On Single-Error-Detecting Codes for DNA-Based Data Storage

Weber, Jos H.; De Groot, Joost A.M.; Van Leeuwen, Charlot J.

DOI
10.1109/LCOMM.2020.3023826

Publication date
2021

Document Version
Accepted author manuscript

Published in
IEEE Communications Letters

On Single-Error-Detecting Codes for DNA-Based Data Storage

Jos H. Weber, *Senior Member, IEEE*, Joost A.M. de Groot, and Charlot J. van Leeuwen

***Abstract*—DNA-based storage is considered to be a promising option to accommodate huge amounts of data. The strings of nucleotides are prone to errors though. To reduce the error probability, these strings should satisfy constraints on the ratio of A's and T's versus the number of G's and C's, and on the maximum number of repeated identical nucleotides. To deal with errors when they occur after all, it is also desirable that the set of DNA-strings possesses certain error correction or detection capabilities. This is established by designing quaternary constrained codes with a specified minimum distance. Here, maximum-sized block codes with a fixed number of G/C symbols, no symbol repetition, and a minimum Hamming distance of two are presented.**

Index Terms: Constant-weight codes, DNA-based data storage, error-detecting codes, runlength-limited sequences.

I. Introduction

Deoxyribonucleic acid (DNA) has been demonstrated to be a promising medium for massive digital data storage [1], as a possible alternative for magnetic and optical discs. An overview of trends and methods in DNA-based storage has been provided in [2]. For robustness purposes, the strings consisting of the nucleotides adenine (A), thymine (T), guanine (G), and cytosine (C) should satisfy some constraints. For example, the number of G/C nucleotides in the string, called the GC-weight, should be (about) the same as the number of A/T's. Furthermore, the number of subsequent identical nucleotides in a string should not be too long, which can be established by imposing a runlength constraint. The GC-weight and runlength constraints lead to a restricted set of quaternary sequences that can be used for representing the digital data. By carefully selecting a code, that is a subset of this set, with a certain minimum (Hamming) distance, we can enforce some error correction or detection capabilities as well [3].

King derived bounds on the sizes of quaternary codes with fixed length, GC-weight, and minimum Hamming distance [4]. Immink and Cai focused on the runlength constraint in DNA codes [5], and in subsequent studies they involved the GC-weight as well [6], [7]. Limbachiya et al. [8] and Cao et al. [9] derived lower bounds on the sizes of optimal codes with a fixed GC-weight and a specified minimum Hamming distance, under the strong runlength restriction that identical nucleotides are not allowed to occur next to each other. They call the latter the no-runlength constraint. It reduces the error probability when retrieving the stored data, but it may also reduce the code rate.

In this paper, we are interested in finding the largest codes among the ones meeting given specifications. We present a recursive formula to determine the size of the set of quaternary words with any fixed length, GC-weight, and runlength constraint. A technique based on generating functions to calculate such quantities was already presented in [6], but our simple recursive expression has the advantage that it can also be easily evaluated for large lengths. Our major result is that for the specific case of imposing the no-runlength constraint, as considered in [8], [9], we determine the maximum size of any code within the mentioned set having minimum Hamming distance two, i.e., optimal single-error-detecting codes. This settles an open problem from [8] and comes with an explicit construction for optimal codes.

The rest of the paper is organized as follows. In Section II, notation and basic definitions are provided. Then, our results on the sets and codes under consideration are presented and proved in Sections III and IV. Finally, concluding remarks are given in Section V.

