A Three-State Model for DNA Protein-Coding Regions

Armando J. Pinho*, *Member, IEEE*, António J. R. Neves, *Student Member, IEEE*, Vera Afreixo, Carlos A. C. Bastos, *Member, IEEE*, and Paulo J. S. G. Ferreira, *Member, IEEE*

***Abstract*—It is known that the protein-coding regions of DNA are usually characterized by a three-base periodicity. In this paper, we exploit this property, studying a DNA model based on three deterministic states, where each state implements a finite-context model. The experimental results obtained confirm the appropriateness of the proposed approach, showing compression gains in relation to the single finite-context model counterpart. Additionally, and potentially more interesting than the compression gain on its own, is the observation that the entropy associated to each of the three base positions of a codon differs and that this variation is not the same among the organisms analyzed.**

***Index Terms*—DNA compression, DNA sequence modeling, finite-context models.**

I. Introduction

In general, the purpose of studying data compression algorithms is twofold. The need for efficient storage and transmission is often the main motivation, but underlying every compression technique there is a model that tries to reproduce as closely as possible the information source to be compressed. This model can have independent interest, as it can shed light on the statistical properties of the source.

DNA data are no exception. We urge to find out efficient methods able to reduce the storage space taken by the impressive amount of genomic data that are continuously being generated. For example, the human genome has about 3×10^9 pairs of bases [1], whereas the genome of the wheat has about 16×10^9 [2]. Notwithstanding, we also desire to know how the code of life works and what structure does it possess. Creating good models for DNA is one of the ways to achieve this knowledge.

DNA is sufficiently determined by a sequence of four molecules called nucleotides (or bases): Adenine, Cytosine, Guanine, and Thymine. These nucleotides can be represented by an alphabet of four letters, $\{A, C, G, T\}$, and can be coded using two bits per base. According to functionality, DNA is subdivided in two parts: Coding and noncoding DNA. The proteins are synthesized based on the coding regions, which are characterized by triplets of bases (codons), each of which codes a protein unit or amino acid according to the genetic code. There are $4^3 = 64$ possible codons, although they represent only 20 amino acids. Hence, the genetic code (which maps codons to amino acids) is redundant. The noncoding regions, also called "junk DNA," are DNA segments that do not comprise code for proteins. However, in eukaryotes, they encode functionally important signals for the regulation of chromosomes [3]. These noncoding regions are interspersed throughout the DNA.

Although the ideas developed in this paper are relevant for DNA-specific data compression, its goals go beyond compression. We seek to better understand and model specific regions of the DNA data, the protein-coding zones. It is known that the protein-coding regions possess specific properties. In particular, they are generally more difficult to compress than the noncoding regions, because the main characteristic used by most DNA compressors, the occurrence of sequence repeats, is not so frequent in coding segments [4]. However, there is a characteristic of coding regions that has not been yet exploited for compression: The three-base periodicity [5], [6].

The aim of this paper is to explore the three-base periodicity property of protein-coding regions in the context of data compression. To achieve this we propose a model composed of three states. Each of the models is selected periodically, according to the three-base period, and each state is implemented using a finite-context model. The comparison of this cyclically varying three-state model with single finite-context model counterparts has shown its ability to better capture the statistics of the data.

Our model has another interesting characteristic, namely, the entropy of each of the three states can be individually estimated, and the results interpreted in reference to the genetic code and the biological characteristics of the organism under study. To see why, recall that most of the amino acids are encoded by more than one codon. For example, there are five amino acids that can be encoded with any of four codons (any of GCA, GCC, GCG, or GCT represent Alanine, for example). Because of the many-to-one property of the genetic code, the entropy associated with the first, second and third nucleotides varies as a function of the distribution of the synonyms, and the variation is contained between two extreme examples: 1) DNA sequences that exhibit such a strong preference for one of the synonyms that none of the others appear in the sequence; 2) sequences that reveal no preference at all for any synonym, so that all synonyms of each amino acid are equally likely to occur in the sequence. In the first case the entropy carried by the third nucleotide is zero, because that nucleotide is completely determined by the previous two. In the second case, the entropy

Manuscript received July 13, 2005; revised June 23, 2006. This work was supported in part by the Fundação para a Ciência e a Tecnologia (FCT). *Asterisk indicates corresponding author.*

*A. J. Pinho is with the Signal Processing Laboratory, DETI/IEETA, University of Aveiro, 3810-193 Aveiro, Portugal (e-mail: ap@det.ua.pt).

