IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received April 27, 2021, accepted May 3, 2021, date of publication May 6, 2021, date of current version May 14, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3078138

Is BCH Code Useful to DNA Classification as an Alignment-Free Method?

MILENA M. ARRUDA[1], (Student Member, IEEE), FRANCISCO M. DE ASSIS[1], AND TACIANA A. DE SOUZA[2]

[1]Department of Electrical Engineering, Federal University of Campina Grande, Campina Grande 58429-970, Brazil
[2]Department of Mathematics, Federal Institute of Paraíba, Cajazeiras 58900-000, Brazil

Corresponding author: Milena M. Arruda (milena.arruda@ee.ufcg.edu.br)

This work was supported by the Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq).

ABSTRACT Similarities between biological and digital communication systems have been investigated since biology also uses a discrete alphabet to represent and transmit information. The genetic information of an organism is encoded in DNA molecules by units called bases. However, there is no a definitive model and the question as what error-correcting code underlies DNA sequences remains an open problem. Recent works show that DNA sequences can be identified as codewords in a class of cyclic error-correcting codes known as BCH codes. We propose improvements regarding the code construction process that resulted in a novel algorithm for searching BCH codes whose codeword differ from a given DNA sequence (mapped to finite field $\mathbb{F}_4$) in up to only one symbol. The most important improvement is to replace brute force decoding with syndrome decoding. In this sense, based on a statistical analysis, we verify whether in a collection of sequences with the same taxonomic rank there is a code that identifies most of these sequences, called dominant code. Furthermore, we check whether the dominant code can provides a biological information to DNA classification being an alignment-free method. Finally, we show that the probability of a DNA sequences with odd-length n be identified by a BCH code tends to analytical probability of the same code identifying a random vector.

INDEX TERMS BCH codes, DNA sequences, error-correcting codes, finite field, genetic coding, genomic signal processing.

I. INTRODUCTION

The use of coding and information theory tools has been proposed in bioinformatics. For example, DNA based data storage systems [1], [2], hiding data in DNA [3], [4] and find error-correcting code underlying DNA sequences [5], [6]. The latter is the research focus of this paper, whose most common questions are: is there an error-control mechanism in biological sequences similar to the error-correcting codes employed in digital communication systems? [7]–[10]; are there codes which are able to identify and reproduce DNA sequences? [11]–[15].

These similarities between biological and digital communication systems have been investigated since biology also uses a digital code to represent and transmit information. The genetic information is encoded in DNA molecules by a sequence of units called bases: adenine (A), cytosine (C), guanine (G) and thymine (T); these bases are the fundamental elements of DNA strands which are translated to proteins that carry out the functions of living cells [16].

Although relevant, such studies have not yet provided a definitive answer or model. Liebovitch *et al.* [11] were the first ones to introduce a methodology to determine whether a linear block error-correcting code is present in DNA, and Rosen [12] continued this investigation presenting a method to uncover an error-correcting code structure in the nucleotide sequence and detect tandem repeats. Faria *et al.* [13] and Rocha *et al.* [14] were more specific and proposed an algorithm, known as DNA Sequence Generator Algorithm, which verifies whether a given DNA sequence can be identified as codewords of a BCH code of design distance $d = 3$ over finite fields and rings of integers, respectively. In this context, a DNA sequence is said to be identified whether it is a codeword for a BCH code or whether it differs from the codeword

The associate editor coordinating the review of this manuscript and approving it for publication was Faissal El Bouanani.

up to one single nucleotide. In the biological context, this mismatch is known as a single nucleotide polymorphism (SNP).

Despite the fact that the algorithms found in the literature still have limitations to be solved, these algorithms have been used by the scientific community. For example, the DNA Sequence Generator Algorithm was used by Faria *et al.* [15] to show that a gene and even a plasmid genome can be identified as codeword of BCH codes; by Brandão *et al.* [17], as a tool to address the evolutionary pathway of the genetic code of some genes and to investigate the biological meaning of the single mismatch between the original DNA sequence and the sequence identified as the codeword; and by Duarte-González *et al.* [10], to represent protein sequences and explore evolutionary relationships. In order to solve restrictions with respect to the DNA sequence length and to reduce the computational effort of this algorithm, Rodríguez-Sarmiento *et al.* [5] and Hernádez *et al.* [6] proposed a novel algorithm to identify odd-length biological sequences as codewords of BCH codes over finite field and ring of integers, respectively.

However, the DNA Sequence Generation Algorithm runs $3n + 1$ times, where n is the sequence length, since for each sequence in the set of neighboring sequences (i.e., the set of sequences whose Hamming distance between it and the original is unitary) it is checked whether it is a codeword of some BCH code. This paper investigates such limitation and proposes a solution to this problem by replacing brute force for a syndrome decoding approach. For the sake of easiness, the Peterson–Gorenstein–Zierler (PGZ) decoding algorithm [18] was used here, despite other more sophisticated algorithms that could, in principle, be used with the similar results.

In this sense, new questions may emerge, such as: if there is an error-correcting code underlying a DNA sequence, does it reveal similarities among DNA sequences of neighboring organisms on a phylogenetic tree? If so, since numerical and graphical representations of DNA sequences (these operations are also referred to as mapping) have been used to propose alignment-free classification methods of DNA sequences [19]–[21], can the BCH codes be used as an alignment-free classification method? In order to answer these questions, we analyzed the statistical significance of finding such codes. The idea was to compare the probability of a BCH code identifying a random vector with the probability of identifying a true DNA sequence.

Therefore, in this paper, besides proposing improvements in the DNA Sequence Generation Algorithm that resulted in a novel algorithm, we investigate if it is effective to determine the BCH code that identifies most of DNA sequences in a collection, where the sequences stem from neighboring organisms in a phylogenetic tree. We refer to this code as the dominant code. The goal is to verify whether the dominant code can provide a biological classification being an alignment-free method. Finally, the statistical analysis reveals that a dominant code can be designed only according to the length of sequences and without biological classification. The algorithm was implemented in SageMath, a free open-source mathematics software that uses a syntax resembling Python.

This paper is organized as follows: Section II provides notations and definitions. In Section III, we derive the main result, that is, we detail the proposed algorithm and discusses its most important features. The results are then presented in Section IV, and the conclusions are elaborated in Section V.

II. PRELIMINARIES

In this section, we describe some notations and definitions that are essentially important to coding theory analysis in this paper, for more details we recommend [18].