II. Preliminaries

For convenience, we will represent the nucleotides by numerical symbols rather than letters in the rest of this paper, using the mapping

$$A \leftrightarrow 0, T \leftrightarrow 1, G \leftrightarrow 2, C \leftrightarrow 3. \tag{1}$$

We consider words of length n over the $\{0,1,2,3\}$ alphabet. The full set of size 4^n of such words is denoted as $\mathcal{B}(n)$, i.e.,

$$\mathcal{B}(n) = \{0,1,2,3\}^n. \tag{2}$$

With every $\mathbf{x} = (x_1, \ldots, x_n) \in \mathcal{B}(n)$ we associate the following two words of length n. The first one is the low/high word $\mathbf{x}^{\mathrm{LH}} = (x_1^{\mathrm{LH}}, \ldots, x_n^{\mathrm{LH}})$ with

$$x_i^{\mathrm{LH}} = \begin{cases} \mathrm{L} & \text{if } x_i = 0,1, \\ \mathrm{H} & \text{if } x_i = 2,3, \end{cases} \tag{3}$$

for all i. It indicates whether a symbol x_i is in the lower category $\{0,1\}$ or in the higher category $\{2,3\}$. The second one is the cluster word $\mathbf{x}^{\mathrm{C}} = (x_1^{\mathrm{C}}, \ldots, x_n^{\mathrm{C}})$ with

$$x_i^{\mathrm{C}} = \begin{cases} \mathrm{L} & \text{if } x_i^{\mathrm{LH}} = \mathrm{L} \text{ and } x_j^{\mathrm{LH}} = \mathrm{H} \\ & \qquad \forall j : 1 \le j \le n \wedge |j-i| = 1, \\ \mathrm{H} & \text{if } x_i^{\mathrm{LH}} = \mathrm{H} \text{ and } x_j^{\mathrm{LH}} = \mathrm{L} \\ & \qquad \forall j : 1 \le j \le n \wedge |j-i| = 1, \\ x_i & \text{otherwise}, \end{cases} \tag{4}$$

for all i. It will be used in Section IV for clustering purposes.

Manuscript accepted for publication in IEEE Communications Letters, Sept. 9, 2020. The authors are with Delft University of Technology, Dept. of Applied Mathematics, Delft, The Netherlands, j.h.weber@tudelft.nl, j.a.m.degroot@tudelft.nl, lot.leeuwen@gmail.com.

We define the GC-weight $w(\mathbf{x})$ of $\mathbf{x}$ as the number of symbols in $\mathbf{x}$ that are equal to 2 or 3, i.e.,

$$w(\mathbf{x}) = |\{i : x_i^{\mathrm{LH}} = H\}|. \tag{5}$$

Further, the (maximum) runlength $r(\mathbf{x})$ of $\mathbf{x}$ is the maximum number of subsequent identical symbols in $\mathbf{x}$, i.e.,

$$r(\mathbf{x}) = \max\{r : \exists i \text{ such that } x_i = x_{i+1} = \cdots = x_{i+r-1}\}. \tag{6}$$

The index $i(\mathbf{x})$ of $\mathbf{x}$ is the number of symbols in $\mathbf{x}^C$ that are valued L or H, i.e.,

$$i(\mathbf{x}) = |\{i : x_i^{\mathrm{C}} \in \{\mathrm{L}, \mathrm{H}\}\}|. \tag{7}$$

For example, if $\mathbf{x} = (0, 1, 3, 3, 1, 2, 2, 0, 2)$, also shortly denoted as 013312202, then $\mathbf{x}^{\mathrm{LH}}$ = LLHHLHHLH, $\mathbf{x}^{\mathrm{C}}$ = 0133L22LH, $w(\mathbf{x}) = 5$, $r(\mathbf{x}) = 2$, and $i(\mathbf{x}) = 3$.

In order to reduce the error probability, a well-chosen subset of $\mathcal{B}(n)$ should be used for data storage purposes rather than the entire set $\mathcal{B}(n)$ itself. Usually, constraints are put on the GC-weights and the maximum runlengths of the words. Therefore, we consider the subset $\mathcal{B}_r(n, w)$ that contains all the words from $\mathcal{B}(n)$ that have GC-weight w and runlength at most r, i.e.,

$$\mathcal{B}_r(n, w) = \{\mathbf{x} \in \mathcal{B}(n) : w(\mathbf{x}) = w \wedge r(\mathbf{x}) \leq r\}. \tag{8}$$

Its cardinality is denoted by $B_r(n, w)$. In Section III we will present a formula for $B_r(n, w)$.