A. J. R. Neves, V. Afreixo, C. A. C. Bastos, and P. J. S. G. Ferreira are with the Signal Processing Laboratory, DETI/IEETA, University of Aveiro, 3810-193 Aveiro, Portugal.

Digital Object Identifier 10.1109/TBME.2006.879477

0018-9294/$20.00 © 2006 IEEE

of the third nucleotide is maximum, since any of the possible nucleotides is equally likely to occur. For example, if the four synonyms of Alanine are equally likely, then the probability of a A, C, G or T after GC is $1/4$, and the entropy of the third nucleotide equal to two bits.

The DNA sequences that correspond to organisms fall between these two extremes, and a three-state model should be able to detect entropy variations among nucleotides at positions $3n, 3n+1$, and $3n+2$. Our results confirm that the entropy of the three states differs, and that the variation is not the same among the organisms analyzed, facts that may have independent interest. The organism-dependence could be due to several reasons. Warmer environment organisms may have a C/G content higher than A/T, because $C-G$ bonds are harder to break; and codon preference is known to vary significantly across different species. In fact, the level of expression of a foreign protein in a particular expression system can be enhanced by matching the codon frequency to that of the host system (see, for example, [7]).

II. Three-Base Periodicity

The protein-coding regions of DNA often exhibit periodicities, the strongest of which is associated with the period three. This periodicity is probably due to the structure of the genetic code itself, which consists of code words of length three (codons).

According to [6], the existence of the period three can be traced back to the work of Trifonov *et al.* [5], published more than two decades ago. Since then, this finding has been used mostly in the difficult task of locating the protein-coding regions in DNA (see, for example, [8]–[12]). This application motivated the development of fast algorithms for calculating the spectral coefficient of interest [13].

The spectrum of a symbolic sequence can be defined in several ways. The simplest solution would be to map each symbol to a number, but this is far from ideal because the results would depend on the particular labeling adopted (see [14] for an example). The symbolic autocorrelation concept provides one way of obtaining results that are independent of any symbolic-to-numeric labeling. Given the sequence of symbols x_i, its autocorrelation is the numeric sequence r_k

$$r_k = \sum_{i=0}^{n-1} d(x_i, x_{i+k})$$

where for any two symbols a and b

$$d(a,b) = \begin{cases} 1, & a = b \\ 0, & \neq b \end{cases}.$$

The discrete Fourier transform (DFT) of this autocorrelation is the spectrum of the symbolic data. In this case no hypothesis are necessary regarding the symbols, except the ability to check if they are equal or distinct.

A DNA sequence can also be represented using four indicator sequences, that is, zero/one sequences that indicate the positions of the symbols in the symbolic sequence. The spectrum obtained from the DFT of the symbolic autocorrelation turns out to be equal to the sum of the squared DFTs of the four indicator sequences. From the computational point of view, the latter procedure is preferable. The concept of spectral envelope provides yet another approach to the Fourier analysis of symbolic data. The connections between these approaches and others are discussed in [14], which gives a unified perspective of the methods that can be used to perform spectral analysis of symbolic data, such as DNA data.

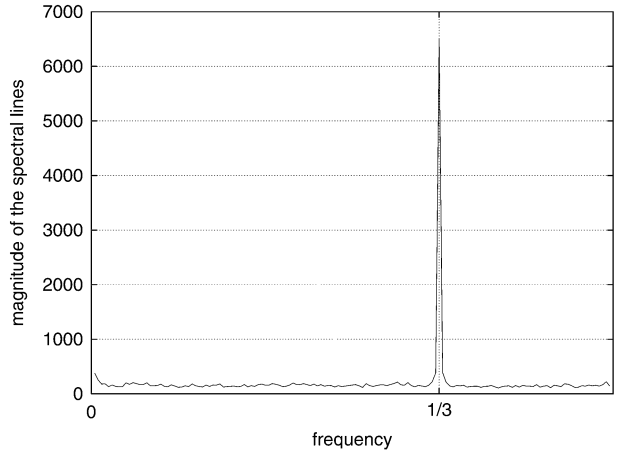

Fig. 1. The spectrum of a DNA sequence, showing the period three. Data: human chromosome 22.