Let $\mathbb{F}_q$ be a finite field of order q, where q is a power of a prime, and for any positive integer m, let $\mathbb{F}_{q^m}$ be an extension field of $\mathbb{F}_q$. The set of all vectors of length n over $\mathbb{F}_q$ is denoted by $\mathbb{F}_q^n$. A vector $\mathbf{u} \in \mathbb{F}_q^n$ is denoted by bold letter. Two finite fields with the same number of elements are said to be isomorphic. Let α be a primitive element of the field $\mathbb{F}_{q^m}$, where $\alpha^s \in \mathbb{F}_{q^m}$ and $0 \leq s < q^m$, then the cyclotomic coset of α^s is given by $\{\alpha^s, \alpha^{sq}, \alpha^{sq^2}, \ldots, \alpha^{sq^{m-1}}\}$ whose elements are the conjugates of α^s. The prime polynomial of the smallest degree over $\mathbb{F}_q$ with $f(\alpha^s) = 0$ is called the minimal polynomial of α^s whose degree is a divisor of m and its roots are α^s and its conjugates.

A linear code $\mathcal{C}$ is a subspace of $\mathbb{F}_q^n$ referred to as an (n, k, d) code where n is the length, k is the dimension and d is the minimum Hamming distance. The Hamming distance between two vectors $\mathbf{u}$ and $\mathbf{v}$ is the number of places in which $\mathbf{u}$ and $\mathbf{v}$ differ, that is, $|\{i \mid u_i \neq v_i\}|$. These codes can always decode uniquely if the number of errors is up to $t = \lfloor \frac{d-1}{2} \rfloor$ errors. The BCH codes form a class of linear codes. In this paper, we denote a BCH code denoted by $\mathcal{C}_{BCH}$ over $\mathbb{F}_q$ as a (n, k, d) code, where n is a positive integer, n divide $q^m - 1$ and α is an element of multiplicative order n in $\mathbb{F}_{q^m}$, defined by the following generator polynomial,

$$g(x) = \text{lcm}(f_b(x), \ldots, f_{b+(d-2)\ell}(x)), \tag{1}$$

where b, d and ℓ are positive integers such that $0 \leq b < n$, $1 \leq d \leq n$ and $\gcd(\ell, n) = 1$; in addition, $f_i(x)$ is the minimal polynomial of α^i.

Suppose that $g_1(x)$ and $g_2(x)$ are generator polynomials for two codes, $\mathcal{C}_1$ and $\mathcal{C}_2$, of the same blocklength over $\mathbb{F}_q$, if all roots of $g_1(x)$ are also roots of $g_2(x)$, which implies that $g_1(x)$ divides $g_2(x)$, then $\mathcal{C}_2$ is a subcode of $\mathcal{C}_1$. Once in our applications we are interested in $\mathcal{C}_{BCH}$ over $\mathbb{F}_4$ with design distance $d \geq 3$, notice that, for fixed b and ℓ, codes where $d > 3$ are subcodes of codes where $d = 3$. So, from now on $\mathcal{C}_{BCH}$ is an $(n, k, 3)$ code.

Given a code, the decoding is the process of error detection and correction; thus, decoding a received vector means translating it into a codeword. Notice that each codeword has a decoding sphere of radius $t = \lfloor \frac{d-1}{2} \rfloor$ drawn around it, in which there are all words that will be decoded into that codeword. Fig. 1 illustrates the three regions in vector space, into which a vector can fall. The first region is when the vector

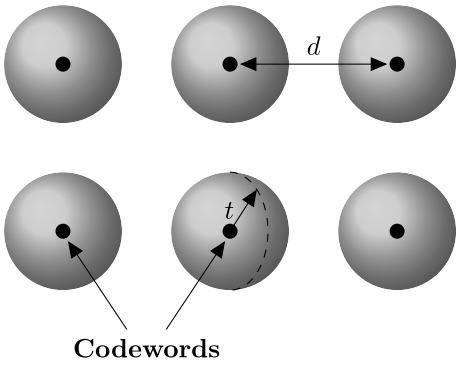


FIGURE 1. Decoding regions.

is a codeword; the second one is the shaded region, that is, when the vector is in any sphere; and the third region is the white region, in which lie words that are not within distance t of any codeword.

The PGZ decoder [18] for these codes can, therefore, correct up to 1 error. It means that, given a $\mathcal{C}_{BCH}$, if the PGZ decoder translates a received vector, $\mathbf{r}$, into valid codeword, $\mathbf{c}$, the Hamming distance between $\mathbf{r}$ and $\mathbf{c}$ is 1. The first step of the PGZ decoder is to calculate the syndromes: $\mathbf{r}$ is considered as a polynomial and evaluating it at roots of $g(x)$. The decoding problem is then reduced to solve the following system of nonlinear equations:

$$\begin{cases} s_b = \epsilon\alpha^{bi}, \\ s_{b+\ell} = \epsilon\alpha^{(b+\ell)i} = s_b\alpha^i, \end{cases} \tag{2}$$

where $s_j = r(\alpha^j)$ for $j = b, b+\ell$ is the syndrome for a given vector $\mathbf{r}$; in addition, the unknown variables are the location of the error i (the first position of a vector corresponds to $i = 0$) and its magnitude ϵ. If the syndromes are all zero, then there is no error and the decoding is done. Otherwise, once the error location is known by solving the second equation of the nonlinear system (2), the error magnitudes must be computed by the first equation. The decoder, therefore, returns a valid codeword (whose Hamming distance from the codeword is less than or equal to one) whether $0 \leq i < n$ and $\epsilon \in \mathbb{F}_q$. If only one syndrome is zero, then the number of errors will exceed the error correction capability of a $\mathcal{C}_{BCH}$ code; translation would fail and an empty word would be returned.

III. ALGORITHM DESIGN

A. DNA SEQUENCE GENERATION ALGORITHM OVERVIEW

The DNA Sequence Generation Algorithm checks whether a given DNA sequence can be identified by an error-correction code, specifically by a BCH code of design distance $d = 3$, considering the simplicity of its encoding and decoding processes. A DNA sequence is said to be identified whether it is a codeword for a BCH code or it differs from the codeword up to one single nucleotide.

In order to evaluate it, several attributes of DNA sequences are associated with code parameters. Firstly, the BCH codes have an associated algebraic structure for which the nucleotides must be mapped. The algebraic structures used in the DNA Sequence Generation Algorithm were finite field [13] and ring of integers [14], in which they both have four elements. There are twenty-four possible bijection labeling maps from $\{A, C, G, T\}$ to $\mathbb{F}_4$; however, according to [13], any of these labels arise with equal identification results when using BCH codes over $\mathbb{F}_4$.