It is often desirable to equip the set of words that is used for the data representation with some error correcting or detecting capabilities. In order to establish this, one could select a code $\mathcal{C}$, i.e., a subset of $B_r(n, w)$, satisfying a certain distance property. The Hamming distance between two sequences $\mathbf{x}$ and $\mathbf{y}$ from $\mathcal{B}_n$ is defined by $d(\mathbf{x}, \mathbf{y}) = |\{i : x_i \neq y_i\}|$. The (minimum) Hamming distance $d(\mathcal{C})$ of a code $\mathcal{C}$ is defined as the smallest Hamming distance between any two different codewords, i.e., $d(\mathcal{C}) = \min\{d(\mathbf{x}, \mathbf{y}) : \mathbf{x}, \mathbf{y} \in \mathcal{C}, \mathbf{x} \neq \mathbf{y}\}$. A code with Hamming distance d is known to correct up to $\lfloor (d-1)/2 \rfloor$ substitution errors. Alternatively, it could also be used to detect up to $d-1$ substitution errors. An important research challenge is to determine the largest possible code in $\mathcal{B}_r(n, w)$ with Hamming distance at least equal to d. Its size will be denoted by $B_r(n, w, d)$, i.e.,

$$B_r(n, w, d) = \max\{|\mathcal{C}| : \mathcal{C} \subseteq \mathcal{B}_r(n, w), d(\mathcal{C}) \geq d\}. \tag{9}$$

In Section IV we will determine $B_1(n, w, 2)$, i.e., the largest possible size of a single-error-detecting code of length n, in which all codewords have GC-weight w and no subsequent identical symbols.

III. A FORMULA FOR $B_r(n, w)$

In this section we will focus on $B_r(n, w)$, i.e., the number of words in the set $\mathcal{B}_r(n, w)$. In [8], an explicit formula for $B_1(n, w)$ is provided. In [6], it is shown how $B_r(n, w)$ can be obtained using generating functions. Here, we give a simple recursion to find $B_r(n, w)$. In order to do so, we define $\mathcal{N}_r(n, w)$ as the set of all words in $\mathcal{B}_r(n, w)$ that do not end with a zero, i.e.,

$$\mathcal{N}_r(n, w) = \{\mathbf{x} \in \mathcal{B}_r(n, w) : x_n \neq 0\}. \tag{10}$$

Its cardinality is denoted by $N_r(n, w)$. By symmetry arguments, we have

$$|\{\mathbf{x} \in \mathcal{B}_r(n, w) : x_n \neq i\}| = \begin{cases} N_r(n, w) & \text{if } i = 0, 1, \\ N_r(n, n-w) & \text{if } i = 2, 3. \end{cases} \tag{11}$$

Numerical values of $N_r(n, w)$ and $B_r(n, w)$ can be found as follows.

Theorem 1. *For $0 \leq w \leq n$ and $r \geq 1$, it holds that $N_r(0, 0) = 1$,*

$$N_r(n, w) = 2^{n-1}\binom{n-1}{w} + 2^n\binom{n-1}{w-1} \tag{12}$$

if $1 \leq n \leq r$,

$$\begin{aligned} N_r(n, w) &= \sum_{j=1}^{\min\{r, n-w\}} N_r(n-j, w) \\ &\quad + 2 \sum_{j=1}^{\min\{r, w\}} N_r(n-j, n-w) \end{aligned} \tag{13}$$

if $n \geq r+1$, and

$$B_r(n, w) = \sum_{j=0}^{\min\{r, n-w\}} N_r(n-j, w). \tag{14}$$

Proof: The result $N_r(0, 0) = 1$ follows from the observation that $\mathcal{B}(0)$ contains only the empty word, that has length and GC-weight both equal to zero, does not contain symbol runs longer than r, does not end with a zero, and thus is in $\mathcal{N}_r(0, 0)$.