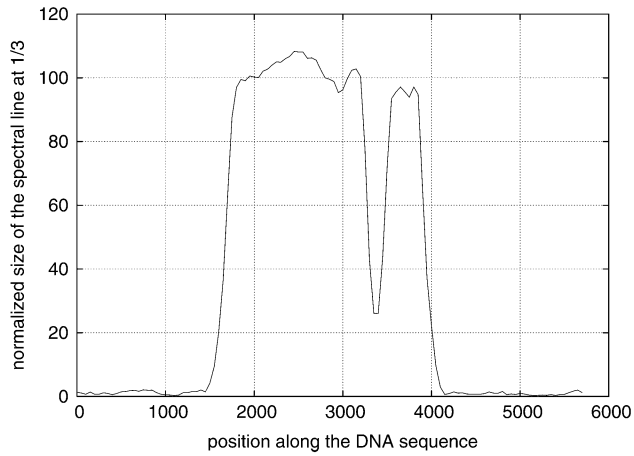

Fig. 2. The evolution of the magnitude of the spectral line corresponding to the period three along a segment of DNA. Data: human chromosome 22.

Figs. 1 and 2 illustrate the three-base periodicity of a DNA sequence and its variation along the sequence. They were obtained using a sliding window approach. The process started with the computation of the spectrum of $N = 300$ nucleotides of a segment of the human chromosome 22, starting with the nucleotide at position $M = 0$. After each computation, the position M was incremented by 50, and another spectrum was computed. This process was repeated 115 times, corresponding to 6000 nucleotides. The 115 spectra were averaged, and Fig. 1 shows the result. The sharp peak at frequency $1/3$ is associated with a period of length three. Fig. 2 is intended to show how the strength of this peak varies over the 115 spectra. The quantity plotted is the size of the peak at $1/3$ divided by the average value of the spectral lines. A value of 50, for example, corresponds to

a peak value that is 50 times larger than the spectral average. The three-base periodicity is not equally strong over the segment under analysis, possibly due to the presence of noncoding regions.

The three-base periodicity in fact has been accepted as a usual property of DNA coding regions. Its usefulness for gene-detection has been noted and applied in practice, and recently a very fast algorithm for computing the size of the spectral coefficient at $1/3$ has been proposed [13]. That work also studies the connection between the spectrum of a segment of DNA and the nucleotide distribution along that segment, and gives necessary and sufficient conditions for a spectral coefficient to have a prescribed magnitude.

The three-state model proposed in this paper explores this well known periodicity. Our results confirm that it is worthwhile to explore its presence for data compression and modeling. To achieve this we use a three-state model, and we have found that the entropies of the three states varies across organisms. This other finding may be a new result with biological significance.

III. DNA Compression Methods

Although we are not proposing a complete technique for DNA compression, it is nevertheless interesting to give a brief overview of the main approaches that have been taken in this direction. Moreover, this overview also serves to show that the three-base periodicity has not been used previously in the context of DNA data compression.

The first method designed specifically for compressing DNA sequences was proposed by Grumbach *et al.* in 1993 and was named *Biocompress* [15]. This technique is based on the sliding window algorithm proposed by Ziv and Lempel, also known as LZ77 [16]. According to this universal data compression technique, a subsequence is encoded using a reference to an identical subsequence that occurred in the past. The longer the matching sequences, the higher the compression ratios attained. The *Biocompress* algorithm makes use of a characteristic usually found in DNA sequences which is the occurrence of complemented inverted repeats (also known as complementary palindromes). These are subsequences that are both reversed and complemented $(A \leftrightarrow T, C \leftrightarrow G)$, and that are also searched for referencing by the encoding algorithm. Besides this DNA-specific characteristic, *Biocompress* is able to commute between the LZ-based compression mode and a transparent, uncompressed, mode. Switching between modes is done using forward adaptation, according to a shortest code criterion. The second version of *Biocompress*, *Biocompress-2*, introduced a third mode of operation, based on a second order finite-context arithmetic encoder [17]. Again, during compression, the mode given the shortest code is the one chosen for coding that subsequence.