For a fixed length n, there is more than one BCH code defined by different generator polynomials. The authors propose that each code is unique in an extension field; therefore, for each existing primitive polynomial, the elements of extension field is calculated and a generator polynomial is defined (this procedure has an expensive memory cost, since isomorphic extension fields are repeatedly constructed for each primitive polynomial).

Next, for a given DNA sequence, the following verification is repeated: the parity-check matrix of a code is used to decide whether a given DNA sequence is a codeword, so brute force is used to analyze the sequences with only one different nucleotide (the other three nucleotide possibilities were considered at each symbol position of the DNA sequence, and, then, using the parity-check matrix, it is decided whether each edited sequence is a codeword, more specifically, this process is repeated $3n + 1$ times). Notice that this yields a large number of tests to be fulfilled.

Whereas this algorithm checks if a sequence is the codeword of a BCH code by using the parity-check matrix, the novel algorithm proposed by Rodríguez-Sarmiento *et al.* [5] uses a root finding or factorization method. However, the brute force is still an open problem with respect to computational efforts.

B. PROPOSED ALGORITHM

The algorithm starts by associating attributes (like alphabet and length) of DNA sequences with code parameters. The alphabet of genetic code (given by the set $\mathcal{N} = \{A, C, G, T\}$) and the alphabet of $\mathcal{C}_{BCH}$ (given by the set $\mathbb{F}_4 = \{0, \beta, \beta^2, 1\}$) must be linked by a bijective mapping. Although there are twenty-four bijective maps from $\mathcal{N}$ to $\mathbb{F}_4$, it is well-known that finite fields with the same cardinality are isomorphic; therefore, we have considered just one of those maps along the proofs [13].

From now on, let $\mathcal{C}_{BCH}$ be a BCH code over $\mathbb{F}_4$. Bases of a DNA sequence will be mapped onto $\mathbb{F}_4$ as shown in Fig. 2. For instance, suppose that a given DNA sequence is *CTGATCCTTCAAGCG*, then, this is mapped to the vector $\mathbf{r} = (\beta, 1, \beta^2, 0, 1, \beta, \beta, 1, 1, \beta, 0, 0, \beta^2, \beta, \beta^2)$. Notice that the BCH codes are defined under formal definition, in which different generator polynomials are specified by b and ℓ; therefore, it is not necessary to use different primitive polynomials to construct isomorphic extension fields, finding only one correspondent generator polynomial.

The codeword length n must match the length of the DNA sequence; consequently, we will consider only those sequences whose length can be written as a divisor of $4^m - 1$, where m is a positive integer. Thus, n is always odd and

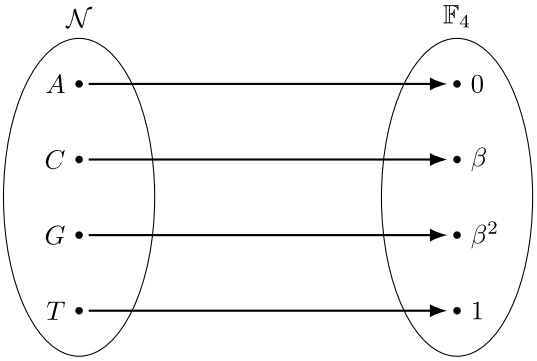


FIGURE 2. Bijective mapping of alphabets. Each element of the genetic code in the set $\mathcal{N}$, is mapped to exactly one element of the finite field with order four denoted by $\mathbb{F}_4$. This will result in a bijective mapping as follows: $\mathcal{N} \to \mathbb{F}_4 : A \mapsto 0,\ C \mapsto \beta,\ G \mapsto \beta^2,\ T \mapsto 1$.

Algorithm 1 Proposed Algorithm

Input: DNA sequence
Output: BCH codes

```
Initialize r to DNA sequence
Initialize m according to (3)
Initialize range_b
Initialize range_ℓ
for each b in range_b do
    for each ℓ in range_ℓ do
        if b and ℓ solve (2) then
            Add g(x) = lcm(f_b(x), f_{b+ℓ}(x)) to R
        end if
    end for
end for
return R
```

satisfies,

$$n \mid 4^m - 1 \tag{3}$$

where $m \in \mathbb{N}$. The $\mathcal{C}_{BCH}$ will be completely specified whether there is any b and ℓ such that the decoder, summarized by the nonlinear system of equations (2), has as solution a valid codeword.

The proposed algorithm starts with a given DNA sequence mapped to the vector $\mathbf{r}$ and a default $\mathcal{C}_{BCH}$ (we picked out usual parameters $b = 0$ and $\ell = 1$). Remember that the generator polynomial is

$$g(x) = \text{lcm}(f_b(x), f_{b+\ell}(x)) \tag{4}$$

where the conditions about the range of b and ℓ are $0 \leq b, \ell < n$ and $\gcd(n, \ell) = 1$. Then, $\mathbf{r}$ is mapped to the vector n-dimensional with elements in $\mathbb{F}_4$ and is considered as a polynomial. Next, the algorithm checks if the decoder returns a valid codeword to the default $\mathcal{C}_{BCH}$, and if so, it saves the generator polynomial in a set $\mathcal{R}$. The algorithm stops when the entire range of b and ℓ is checked, and returns the set $\mathcal{R}$ of all generator polynomials that identify the input DNA sequence. The pseudocode is shown in Algorithm 1. In addition, the SageMath implementation is available in a GitHub repository [22].

However, two restrictions about ℓ can improve its range. First, take into account only one value for ℓ in each set $\{s, sq, \ldots, sq^{m-1}\} \mod n$ where $0 \leq s < n$. Second, consider either ℓ or $-\ell \mod n$. To understand the reason for these restrictions remember that $\alpha^{q^m} = \alpha$ and consider the codes defined by the following generator polynomials,

$$\begin{cases} g_1(x) = \text{lcm}(f_b(x), f_{b+s}(x)), \\ g_2(x) = \text{lcm}(f_{b'}(x), f_{b'+sq}(x)), \\ \vdots \\ g_j(x) = \text{lcm}(f_{b''}(x), f_{b''+sq^{m-1}}(x)). \end{cases} \tag{5}$$

The equality $g_1(x) = g_2(x)$ occurs if $b' = bq$. Similarly, the equality $g_1(x) = g_j(x)$ occurs if $b'' = bq^{m-1}$. Of course, since the range of b is 0 to $n-1$, it is redundant to take more than one ℓ in each set $\{s, sq, \ldots, sq^{m-1}\} \mod n$. Thus, the first restriction was proved and the second follows the same argument. For example, for a $\mathcal{C}_{BCH}$ of length 15, the set in which $\gcd(\ell, n) = 1$ is $\{1, 2, 4, 7, 8, 11, 13, 14\}$. However, it can be seen that by the first restriction $\ell \in \{1, 2, 7, 11\}$ and by the second $\ell \in \{1, 2\}$.