If $1 \leq n \leq r$, then any word in $\mathcal{B}(n)$ satisfies the runlength constraint, so we only need to count the number of words in $\mathcal{B}(n)$ that have GC-weight w and end with a symbol $i \in \{1, 2, 3\}$. If $i = 1$, then this number is $2^{n-1}\binom{n-1}{w}$, since there should be w symbols from $\{2, 3\}$ in the first $n-1$ positions, with the remaining symbols from $\{0, 1\}$. Similarly, this number is $2^{n-1}\binom{n-1}{w-1}$ if $i \in \{2, 3\}$. Hence, (12) follows by summation over $i = 1, 2, 3$.

Next, we consider the case $n \geq r+1$. Any word in $\mathcal{N}_r(n, w)$ can be uniquely decomposed into a word from $\mathcal{B}_r(n-j, v)$ not ending with an i, followed by a run of j equal symbols i where $i \in \{1, 2, 3\}$. If $i = 1$, then $v = w$ and $j \in \{1, 2, \ldots, \min\{r, n-w\}\}$, where $j \leq r$ is due to the runlength constraint, and $j \leq n-w$ is due to the fact that the GC-weight of a word in $\mathcal{B}_r(n-j, w)$ cannot exceed its length. Similarly, if $i \in \{2, 3\}$, then $v = w-j$ and $j \in \{1, 2, \ldots, \min\{r, w\}\}$. On the other hand, for all indicated values of i and j, appending a string of j symbols i to a word from $\mathcal{B}_r(n-j, v)$ not ending with an i leads to a unique word in $\mathcal{N}_r(n, w)$. Hence, by applying (11) and summation over all j for each i and then over $i = 1, 2, 3$, the number of sequences in $\mathcal{B}_r(n, w)$ not ending with a zero is as stated in (13).

Finally, we prove (14). Any word in $\mathcal{B}_r(n, w)$ can be uniquely decomposed into a word from $\mathcal{N}_r(n-j, w)$, followed by a string of j zeroes. Here, j is a nonnegative integer satisfying both $j \leq r$, due to the runlength constraint, and $j \leq n-w$, since the length $n-j$ of a word from $\mathcal{N}_r(n-j, w)$ is at least equal to its GC-weight w. On the other hand, appending j zeroes to any word from $\mathcal{N}_r(n-j, w)$, with

TABLE I
VALUES OF $B_2(n,w)$.

	$w=0$	$w=1$	$w=2$	$w=3$	$w=4$	$w=5$
$n=1$	2	2				
$n=2$	4	8	4			
$n=3$	6	24	24	6		
$n=4$	10	56	96	56	10	
$n=5$	16	120	296	296	120	16

j any value from $\{0,1,\ldots,\min\{r,n-w\}\}$, gives a unique word from $\mathcal{B}_r(n,w)$. The result thus follows by summation of $N_r(n-j,w)$ over all j. ∎

Table I gives example $B_r(n,w)$ values with $r=2$, $1\leq n\leq 5$, and $0\leq w\leq n$.

IV. A FORMULA FOR $B_1(n,w,2)$

Limbachiya et al. presented a general lower bound on $B_1(n,w,d)$ [8, Theorem 2]. In the same paper, they also obtained specific lower bounds by running an algorithm, giving better results. Here, we settle the problem for the case $d=2$, i.e., we determine $B_1(n,w,2)$.

We start by partitioning $\mathcal{B}_1(n,w)$ into disjoint clusters. Two of its words $\mathbf{x}$ and $\mathbf{y}$ are in the same cluster if and only if their corresponding cluster words $\mathbf{x}^{\mathrm{C}}$ and $\mathbf{y}^{\mathrm{C}}$, as defined in (4), are equal. We show that words in different clusters cannot have Hamming distance one.