Rivals *et al.* proposed another compression technique based on exact repetitions, *Cfact*, which relies on a two-pass strategy [18], [19]. In the first pass, the complete sequence is parsed using a suffix tree. This parsing phase produces a list of the longest repeating subsequences that have a potential coding gain. In the second pass, those subsequences are encoded using references to the past, whereas the rest of the symbols are left uncompressed. Contrary to *Biocompress, Cfact* does not try to exploit the potential redundancy of complementary palindromes. In fact, *Cfact* does not incorporate any particularity of DNA sequences and hence it can be considered a general purpose compression algorithm.

The idea of using repeating subsequences as a means of achieving compression was also exploited by Chen *et al.* [20], [21]. However, in this case, the authors proposed a generalization of this strategy such that approximate repeats of subsequences and of complementary palindromes could also be handled. In order to be lossless, the algorithm, named *GenCompress*, has to be able to reconstruct the original subsequence based on an approximation occurring in the past. This can be done using operations such as replacements, insertions and deletions. One version of the algorithm, *GenCompress-1*, used only replacement operations. A second version, *GenCompress-2*, besides replacements, could also perform deletion and insertion operations in the subsequence. However, from the experimental results presented by the authors, both schemes presented virtually identical compression performance, which seems to indicate that replacements should be enough. As in *Biocompress, GenCompress* includes a mechanism for deciding if it is worthwhile to encode the subsequence under evaluation using the substitution-based model. If not, it falls back to a second mode of operation based on an order-2 finite-context arithmetic encoder.

A further modification of *GenCompress* led to a two-pass algorithm, *DNACompress*, relying on a separated tool for approximate repeat searching, *PatternHunter*, [22]. Besides providing additional compression gains, *DNACompress* is considerably faster than *GenCompress*.

Before the publication of *DNACompress*, a technique based on context tree weighting (CTW) and LZ-based compression, $\mathrm{CTW}+\mathrm{LZ}$, was proposed by Matsumoto *et al.* [23]. Basically, long repeating subsequences or complementary palindromes, exact or approximate, are encoded by a LZ-type algorithm, whereas short subsequences are compressed using CTW (as in the other cases, this acts as a fall back mechanism). Compared with *GenCompress*, $\mathrm{CTW}+\mathrm{LZ}$ provided a slight compression gain [23]. However, the time needed to run the program on relatively large sequences showed to be prohibitive [22].

The paradigm of exact matching was addressed again recently by Manzini *et al.* [24]. One of the aims of their work was to design a fast, although competitive, DNA encoder. One of the key problems of compression techniques based on subsequence matching is the time taken by the search operation. Manzini *et al.* addressed this drawback by proposing a solution based on fingerprints. Basically, in this approach, the possibility of matching small subsequences is waived in exchange for increased speed. Their argument is that in DNA we are most interested in matching long subsequences (or reverse complements) and, therefore, the proposed method is viable. In fact, most often, the compression performance of Manzini's method is lower than that offered by *DNACompress*, but it is considerably faster. Like other methods already discussed, this technique also uses fall back mechanisms for the zones where matching fails. In this case, finite-context arithmetic coding of order two (named method *Dna2* by Manzini *et al.*) or three (*Dna3*) play this role.

Tabus *et al.* proposed a DNA sequence compression method based on normalized maximum likelihood discrete regression for approximate block matching [25]. This work, later improved for compression performance and speed [26], encodes fixed-size blocks by referencing a previously encoded subsequence with minimum Hamming distance. Only replacement operations are allowed for editing the reference subsequence which, therefore, always have the same size as the block, although may be located in an arbitrary position inside the already encoded sequence. As others, this compression method (*GeNML*) incorporates fall back modes of operation for avoiding degradation of performance when the main algorithm fails. In this case, two additional modes are included: a finite-context arithmetic encoder of order one and a transparent mode in which the block passes uncompressed (i.e., each base is represented directly with a binary word of two bits).