Once the BCH code for a given DNA sequence is known, the dominant code is found as follows. Let $\mathcal{A} = \{\mathbf{r}_1, \mathbf{r}_2, \ldots, \mathbf{r}_N\}$ be a collection of N DNA sequences and $\mathcal{R}_1, \mathcal{R}_2, \ldots, \mathcal{R}_N$ be the sets of $\mathcal{C}_{BCH}$ that identify each one, respectively. The dominant code is a $\mathcal{C}_{BCH}$ such that it identifies most of the DNA sequences in $\mathcal{A}$. This code lies in the intersection between M sets of $\mathcal{R}_1, \mathcal{R}_2, \ldots, \mathcal{R}_N$ where M is the largest integer such that this intersection is nonzero. Finally, M/N is the fraction of the DNA sequences identified by the same $\mathcal{C}_{BCH}$, the dominant code.

IV. RESULTS

A. EXPERIMENTAL DATA

The data is available at nucleotide database from National Center for Biotechnology Information (NCBI) that provides open access to biomedical and genomic information [23]. All sequences have an identifier, the GI number, a simple series of digits processed by NCBI. When we analyze an individual DNA sequence, we refer to it by its GI number. When we analyze collections of N sequences the following search is done in NCBI: txid$X$[Organism] AND "cds" [Feature key] AND Y [Sequence Length] NOT hypothetical protein NOT predicted NOT partial, where txid$X$ is replaced by the taxonomic identifier of specific organism and Y is replaced by desired sequence length. The taxonomic identifiers of Bacteria, Fungi and Plantae are txid2, txid4751 and txid33090, respectively. Next, the analysis is performed drawing N sequences at random. Furthermore, whenever there were any degenerate bases in a DNA sequence, this was discarded (for example, W is a degenerate base because it can represent A or T).

B. ALGORITHM DETAILS

With some effort, the steps of the algorithm can be made by hand in a simple example as for the short DNA sequence 33079225 given by *CTGATCCTTCAAGCG*. Notice that this sequence has 15 bases that can be written as $4^2 - 1$; thus, the codeword length is $n = 15$ and the extension degree is $m = 2$.

The algorithm checks which BCH codes over $\mathbb{F}_4$ with length 15 and design distance 3 can decode this sequence. Following the steps of our algorithm, this sequence is mapped to the vector $\mathbf{r} = (\beta, 1, \beta^2, 0, 1, \beta, \beta, 1, 1, \beta, 0, 0, \beta^2, \beta, \beta^2)$ whose polynomial is:

$$r(x) = \beta^2 x^{14} + \beta x^{13} + \beta^2 x^{12} + \beta x^9 + x^8 + x^7 + \beta x^6 + \beta x^5 + x^4 + \beta^2 x^2 + x + \beta. \quad (6)$$

The default $\mathcal{C}_{BCH}$ has generator polynomial specified by $b = 0$ and $\ell = 1$. Thus, from (2) and using the arithmetic of $\mathbb{F}_{16}$, the syndromes are: $s_0 = 1$ and $s_1 = \alpha^{12}$. Then, we find that the error occurred in the twelfth component ($\alpha^i = \frac{s_1}{s_0} = \alpha^{12}$) and that the error magnitude is $\epsilon = 1$. So, the decoder returns the following codeword whose Hamming distance from $\mathbf{r}$ is one: $\mathbf{c} = (\beta, 1, \beta^2, 0, 1, \beta, \beta, 1, 1, \beta, 0, 0, \beta, \beta, \beta^2)$. It corresponds to the sequence *CTGATCCTTCAACCG*. This generator polynomial is saved in $\mathcal{R}$. Next, the remaining codes specified by $0 \leq b < n$ and $\ell \in \{1, 2\}$ are checked, and so on. When the algorithm stops, there are twelve codes in $\mathcal{R}$. These codes have generator polynomials as listed in Table 1. Furthermore, for each one of these, the position, the old symbol and the new symbol in this position are listed. Remember that in algebraic arithmetic the first position of a vector corresponds to $i = 0$. The old symbol is the nucleotide in position i of the original DNA sequence and the new symbol is the nucleotide in position i of the codeword whose distance from DNA sequence is one.

TABLE 1. Generator polynomials of $\mathcal{C}_{BCH}$ with $n = 15$ that identify the sequence *CTGATCCTTCAAGCG*.

b	ℓ	*Generator polynomial*	*Position i*	*Old*	*New*
0	1	$x^3 + \beta^2 x + \beta$	12	G	C
3	1	$x^4 + \beta x^3 + \beta$	2	G	A
4	1	$x^3 + \beta^2 x^2 + \beta^2$	2	G	A
7	1	$x^4 + \beta^2 x^3 + \beta^2 x^2 + \beta^2 x + 1$	10	A	G
10	1	$x^3 + x + \beta$	3	A	C
14	1	$x^3 + \beta x^2 + \beta^2$	13	C	G
0	2	$x^3 + \beta x + \beta^2$	0	C	G
3	2	$x^3 + x^2 + \beta$	2	G	A
5	2	$x^3 + x + \beta^2$	0	C	A
8	2	$x^3 + \beta x^2 + \beta$	10	A	G
10	2	$x^3 + \beta^2 x + \beta^2$	7	T	C
13	2	$x^3 + \beta^2 x^2 + \beta$	5	C	G

C. BIOLOGICAL ANALYSIS

For the example above (DNA sequence 33079225), there is always a difference of a single nucleotide between the original DNA sequence and the one returned by the algorithm. In the biological context, it can mean a SNP or an ancestral sequence (since the BCH code is an error-correcting code, so it may be acting to protect against mutations). Observe that for the code whose generator polynomial is $g(x) = x^3 + \beta^2 x + \beta$, this mismatch occurred at position 12 resulting in a transversion mutation. Furthermore, five other codes identify the DNA sequence by a transversion mutation. For the remaining codes, the SNP causes a transition mutation. Transversions are a point mutation that changes a purine base (A or G) to a pyrimidine (C or T), and transitions refer to changes from a purine base to another purine, or a pyrimidine base to another pyrimidine. Although the SNP represents a mutation, it can be silent during protein translation.