Lemma 1. *Let n and w be integers satisfying $0\leq w\leq n$ and $n\geq 2$. It holds for any $\mathbf{x},\mathbf{y}\in\mathcal{B}_1(n,w)$ with $\mathbf{x}^{\mathrm{C}}\neq\mathbf{y}^{\mathrm{C}}$ that $d(\mathbf{x},\mathbf{y})\geq 2$.*

Proof: Suppose there exist $\mathbf{x},\mathbf{y}\in\mathcal{B}_1(n,w)$ with $\mathbf{x}^{\mathrm{C}}\neq\mathbf{y}^{\mathrm{C}}$ and $d(\mathbf{x},\mathbf{y})=1$. Let i denote the position in which $\mathbf{x}$ and $\mathbf{y}$ differ. Note that $\mathbf{x}^{\mathrm{LH}}=\mathbf{y}^{\mathrm{LH}}$, since $x_j=y_j$ for all $j\neq i$ implies that $x_j^{\mathrm{LH}}=y_j^{\mathrm{LH}}$ for all $j\neq i$, but also that $x_i^{\mathrm{LH}}=y_i^{\mathrm{LH}}$ due to the fact that $w(\mathbf{x})=w(\mathbf{y})=w$.

Observe from (4) that $x_i^{\mathrm{C}}\in\{\mathrm{L},\mathrm{H}\}$ would imply that $y_j^{\mathrm{LH}}=x_j^{\mathrm{LH}}\neq x_i^{\mathrm{LH}}=y_i^{\mathrm{LH}}$ for all $j\in\{1,2,\ldots,n\}$ with $|j-i|=1$. This gives that $y_i^{\mathrm{C}}=x_i^{\mathrm{C}}$, and thus, in combination with the facts that $\mathbf{x}^{\mathrm{LH}}=\mathbf{y}^{\mathrm{LH}}$ and $x_j=y_j$ for all $j\neq i$, that $\mathbf{x}^{\mathrm{C}}=\mathbf{y}^{\mathrm{C}}$, which contradicts our assumption. Hence, $x_i^{\mathrm{C}}\notin\{\mathrm{L},\mathrm{H}\}$ and thus $x_i^{\mathrm{C}}\in\{0,1,2,3\}$. However, this implies that a) $x_{i-1}^{\mathrm{LH}}=x_i^{\mathrm{LH}}$ or b) $x_{i+1}^{\mathrm{LH}}=x_i^{\mathrm{LH}}$. If a) holds, then $y_{i-1}^{\mathrm{LH}}=x_{i-1}^{\mathrm{LH}}=x_i^{\mathrm{LH}}=y_i^{\mathrm{LH}}$ and $y_{i-1}=x_{i-1}\neq x_i$. Since $x_i\neq y_i$, this implies $y_{i-1}=y_i$, which violates the runlength constraint in $\mathbf{y}$. Similarly, we obtain a contradiction if b) holds.

In conclusion, if $\mathbf{x},\mathbf{y}\in\mathcal{B}_1(n,w)$ with $\mathbf{x}^{\mathrm{C}}\neq\mathbf{y}^{\mathrm{C}}$, then $d(\mathbf{x},\mathbf{y})\neq 1$. Since $\mathbf{x}^{\mathrm{C}}\neq\mathbf{y}^{\mathrm{C}}$ also implies $d(\mathbf{x},\mathbf{y})\neq 0$, of course, the result stated in the lemma follows. ∎

Next, we determine the sizes of the clusters.

Lemma 2. *Let n and w be integers satisfying $0\leq w\leq n$ and $n\geq 2$. Then, for any $\mathbf{x}\in\mathcal{B}_1(n,w)$, the cluster that contains $\mathbf{x}$ is $C_{\mathbf{x}}=\left\{\mathbf{y}\in\mathcal{B}_1(n,w):\mathbf{y}^C=\mathbf{x}^C\right\}$ and it has cardinality $2^{i(\mathbf{x})}$.*

Proof: The $C_{\mathbf{x}}$ expression follows from the definition of a cluster. From (7) it follows that $\mathbf{x}^{\mathrm{C}}$ has exactly $i(\mathbf{x})$ entries equal to L or H. Replacing the L-valued entries in $\mathbf{x}^{\mathrm{C}}$ by 0 or 1 and its H-valued entries by 2 or 3 generates all the words in the cluster. Hence, there are $2^{i(\mathbf{x})}$ words in the cluster that contains $\mathbf{x}$. ∎

As an immediate consequence of the previous two lemmas we have the following result.