More recently, Behzadi *et al.* proposed a new algorithm, *DNAPack*, which uses the Hamming distance (i.e., relies only on substitutions) for the repeats and complementary palindromes, and either CTW or order-2 arithmetic coding for nonrepeating regions [27]. Moreover, *DNAPack* relies on dynamic programming techniques for choosing the repeats, instead of greedy approaches as others do. Tested in a number of small sequences, on average, *DNAPack* provided compression gain in relation to *DNACompress* [27].

IV. The Models

The overview presented in Section III allow us to conclude that most of the effort spent by current DNA compressors is in the task of finding good exact or approximate repeats of subsequences or of their inverted complements. No doubt, this approach has proved to give good returns in terms of compression gains. However, in our opinion, other potentially important aspects, such as the particular characteristics of protein-coding regions, also deserve attention. In this paper, we address this issue. Using finite-context models and a periodicity property, we study the compressibility of protein-coding regions.

The three-base periodicity suggests an underlying statistical model that might be driven by three different, although related, information sources. In this paper, we evaluate this conjecture, using a setup based on three states, each one composed of a finite-context model, and where switching between states follows the three-base periodicity.

A. Finite-Context Models

Consider an information source that generates symbols, s, from an alphabet $\mathcal{A}$. At time t, the sequence of outcomes generated by the source is $x^t = x_1 x_2 \cdots x_t$. A finite-context model (see Fig. 3) of an information source assigns probability estimates to the symbols of the alphabet, according to a conditioning context computed over a finite and fixed number, M, of past outcomes (order-M finite-context model) [28]–[30]. At time t, we represent these conditioning outcomes by $c^t = x_{t-M+1}, \ldots, x_{t-1}, x_t$. The number of conditioning states of the model is $|\mathcal{A}|^M$, dictating its complexity (or model cost).

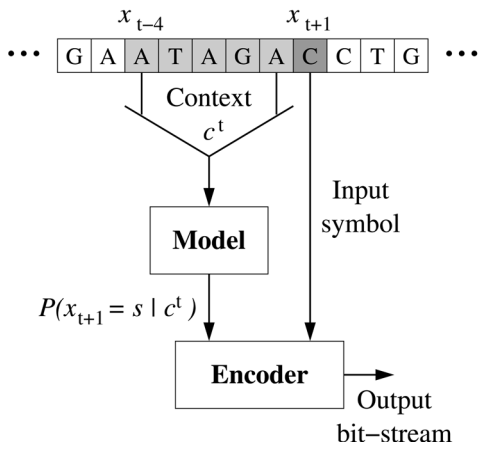

Fig. 3. Finite-context model: the probability of the next outcome, x_{t+1}, is conditioned by the M last outcomes. In this example, $M = 5$.

TABLE I
Simple Example Illustrating How Finite-Context Models Are Implemented. The Rows of the Table Represents a Probability Model at a Given Instant t. In This Example, the Particular Model That Is Chosen for Encoding a Symbol Depends on the Last Five Encoded Symbols (Order-5 Context)

Context, c^t	$n(A, c^t)$	$n(C, c^t)$	$n(G, c^t)$	$n(T, c^t)$	$\sum_{a \in \mathcal{A}} n(a, c^t)$
$AAAAA$	23	41	3	12	79
$AAAAC$	16	6	21	15	58
$AAAAG$	19	30	10	50	109
$AAAAT$	34	47	9	31	121
$AAACA$	36	17	14	15	82
⋮	⋮	⋮	⋮	⋮	
$TTTTT$	8	2	18	11	39

In practice, the probability that the next outcome, x_{t+1}, is $s \in \mathcal{A}$, is obtained using the following estimator:

$$P(x_{t+1} = s \,|\, c^t) = \frac{n(s, c^t) + \delta}{\sum_{a \in \mathcal{A}} n(a, c^t) + |\mathcal{A}|\delta}$$

where $n(s, c^t)$ represents the number of times that, in the past, the information source generated symbol s having c^t as the conditioning context. The parameter $\delta > 0$, besides allowing fine tuning the estimator, avoids generating zero probabilities when a symbol is encoded for the first time. In our case we used $\delta = 1$, which can be seen as an initialization of all counters to one. These counters are updated each time a symbol is encoded. Since the context template is causal, the decoder is able to reproduce the same probability estimates without needing additional information.