This is what happens for the DNA sequence whose GI number is 1852346641 (a genome shotgun sequence of *Streptomyces coelicolor* of length 255 whose molecule type is genomic DNA and product is MFS protein transporter, one of the two largest families of membrane transporters found on Earth [24]). This sequence of nucleotides is translated to the protein WP_173944011.1 by using the bacterial, archaeal and plant plastid code and making the codon start at position 3. There are 128 codes that identify this sequence, including the codes with the following generator polynomials:

$$g_1(x) = x^5 + \beta^2 x^4 + \beta^2 x^3 + x^2 + 1,$$
$$g_2(x) = x^5 + x^4 + \beta x^3 + \beta. \quad (7)$$

Table 2 shows this sequence and the codewords returned by the algorithm for $g_1(x)$ and $g_2(x)$. In this table, the first column refers to the index position of the first base of each DNA sequence in the second column. Each of these is labeled with an abbreviation as follows: Ont is the original nucleotide, Gnt1 is the codeword nucleotide given by $g_1(x)$ and Gnt2 is the codeword nucleotide given by $g_2(x)$. The mismatches with respect to the original sequence given by both codes are highlighted in red whose respective codons are underlined.

Notice that between Ont and Gnt1, the SNP results in a transition mutation ($G \rightarrow A$ in position 155); however, the codon where the mismatch occurred is translated to the same amino acid, an aspartic acid whose abbreviation is Asp or D. In the biological context, this mismatch represents a silent mutation. In contrast, although between Ont and Gnt2, there is also a transition mutation ($A \rightarrow G$ in position 204), the respective codon is translated to different amino acids, change from a Leucine (Leu or L) to a Proline (Pro or P). Remember that the original sequence is a molecule of DNA; therefore, this sequence must be converted into its reverse complement before translating the codons. For this sequence, the algorithm returns 128 codes that identify it, among which, 47 result in a silent mutation.

TABLE 2. ***Streptomyces coelicolor*** **genome shotgun sequence with GI number 1852346641.**

Index	*Sequence*			
1	Ont:	GCTGGGAGAC	GGCGATGCCG	ATGACGATCG
	Gnt1:	GCTGGGAGAC	GGCGATGCCG	ATGACGATCG
	Gnt2:	GCTGGGAGAC	GGCGATGCCG	ATGACGATCG
31	Ont:	CGGCCTGCTG	GAAGGAACCG	AGCCGGCCGC
	Gnt1:	CGGCCTGCTG	GAAGGAACCG	AGCCGGCCGC
	Gnt2:	CGGCCTGCTG	GAAGGAACCG	AGCCGGCCGC
61	Ont:	GGTAGGCGGG	CGGGGACACC	TCGGCGATGT
	Gnt1:	GGTAGGCGGG	CGGGGACACC	TCGGCGATGT
	Gnt2:	GGTAGGCGGG	CGGGGACACC	TCGGCGATGT
91	Ont:	AGGCGGGGCC	GATCACCGAG	GCCATGCCGA
	Gnt1:	AGGCGGGGCC	GATCACCGAG	GCCATGCCGA
	Gnt2:	AGGCGGGGCC	GATCACCGAG	GCCATGCCGA
121	Ont:	TCGCGAAACC	GCCGATGATC	CGCCACATGG
	Gnt1:	TCGCGAAACC	GCCGATGATC	CGCCACATGG
	Gnt2:	TCGCGAAACC	GCCGATGATC	CGCCACATGG
151	Ont:	CCAGGTCCCA	CAGCGCGAAG	GGCAGCGCGG
	Gnt1:	CCAGATCCCA	CAGCGCGAAG	GGCAGCGCGG
	Gnt2:	CCAGGTCCCA	CAGCGCGAAG	GGCAGCGCGG
181	Ont:	AGCCGACGGC	GCTCACCGTG	AACAGGACCG
	Gnt1:	AGCCGACGGC	GCTCACCGTG	AACAGGACCG
	Gnt2:	AGCCGACGGC	GCTCACCGTG	AACGGGACCG
211	Ont:	CGGCGATCTG	CATGCAGCGG	ATACGGCCGA
	Gnt1:	CGGCGATCTG	CATGCAGCGG	ATACGGCCGA
	Gnt2:	CGGCGATCTG	CATGCAGCGG	ATACGGCCGA
241	Ont:	TACGGTCCGC	GATAC	
	Gnt1:	TACGGTCCGC	GATAC	
	Gnt2:	TACGGTCCGC	GATAC	

D. PERFORMANCE COMPARISON

For the purpose of comparison, we consider almost the same set of DNA sequences as in [13] (some records have already been removed from database). In general, our algorithm returns more BCH codes near the DNA sequence. For instance, for the DNA sequences 78096542 and 45368559, instead of only one BCH code, our algorithm returns 34 and 92 codes, respectively. For the sequences 51093376 and 832917, instead of two BCH codes, our algorithm returns 32 codes for both. As expected, the BCH codes found by [13] were also found by us; however, our algorithm returns more codes for each DNA sequence.

Compared with the algorithm proposed by [5], the result could not be different: our algorithm returns the same codes for the following DNA sequences HP283558.1, HP425961.1, HP347514.1, HP253977.1, HP296666.1, HP320974.1, HP352962.1, HP278326.1, EZ071796.1 and EZ111718.1. For the remaining sequences (AK280992.1, HP466062.1, HP933108.1, HP823668.1), the algorithm in [5] returns BCH codes with $d \geq 3$ (remember that they are subcodes of codes with $d = 3$ and, therefore, there are common codewords between them). Thus, whether a given sequence is identified by a code $\mathcal{C}_1$ with $d > 3$ and $g_1(x)$, it is also identified by a code $\mathcal{C}_2$ with $d = 3$ and $g_2(x)$, in which, $g_1(x)$ divides $g_2(x)$. In this case, both codes translated the original sequence to the same codeword.

For instance, when the DNA sequence AK280992.1 is the input of Algorithm 1, the cardinality of $\mathcal{R}$ is 18. Whether, in this set, there is a code whose translated codeword has more than two consecutive roots, so, there is a subcode with $d > 3$ completely defined by the number of consecutive roots according to (1). In this particular case, all codes in $\mathcal{R}$ translated the original sequence to the same codeword. This codeword has up to 18 consecutive roots, therefore, this sequence is also identified by the code $(95, 5, 19)$ with the following generator polynomial,

$$\begin{aligned} g(x) = {} & x^{90} + x^{85} + x^{80} + x^{75} + x^{70} + x^{65} + x^{60} \\ & + x^{55} + x^{50} + x^{45} + x^{40} + x^{35} + x^{30} + x^{25} \\ & + x^{20} + x^{15} + x^{10} + x^5 + 1, \end{aligned} \tag{8}$$

where $b = 1$ and $\ell = 1$.