Corollary 1. *Let n and w be integers satisfying $0\leq w\leq n$ and $n\geq 2$. It holds for any $\mathbf{x}\in\mathcal{B}_1(n,w)$ with $i(\mathbf{x})=0$ that $d(\mathbf{x},\mathbf{y})\geq 2$ for all $\mathbf{y}\in\mathcal{B}_1(n,w)$ with $\mathbf{y}\neq\mathbf{x}$.*

Let $\mathcal{I}(n,w)$ be defined as the subset of $\mathcal{B}_1(n,w)$ containing all words with index zero, i.e.,

$$\mathcal{I}(n,w)=\{\mathbf{x}\in\mathcal{B}_1(n,w):i(\mathbf{x})=0\}. \tag{15}$$

Its size is denoted by $I(n,w)$. We are now ready to state and prove the main result of this paper.

Theorem 2. *For $0\leq w\leq n$ and $n\geq 2$, it holds that*

$$B_1(n,w,2)=\frac{B_1(n,w)+I(n,w)}{2}. \tag{16}$$

Proof: Note that by Lemma 1 a code $\mathcal{C}\subseteq\mathcal{B}_1(n,w)$ with $d(\mathcal{C})=2$ and $|\mathcal{C}|=B_1(n,w,2)$ can be partitioned in maximal subsets, with minimal distance 2, of the clusters that partition $\mathcal{B}_1(n,w)$. So all clusters of size 1 are subsets of $\mathcal{C}$, which means that $\mathcal{I}(n,w)\subseteq\mathcal{C}$. Now take any cluster that contains more than one word. This cluster can be written as $C_{\mathbf{x}}$ for some word $\mathbf{x}$ from $\mathcal{B}_1(n,w)\setminus\mathcal{I}(n,w)$. We will show in the next paragraph that a maximal subset of $C_{\mathbf{x}}$ with minimal distance 2 contains half of the number of words of $C_{\mathbf{x}}$. From this we conclude that $\mathcal{C}$ contains half of the words of $\mathcal{B}_1(n,w)\setminus\mathcal{I}(n,w)$ and all words of $\mathcal{I}(n,w)$, i.e., its size is $(B_1(n,w)-I(n,w))/2+I(n,w)=(B_1(n,w)+I(n,w))/2$.

Let $i=i(\mathbf{x})$ be the cluster index of $C_{\mathbf{x}}$. According to Lemma 2, the cluster size equals 2^i. Replacing the L-valued entries in $\mathbf{x}^C$ by 0 or 1 and its H-valued entries by 2 or 3 generates all the words in the cluster. We map each word in $C_{\mathbf{x}}$ to a binary vector of length i by removing all symbols at positions j for which $x_j^{\mathrm{C}}\in\{0,1,2,3\}$, and then subtracting 2 from all the remaining entries equal to 2 or 3. Note that this mapping is a Hamming-distance-preserving bijection from $C_{\mathbf{x}}$ to the set $\mathcal{V}(i)$ of all binary vectors of length i. It follows from the well-known Singleton bound [3], that the largest subset of $\mathcal{V}(i)$, such that any two different words in the subset differ in at least 2 positions, has size at most 2^{i-1}. This upper bound can be achieved by selecting, e.g., all binary words of length i that contain an even number of ones. The inverse image of this set is a maximal subset of $C_{\mathbf{x}}$ with minimal distance 2 that contains half of the words of $C_{\mathbf{x}}$. This completes the proof. ∎