Table I shows an example of how a finite-context is typically implemented. In this example, an order-5 finite-context model is presented. Each row represents a probability model that is used to encode a given symbol according to the last encoded symbols (five in this example). Therefore, if the last symbols were "$AAACA$", i.e., $c^t = AAACA$, then the model communicates the following probability estimates to the arithmetic encoder: $P(A|AAACA) =$

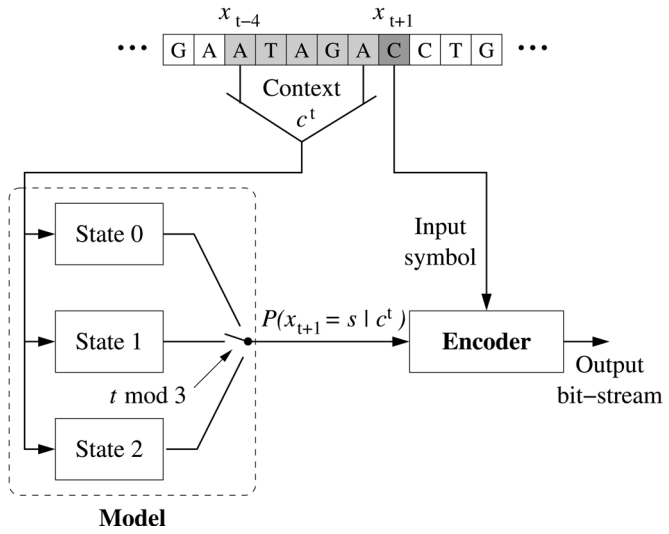

Fig. 4. Three-state model, exploiting the three-base periodicity of the DNA protein-coding regions. In this case, the probability of the next outcome, x_{t+1}, is conditioned both by the M last outcomes and by the value of $(t \bmod 3)$.

$36/82, P(C|AAACA) = 17/82, P(G|AAACA) = 14/82$, and $P(T|AAACA) = 15/82$.

The block denoted "Encoder" in Fig. 3 is an arithmetic encoder. It is well known that practical arithmetic coding generates output bit-streams with average bit-rates almost identical to the entropy of the model [28]–[30]. In our case, the theoretical bit-rate average (entropy) of the model after encoding N symbols is given by

$$H_N = -\frac{1}{N}\sum_{t=0}^{N-1} \log_2 P(x_{t+1} = s \,|\, c^t) \quad \text{bps} \qquad (1)$$

where "bps" stands for "bits per symbol." Since we are dealing with DNA bases, instead of using the generic "bps" measure we use "bpb," which stands for "bits per base." Recall that the entropy of any sequence of four symbols is limited to two bps, a value that is achieved when the symbols are independent and equally likely.

B. The Three-State Model

Fig. 4 shows the model addressed in this paper. It differs from the finite-context model displayed in Fig. 3 by the inclusion of three internal states. Each state is selected periodically, according to a three-base period, and comprises a finite-context model, similar to the one presented in Fig. 3.

With this model, probabilities depend not only on the M last outcomes, but also on the value of $(t \bmod 3)$, which is used for state selectivity. In this case, the probability estimator is given by

$$P(x_{t+1} = s \,|\, c^t) = \frac{n_i(s, c^t) + \delta}{\sum_{a \in \mathcal{A}} n_i(a, c^t) + |\mathcal{A}|\delta}, \quad i = t \bmod 3$$

i.e., three different sets of counters are used, one for each state. Moreover, only the counters associated with the chosen state are updated. It is worth noting that, in order to be able to operate, this model does not require the knowledge of the correct reading frame. However, once having chosen a particular position for starting the model, the corresponding reading frame should be maintained, otherwise the statistics will become mixed and the model will not work properly. Notwithstanding, if we intend to determine the entropies associated with each of the three base positions inside the codons, we need to know which base position corresponds to each state of the model. For the cases reported in t Section V, we always started the model at the beginning of a codon, implying that state zero corresponds to the first base position of the codon, state one to the second base position and state two to the third base position.