The algorithms proposed by [5] and [13] repeat the following methods $3n + 1$ times in a brute force decoding: root finding process and matrix-check verification, respectively. These methods can become very complex, mainly when the extension field degree increases. Notice that knowing which codes with $d = 3$ identifying a given DNA sequence will be enough to find subcodes with $d > 3$, if necessary. Improvements in the decoding process, replacing brute force with the PGZ decoder, result in fewer operations performed. As a consequence, the execution time of the algorithm proposed in this paper is shorter.

E. STATISTICAL ANALYSIS

The probability of a $\mathcal{C}_{BCH}$ identifying a DNA sequence can be interpreted as the probability of a DNA sequence lying in sphere decoding of the $\mathcal{C}_{BCH}$ (see Fig. 1). However, in general, the probability of these events is not trivial, but a satisfactory analytical probability can be computed for some special cases [18].

The DNA sequence decoding problem is analogous to considering linear codes being used on channels that make symbol errors independently. A decoder will decode every word to the closest codeword, provided that it is within distance t of the codeword. We can analyze the decoder performance when a vector whose symbols are independent and identically distributed (iid) is picked out from n-dimensional vector space over $\mathbb{F}_4$. The probability of this vector being identified by a $\mathcal{C}_{BCH}$ is the probability of correct decoding (remember that $d = 3$, then, $t = 1$). That is, the ratio of the sum of q^k codewords and $n(q-1)$ words whose Hamming distance from the codeword is unitary to total of words in a

n-dimensional vector space q^n, so,

$$\mathrm{P} = \frac{q^k + q^k n(q-1)}{q^n}, \tag{9}$$

or, an alternative and more appropriate way is

$$\mathrm{P} = \left(\frac{1}{q^{n-k}} + \frac{n(q-1)}{q^{n-k}}\right). \tag{10}$$

Notice that this probability is higher when the $\mathcal{C}_{BCH}$ has the largest dimension k. In this case, the generator polynomial has minimum degree. Thus, for m fixed, the probability P in (10) is maximized when

$$\text{n-k} = \min_{\mathcal{C}_{BCH}} \deg g(x). \tag{11}$$

Equation (11) returns the minimum degree of a generator polynomial given all BCH codes over a vector space $\mathbb{F}_4$ whose length is n and degree extension is m. For some specific codes, these values are summarized in the third column of Table 3. Furthermore, the probability that a n-dimensional random vector over $\mathbb{F}_4$ will be identified by a $\mathcal{C}_{BCH}$ with generator polynomial of minimum degree is a function of n, and it is summarized in the fourth column of Table 3 for some codes. Since a perfect code is one for which there are spheres of equal radius about the codewords that are disjoint and that completely fill the space [18], from Table 3 the codes (341, 336, 3) and (5461, 5454, 3) are perfect codes.

TABLE 3. Probability that a n-dimensional vector over $\mathbb{F}_4$ whose symbols are iid will be identified by a $\mathcal{C}_{BCH}$ in which the generator polynomial has minimum degree.

m	n	$\min_{\mathcal{C}_{BCH}} \deg g(x)$	P
4	255	5	0.75
5	341	5	1
6	273	7	0.05
7	5461	7	1
12	7735	10	0.022

The probability of a given $\mathcal{C}_{BCH}$ identifying a DNA sequence will be estimated by performing computer simulations. In this sense, we must observe, in a collection of N sequences with the same taxonomic rank, whether the fraction M/N tends to P. If so, we can conclude that the DNA sequences are distributed almost uniformly, under Hamming metric, in a n-dimensional vector space. Therefore, a classification would be possible whether M/N does not converge to P and there are different dominant codes for different collections. In this case, we are able to distinguish two or more collections.

In Table 4, we present the dominant codes for collections of N DNA sequences (coding sequences) according to the three kingdom classification. Notice that, in a general sense, the percent of sequences identified by a code (the ratio of M to N) tends to the analytical probability of the same code identifying a random vector. This is especially observed when

TABLE 4. Dominant codes for different codewords lengths and for both collections.

n	*Kingdom*	N	$\frac{M}{N}$	$g(x)$	P
255	Bacteria	500	77,15 %	$x^5 + x^4 + \beta^2 x^3 + x + \beta$	75%
		700	75,86 %	$x^5 + \beta x^3 + \beta x^2 + x + \beta$	
	Fungi	500	78,74 %	$x^5 + \beta x^2 + 1$	
	Plantae	500	83,99 %	$x^5 + \beta x^4 + \beta x^3 + x^2 + \beta^2 x + 1$	
341	Bacteria	500	100%	$x^5 + \beta^2 x^3 + \beta x^2 + \beta x + 1$	100%
	Fungi	479			
	Plantae	500			
273	Bacteria	500	8,45 %	$x^7 + \beta x^6 + \beta^2 x^5 + \beta^2 x^4 + x^3 + x^2 + \beta x + 1$	5%
		2000	6,37 %	$x^7 + x^6 + x^4 + \beta^2 x^3 + \beta^2 x + 1$	
	Fungi	1800	7,91 %	$x^7 + \beta x^5 + x^4 + x^2 + \beta$	
	Plantae	990	10,04 %	$x^7 + \beta^2 x^6 + x^4 + \beta x^3 + \beta^2 x + \beta^2$	

we compare the first two lines (or lines eighth and nine) of Table 4, in which the cardinality of collections grows and the M/N tends to P.

The first important observation is that, as expected, a perfect code with parameters (n, k, d) can identify any n-dimensional vector, and, consequently, any DNA sequence. In Table 4, the perfect code is the $(341, 336, 3)$. Another relevant observation is that all dominant codes have minimum degree. Furthermore, codes, whose generator polynomials have the same degree, identify DNA sequences with approximately the same probability. Although the probabilities of classification to *Plantae* are slightly different from P, the same effect occurs when we analyze the generator polynomials with the same degree.

Similar results were obtained even for other collections with more specific taxonomic rank. That is, when we restrict the analysis to *Streptomyces* (txid1883), the largest genus of *Actinobacteria*. In this case, two collections $\mathcal{A}$ and $\mathcal{B}$ were analyzed whose sequence lengths are $n = 255$ and $n = 341$, respectively. The dominant codes returned by our algorithm are

$$\begin{aligned} g_{\mathcal{A}}(x) &= x^5 + \beta^2 x^4 + \beta^2 x^3 + x^2 + 1, \\ g_{\mathcal{B}}(x) &= x^5 + \beta^2 x^3 + \beta^2 x^2 + \beta^2 x + 1, \end{aligned} \tag{12}$$

where $g_{\mathcal{A}}(x)$ identifies about 78.27% of the collection $\mathcal{A}$, and $g_{\mathcal{B}}(x)$ identifies 100% of the collection $\mathcal{B}$. Once again, the dominant codes have minimum degree and the perfect code identifies any DNA sequence.