From the proof of Theorem 2 it is apparent how to construct a code $\mathcal{C}$ in $\mathcal{B}_1(n,w)$ with $d(\mathcal{C})\geq 2$ and $|\mathcal{C}|=B_1(n,w,2)$. For example, such an optimal code is

$$\mathcal{C}=\{\mathbf{x}\in\mathcal{B}_1(n,w):\sum_{i:x_i^{\mathrm{C}}\in\{\mathrm{L},\mathrm{H}\}}x_i\text{ is even}\}. \tag{17}$$

Observe that $\mathcal{I}(n,w)$ is indeed a subset of this code $\mathcal{C}$, since any $\mathbf{x}\in\mathcal{I}(n,w)$ has $x_i^{\mathrm{C}}\in\{0,1,2,3\}$ for all i. Hence, for such words the summation in (17) is over the empty set, resulting

in the value zero, which is even. Further, $\mathcal{C}$ contains half of $\mathcal{B}_1(n,w) \setminus \mathcal{I}(n,w)$.

Note that we have an expression for $B_1(n,w)$ from the previous section. Hence, if we derive an expression for $I(n,w)$ as well, then Theorem 2 enables the computation of $B_1(n,w,2)$. In order to do so, we define $\mathcal{M}(n,w)$ as the set of all words in $\mathcal{I}(n,w)$ ending with a zero or a one, i.e.,

$$\mathcal{M}(n,w) = \{\mathbf{x} \in \mathcal{I}(n,w) : x_n^{\mathrm{LH}} = L\}. \tag{18}$$

Its cardinality is denoted by $M(n,w)$. By a symmetry argument, we have

$$|\{\mathbf{x} \in \mathcal{I}(n,w) : x_n^{\mathrm{LH}} = H\}| = M(n,n-w). \tag{19}$$

Numerical values of $M(n,w)$ and $I(n,w)$ can be found as follows.

Theorem 3. *For $0 \leq w \leq n$ and $n \geq 2$, it holds that $M(n,0) = 2$, $M(n,w) = 0$ if $w \in \{1, n-1, n\}$,*

$$M(n,w) = M(n-1,w) + 2M(n-2,n-2-w) \tag{20}$$

if $2 \leq w \leq n-2$, and

$$I(n,w) = M(n,w) + M(n,n-w). \tag{21}$$

Proof: Note that $\mathcal{M}(n,w)$ consists of all the words $\mathbf{x}$ in $\mathcal{B}_1(n,w)$ for which it holds that $\mathbf{x}^{\mathrm{LH}}$ ends with the symbol L and, furthermore, that it has only runs of L-symbols and H-symbols of length at least two each. Hence, it contains only the words $01010\ldots$ and $10101\ldots$ of length n if $w = 0$, due to the runlength constraint, and no words at all if $w \in \{1, n-1, n\}$. This gives the stated expressions for $M(n,w)$ with $w \in \{0, 1, n-1, n\}$.

If $2 \leq w \leq n-2$, then note that the last three symbols of $\mathbf{x}^{\mathrm{LH}}$ are either a) LLL or b) HLL. The set of words in $\mathcal{M}(n,w)$ for which a) holds can be obtained by appending a unique extra symbol to each of the words from $\mathcal{M}(n-1,w)$. For all $\mathbf{y} \in \mathcal{M}(n-1,w)$, this extra symbol must be 0 if $y_{n-1} = 1$ and 1 if $y_{n-1} = 0$ to obtain a unique word from $\mathcal{M}(n,w)$. The set of words in $\mathcal{M}(n,w)$ for which b) holds can be obtained by appending two well-determined extra symbols to the words from $\mathcal{I}(n-2,w) \setminus \mathcal{M}(n-2,w)$. For all words in $\mathcal{I}(n-2,w) \setminus \mathcal{M}(n-2,w)$, we have two options for this tail, i.e., 01 and 10, to obtain a unique word from $\mathcal{M}(n,w)$. Hence, because of (19), the number of words in $\mathcal{M}(n,w)$ in the b) category is $2M(n-2,n-2-w)$. Together with the fact that the number of words in $\mathcal{M}(n,w)$ in the a) category is $M(n-1,w)$, (20) follows.