V. Experimental Results

The experimental results that we present in this section have been obtained using data from the "`ffn`" files collected form `ftp://ftp.ncbi.nlm.nih.gov/genomes/`. In our study, we included data from the following organisms: *Haemophilus influenzae, Escherichia coli K12, Schizosaccharomyces pombe, Saccharomyces cerevisiae*, and *Arabidopsis thaliana*. Some of the gene descriptions included in these files contain a few symbols not belonging to the $\{A, C, G, T\}$ alphabet. Moreover, in a few cases, the gene length is not multiple of three, causing an inconsistency in the codon structure. Therefore, in order to avoid undesirable interferences motivated from these particularities, the files have been checked and all genes containing spurious symbols or not having length divisible by three have not been included for testing.

Table II presents the compression results, in bits per base (bpb), obtained using four coding approaches: the three-state finite-context model, the single-state finite-context model, *DNACompress* [22] (referred to as "*DnaC*" in the table) and Manzini's method *Dna3* [24]. The reason for choosing *DNACompress* and *Dna3*, besides from being state-of-the-art techniques, was the availability of implementations. Note that, although the theoretical bit-rate given by (1) is expected to be very close to the real bit-rate, all compression values presented in Table II correspond to real code generated by an arithmetic encoder.

For each sequence, the depth of the context, M, was chosen in order to provide the best possible compression. This search has been done for $M = \{1, 2, \ldots, 7\}$. Although we did not try to establish a relation that could provide us a good guess for M, we nevertheless note that, generally, the larger the sequence the larger should be the value of M. This trend can be observed in the data presented in Table II.

We can see from the results of Table II that the three-state finite-context model of Section IV–B is always better than the single-state model of Section IV-A (compare the totals given in columns 8 and 10 of the table), confirming our assumption about the potential advantage of using the three-base periodicity for compression purposes. The three-state model also attained better compression results than the state-of-the-art DNA compression techniques included in the tests, when applied to *Haemophilus influenzae, Escherichia coli K12*, and *Schizosaccharomyces pombe*. For the *Saccharomyces cerevisiae*, the results are mixed, although for more than half of the chromosomes the three-state model

TABLE II

COMPRESSION RESULTS, IN BITS PER BASE (BPB), OBTAINED FOR THE FIVE ORGANISMS TESTED: *HAEMOPHILUS INFLUENZAE, ESCHERICHIA COLI K12, SCHIZOSACCHAROMYCES POMBE, SACCHAROMYCES CEREVISIAE* AND *ARABIDOPSIS THALIANA*. FOR THE THREE-STATE MODEL, COLUMNS "BPB0," "BPB1," AND "BPB2" INDICATE THE COMPRESSION RATES ATTAINED BY EACH OF THE THREE STATES (CORRESPONDING TO THEIR RESPECTIVE SUBSEQUENCE, I.E., TO ONE THIRD OF THE WHOLE SEQUENCE), WHEREAS THE "BPB" COLUMN INDICATES OVERALL COMPRESSION RATE. COLUMNS "M" INDICATE THE ORDER OF THE FINITE-CONTEXT MODEL, WHICH WAS ALWAYS THE BEST POSSIBLE. COLUMNS "*DNAC*" AND "*DNA3*" SHOW COMPRESSION RESULTS USING, RESPECTIVELY, *DNACOMPRESS* AND MANZINI'S *DNA3* METHOD

Haemophilus influenzae												
Reference	Sequence	Bases	Three-state					Single-state		*DnaC*	*Dna3*	
			M	bpb0	bpb1	bpb2	bpb	M	bpb	bpb	bpb	Repeats
GI:16271976	—	1 505 271	4	1.918	1.834	1.684	**1.812**	5	1.889	1.902	1.895	0.9%

Escherichia coli K12												
Reference	Sequence	Bases	Three-state					Single-state		*DnaC*	*Dna3*	
			M	bpb0	bpb1	bpb2	bpb	M	bpb	bpb	bpb	Repeats
GI:49175990	—	4 083 231	5	1.897	1.898	1.750	**1.848**	6	1.917	1.920	1.913	1.3%

Schizosaccharomyces pombe												
Reference	Sequence	Bases	Three-state					Single-state		*DnaC*	*Dna3*	
			M	bpb0	bpb1	bpb2	bpb	M	bpb	bpb	bpb	Repeats
GI:19113674	Chr-I	2 996 109	4	1.961	1.884	1.820	**1.889**	4	1.939	1.918	1.921	1.3%
GI:19111836	Chr-II	2 399 394	4	1.962	1.887	1.818	**1.889**	4	1.940	1.915	1.916	1.7%
GI:19075172	Chr-III	1 169 991	3	1.961	1.889	1.833	**1.895**	4	1.943	1.925	1.930	1.2%