Given a collection of DNA sequences, for example a collection with the following sequences $\{\mathbf{r}_1, \mathbf{r}_2, \ldots, \mathbf{r}_N\}$, the set of BCH codes that identify each one is denoted by $\mathcal{R}_1, \mathcal{R}_2, \ldots, \mathcal{R}_N$, respectively. For different collections, the cardinality dispersion of these sets $\mathcal{R}_i$, where $i = 1, 2, \ldots, N$ is checked by analysis of the boxplot. The boxplot is a statistical representation of data that displays information about the distribution of data through their quartiles for one or more categories of data.

The Fig. 3 illustrates the boxplot for the cardinality of sets $\mathcal{R}_i$ to the collections $\mathcal{A}$ and $\mathcal{B}$ from the previous analysis to *Streptomyces*. One box is assigned for each data category

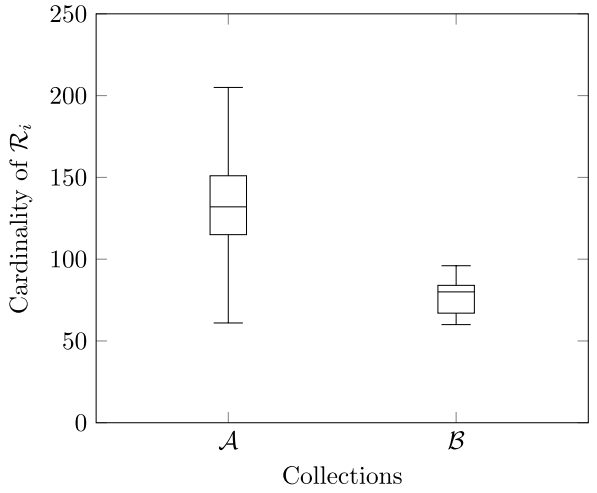


FIGURE 3. Box plot of cardinality of the sets $\mathcal{R}_i$, where $i = 1, 2, \ldots, N$, from collections $\mathcal{A}$ and $\mathcal{B}$ of *Streptomyces* (txid1883). In each collection there are N DNA sequences and the respective $\mathcal{R}_1, \mathcal{R}_2, \ldots, \mathcal{R}_N$ is the set of $\mathcal{C}_{BCH}$ that identify each sequence.

(that is, for $\mathcal{A}$ and $\mathcal{B}$) and for each one is measured the minimum, first quartile, median, third quartile, and maximum value of the cardinality of $\mathcal{R}_i$. In these cases, the sets $\mathcal{R}_i$ have 134.85 codes in average with standard deviation of 79.70 for the collection $\mathcal{A}$, and 77.14 codes in average with standard deviation of 9.37 for the collection $\mathcal{B}$.

Although the standard deviation of the data of collection $\mathcal{A}$ is greater than that of $\mathcal{B}$, the boxplot shape shows that its median is in the middle of the box, and the whiskers are about the same on both sides of the box. Then the cardinality of $\mathcal{R}_i$ is normally distributed and stretched. Whereas for data collection $\mathcal{B}$, the median is closer to the top of the box; then the data is skewed and squeezed. Notice that there are 60 perfect codes in the vector space where the DNA sequences of collection $\mathcal{B}$ are mapped. For this reason, the cardinality of $\mathcal{R}_i$ for sequences in collection $\mathcal{B}$ is at least the number of perfect codes with $n = 341$, that is, 60.

As we said earlier, in general, our algorithm returns $\mathcal{R}_i$ with cardinality more than one. This represents a differential in relation to algorithm in [13]. Furthermore, although the same cardinality is expected as the result of the algorithm in [5], the execution time of our algorithm is shorter.

For all previous cases, the codes, whose generator polynomial has minimum degree, do identify the collections with approximately the same probability; in which case, it is approximately P. Therefore, the dominant codes cannot provide biological classification.

V. CONCLUSION

The algorithm proposed in this paper for identifying DNA sequences as codewords of BCH codes over the finite field $\mathbb{F}_4$ was implemented in SageMath, a free open-source mathematics software system Python-based language. Being a mathematics software, the cost of performing finite field operations is reduced. Our algorithm uses both information theory and abstract algebra approaches to improve several observed computational limitations in [5], [13]. The most important development was replacing the brute force decoding for a PGZ decoder, resulting in a significant reduction in the number of operations.

In previous algorithms, a root finding process [5] or a matrix-check verification [13] was performed $3n + 1$ times (the other three possible nucleotides at each position in the sequence was considered to get the set of neighboring sequences). However, for each run of algorithms, we either count the number of consecutive roots or determine the code using a primitive polynomial. Since our algorithm is restricted to solving (2) within delimited ranges of b and ℓ, the number of operations is reduced and the execution time is shorter. The same idea can be implemented to improve the algorithm whose algebraic structure is the ring of integers $\mathbb{Z}_4$ [6], [14].

Furthermore, we have shown that the probability of a DNA sequence (with odd-length n according to (3)) being identified by a BCH code tends towards the analytical probability of the same code to identify a random vector. Therefore, the dominant codes cannot provide biological information for collections of DNA sequences to allow for alignment-free classification. That is, the DNA sequences mapped to codewords cannot be classified only by these polynomials.

Although this is not a definitive answer to the question of whether or not there is a BCH code underlying DNA sequences, we have checked, by using supervised learning methods, that the SNP in the position pointed out by the code does not influence the classification. We have checked by means of simulations that there is no loss of biological information when we map a DNA sequence to a codeword of some $\mathcal{C}_{BCH}$ and classify it using, for instance, the Kameris method [20].

