 1961. He received the M.Sc. degree in mathematics (with Hons.) and the Ph.D. and Master of Business Telecommunications degrees from the Delft University of Technology, Delft, The Netherlands, in 1985, 1989, and 1996, respectively. Since 1985, he has been with the Delft University of Technology. He is currently an Associate Professor with the Department of Applied Mathematics. He was a Visiting Researcher with the University of California at Davis, Davis, CA, USA, the Tokyo Institute of Technology, Japan, the University of Johannesburg, South Africa, EPFL, Switzerland, and SUTD, Singapore. His main research interests are in the area of channel coding. He has been an Honorary Member of the Werkgemeenschap voor Informatie- en Communicatietheorie, and the Secretary of the IEEE Benelux Chapter on Information Theory since 2008.