Finally, (21) is obvious, since it just states that the size of $\mathcal{I}(n,w)$ is the sum of $|\mathcal{M}(n,w)|$ and $|\mathcal{I}(n,w) \setminus \mathcal{M}(n,w)|$, where the expression for the latter follows from (19). ∎

Table II gives example values for $B_1(n,w)$ from Th. 1, $I(n,w)$ from Th. 3, and $B_1(n,w,2)$ from Th. 2. For $w = \lfloor n/2 \rfloor$ and $4 \leq n \leq 13$, lower bounds on $B_1(n,w,2)$ have been reported in [8, Table I], that were obtained via an altruistic algorithm. These bounds equal the corresponding values from Th. 2 and thus the codes obtained by that algorithm are optimal when $w = \lfloor n/2 \rfloor$ and $4 \leq n \leq 13$. However, the advantages of our code construction (17) are that no algorithm needs to be run to generate the codewords and that it is guaranteed to be the largest possible code for any w and n.

TABLE II
VALUES OF $B_1(n,w)$, $I(n,w)$, AND $B_1(n,w,2)$.

$B_1(n,w)$	$w=0$	$w=1$	$w=2$	$w=3$	$w=4$
$n=2$	2	8	2		
$n=3$	2	16	16	2	
$n=4$	2	24	56	24	2
$n=5$	2	32	128	128	32
$n=6$	2	40	232	424	232
$n=7$	2	48	368	1040	1040
$n=8$	2	56	536	2104	3352
$I(n,w)$	$w=0$	$w=1$	$w=2$	$w=3$	$w=4$
$n=2$	2	0	2		
$n=3$	2	0	0	2	
$n=4$	2	0	8	0	2
$n=5$	2	0	8	8	0
$n=6$	2	0	16	8	16
$n=7$	2	0	24	16	16
$n=8$	2	0	32	24	56
$B_1(n,w,2)$	$w=0$	$w=1$	$w=2$	$w=3$	$w=4$
$n=2$	2	4	2		
$n=3$	2	8	8	2	
$n=4$	2	12	32	12	2
$n=5$	2	16	68	68	16
$n=6$	2	20	124	216	124
$n=7$	2	24	196	528	528
$n=8$	2	28	284	1064	1704

V. CONCLUDING REMARKS

We have presented a recursive expression for $B_r(n,w)$, i.e., the number of quaternary words with length n, GC-weight w, and runlength constraint r. Furthermore, we have derived a recursive expression for $B_1(n,w,2)$, i.e., the size of the largest quaternary code with length n, GC-weight w, minimum Hamming distance 2, and no identical symbols next to each other in each codeword. An interesting research challenge is to find expressions or improve bounds for $B_r(n,w,d)$ with other values of r and/or d, i.e., for cases with a more relaxed runlength constraint and/or stronger error correcting/detecting capabilities.

REFERENCES

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

[2] S. M. H. T. Yazdi, H. M. Kiah, E. Garcia-Ruiz, J. Ma, H. Zhao, and O. Milenkovic, "DNA-based storage: trends and methods," *IEEE Trans. Mol. Biol. Multi-Scale Commun.*, vol. 1, no. 3, pp. 230-248, Sept. 2015.

[3] F. J. MacWilliams and N. J. A. Sloane, *The theory of error-correcting codes*, North-Holland, 1977.

[4] O. D. King, "Bounds for DNA codes with constant GC-content," *Electronic Journal of Combinatorics*, vol. 10, pp. 33-46, 2003.

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

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

[7] 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, 2020.

[8] D. Limbachiya, M. K. Gupta, and V. Aggarwal, "Family of constrained codes for archival DNA data storage," *IEEE Commun. Lett.*, vol. 22, no. 10, pp. 1972-1975, Oct. 2018.

[9] B. Cao, S. Zhao, X. Li, and B. Wang, "K-means multi-verse optimizer (KMVO) algorithm to construct DNA storage codes," *IEEE Access*, vol. 8, pp. 29547-29556, 2020.