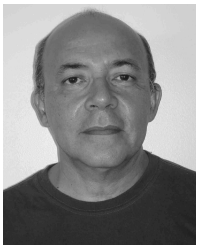
REFERENCES

[1] G. M. Church, Y. Gao, and S. Kosuri, "Next-generation digital information storage in DNA," *Science*, vol. 337, no. 6102, p. 1628, Sep. 2012.
[2] Y. Erlich and D. Zielinski, "DNA fountain enables a robust and efficient storage architecture," *Science*, vol. 355, no. 6328, pp. 950–954, Mar. 2017.
[3] C. T. Clelland, V. Risca, and C. Bancroft, "Hiding messages in DNA microdots," *Nature*, vol. 399, no. 6736, pp. 533–534, Jun. 1999.
[4] M. Beck and R. Yampolskiy, "DNA as a medium for hiding data," *BMC Bioinf.*, vol. 13, no. S12, pp. A23, Jul. 2012.
[5] D. L. Rodríguez-Sarmiento, M. E. Duarte-González, T. Rodríguez-Quiñones, and R. Palazzo, "Procedure for identifying odd-sized nucleotide sequences as codewords of BCH codes over GF(4)," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Paris, France, Jul. 2019, pp. 922–926.
[6] G. L. Hernádez, M. E. Duarte-González, and R. Palazzo, "Identification of odd-sized DNA and mRNA sequences as codewords of BCH codes over Z4," in *Proc. IEEE Int. Conf. Appl. Sci. Adv. Technol. (iCASAT)*, Queretaro, Mexico, Nov. 2019, pp. 1–6.
[7] T. D. Schneider, G. D. Stormo, L. Gold, and A. Ehrenfeucht, "Information content of binding sites on nucleotide sequences," *J. Mol. Biol.*, vol. 188, no. 3, pp. 415–431, Apr. 1986.
[8] G. Battail, "Information theory and error-correcting codes in genetics and biological evolution," in *Introduction to Biosemiotics*, 1st ed. Dordrecht, The Netherlands: Springer, 2007, pp. 299–345.
[9] L. C. B. Faria, A. S. L. Rocha, and R. Palazzo, "Transmission of intracellular genetic information: A system proposal," *J. Theor. Biol.*, vol. 358, pp. 208–231, Oct. 2014.
[10] M. E. Duarte-González, O. Y. Echeverri, J. M. Guevara, and R. Palazzo, "Cyclic concatenated genetic encoder: A mathematical proposal for biological inferences," *Biosystems*, vol. 163, pp. 47–58, Jan. 2018.

[11] L. S. Liebovitch, Y. Tao, A. T. Todorov, and L. Levine, "Is there an error correcting code in the base sequence in DNA?" *Biophys. J.*, vol. 71, no. 3, pp. 1539–1544, Sep. 1996.
[12] G. Rosen, "Examining coding structure and redundancy in DNA," *IEEE Eng. Med. Biol. Mag.*, vol. 25, no. 1, pp. 62–68, Jan. 2006.
[13] L. C. B. Faria, A. S. L. Rocha, J. H. Kleinschmidt, R. Palazzo, and M. C. Silva-Filho, "DNA sequences generated by BCH codes over GF(4)," *Electron. Lett.*, vol. 46, no. 3, pp. 202–203, Feb. 2010.
[14] A. S. L. Rocha, L. C. B. Faria, J. H. Kleinschmidt, R. Palazzo, and M. C. Silva-Filho, "DNA sequences generated by Z4-linear codes," in *Proc. IEEE Int. Symp. Inf. Theory*, Austin, TX, USA, Jun. 2010, pp. 1320–1324.
[15] L. C. B. Faria, A. S. L. Rocha, J. H. Kleinschmidt, M. C. Silva-Filho, E. Bim, R. H. Herai, M. E. B. Yamagishi, and R. Palazzo, "Is a genome a codeword of an error-correcting code?" *PLoS ONE*, vol. 7, no. 5, May 2012, Art. no. e36644.
[16] J. C. Setubal and J. Meidanis, *Introduction to Computational Molecular Biology*. Boston, MA, USA: PWS Publishing Company, 1997, pp. 1–30.
[17] M. M. Brandão, L. Spoladore, L. C. B. Faria, A. S. L. Rocha, M. C. Silva-Filho, and R. Palazzo, "Ancient DNA sequence revealed by error-correcting codes," *Sci. Rep.*, vol. 5, no. 1, p. 12051, Jul. 2015.
[18] R. E. Blahut, *Algebraic Codes for Data Transmission*. New York, NY, USA: Cambridge Univ. Press, 2003.
[19] T. Hoang, C. Yin, and S. S.-T. Yau, "Numerical encoding of DNA sequences by chaos game representation with application in similarity comparison," *Genomics*, vol. 108, nos. 3–4, pp. 134–142, Oct. 2016.
[20] S. Solis-Reyes, M. Avino, A. Poon, and L. Kari, "An open-source k-mer based machine learning tool for fast and accurate subtyping of HIV-1 genomes," *PLoS ONE*, vol. 13, no. 11, Nov. 2018, Art. no. e0206409.
[21] L. He, R. Dong, R. L. He, and S. S.-T. Yau, "A novel alignment-free method for HIV-1 subtype classification," *Infection, Genet. Evol.*, vol. 77, Jan. 2020, Art. no. 104080.
[22] M. Arruda. *DNA BCH*. Accessed: Apr. 26, 2021. [Online]. Available: https://github.com/Milena-Arruda/dna-bch
[23] National Library of Medicine (US), Bethesda, MD, USA. *National Center for Biotechnology Information Nucleotide*. Accessed: Sep. 17, 2020. [Online]. Available: https://www.ncbi.nlm.nih.gov/nuccore
[24] S. S. Pao, I. T. Paulsen, and M. H. Saier, "Major facilitator superfamily," *Microbiol. Mol. Biol. Rev.*, vol. 62, no. 1, pp. 1–34, Mar. 1998.

MILENA M. ARRUDA (Student Member, IEEE) received the B.Sc. and M.Sc. degrees in electrical engineering from the Federal University of Campina Grande (UFCG), Campina Grande, Brazil, in 2015 and 2018, respectively. She is currently pursuing the Ph.D. degree in electrical engineering with UFCG.

Her research interests include information theory and applications, signal processing, genomic signal processing, and engineering education.



FRANCISCO M. DE ASSIS received the B.Sc. and M.Sc. degrees in electrical engineering from the Military Institute of Engineering, Rio de Janeiro, Brazil, in 1984 and 1992, respectively, and the Ph.D. degree from the Pontifical Catholic University of Rio de Janeiro, Rio de Janeiro, in 1994.

He is currently a Professor with the Federal University of Campina Grande, Campina Grande, Brazil. His research interests include coding and information theory.

TACIANA A. DE SOUZA received the B.Sc. degree in mathematics from the Federal University of Paraiba (UFPB), João Pessoa, Brazil, in 2012, the M.Sc. degree in electrical engineering from the Federal Institute of Education, Science and Technology of Paraiba, João Pessoa, in 2015, and the Ph.D. degree in electrical engineering from the Federal University of Campina Grande (UFCG), Campina Grande, Brazil, in 2019.

She is currently a Professor with the Federal Institute of Education, Science and Technology of Paraiba, Cajazeiras, Brazil. Her research interests include information theory and applications, signal processing, algebraic-geometric codes, and cryptography.