Saccharomyces cerevisiae												
Reference	Sequence	Bases	Three-state					Single-state		*DnaC*	*Dna3*	
			M	bpb0	bpb1	bpb2	bpb	M	bpb	bpb	bpb	Repeats
GI:50593113	Chr-I	143 157	2	1.937	1.882	1.909	1.911	3	1.954	**1.884**	1.910	3.4%
GI:50593115	Chr-II	605 184	3	1.936	1.869	1.886	**1.897**	3	1.942	1.912	1.918	1.7%
GI:42759850	Chr-III	217 332	2	1.946	1.874	1.908	**1.911**	3	1.951	1.918	1.923	1.8%
GI:50593138	Chr-IV	1 129 605	3	1.931	1.856	1.882	1.890	4	1.936	**1.846**	1.853	5.0%
GI:7276232	Chr-V	391 086	3	1.935	1.872	1.894	1.901	3	1.947	**1.883**	1.894	3.3%
GI:42742172	Chr-VI	183 702	2	1.938	1.863	1.904	**1.904**	3	1.949	1.932	1.939	1.0%
GI:50593213	Chr-VII	784 707	3	1.935	1.861	1.882	**1.893**	3	1.939	1.897	1.902	2.2%
GI:50882583	Chr-VIII	402 792	3	1.938	1.873	1.896	**1.903**	3	1.946	1.907	1.915	2.1%
GI:6322016	Chr-IX	310 041	3	1.938	1.869	1.900	**1.903**	3	1.947	1.933	1.942	0.9%
GI:42742252	Chr-X	557 103	3	1.935	1.866	1.892	**1.899**	3	1.943	1.907	1.914	1.8%
GI:50593424	Chr-XI	478 620	3	1.935	1.855	1.893	**1.895**	3	1.940	1.938	1.942	0.3%
GI:42742286	Chr-XII	784 695	3	1.936	1.862	1.893	1.898	3	1.942	**1.863**	1.872	4.1%
GI:44829554	Chr-XIII	693 291	3	1.934	1.859	1.889	1.894	3	1.940	**1.886**	1.892	3.0%
GI:50593505	Chr-XIV	576 585	3	1.937	1.869	1.893	**1.900**	3	1.944	1.930	1.934	0.9%
GI:42742309	Chr-XV	785 568	3	1.937	1.865	1.887	**1.897**	3	1.941	1.901	1.917	2.2%
GI:50593503	Chr-XVI	687 666	3	1.937	1.862	1.887	1.896	3	1.941	1.889	**1.880**	3.6%
GI:6226515	MT	24 429	2	1.814	1.767	1.305	1.643	3	1.747	**1.466**	1.511	16.6%

Arabidopsis thaliana												
Reference	Sequence	Bases	Three-state					Single-state		*DnaC*	*Dna3*	
			M	bpb0	bpb1	bpb2	bpb	M	bpb	bpb	bpb	Repeats
GI:42592260	Chr-I	9 595 494	5	1.925	1.904	1.882	1.904	6	1.939	**1.725**	1.743	12.0%
GI:30698031	Chr-II	5 474 178	4	1.926	1.906	1.886	1.906	6	1.942	**1.710**	1.737	12.0%
GI:30698537	Chr-III	7 183 863	5	1.925	1.904	1.886	1.905	6	1.941	**1.736**	1.762	10.8%
GI:30698542	Chr-IV	5 572 038	4	1.926	1.905	1.888	1.906	6	1.942	**1.708**	1.740	12.1%
GI:30698605	Chr-V	8 462 424	5	1.924	1.902	1.883	1.903	6	1.939	**1.736**	1.759	10.9%

TABLE III

THIS TABLE PROVIDES AN EXAMPLE, USING THE *SCHIZOSACCHAROMYCES POMBE* ORGANISM, OF HOW CHANGING PARAMETER M (THE ORDER OF THE FINITE-CONTEXT MODEL) AROUND THE OPTIMAL VALUE, M_{opt}, AFFECTS THE COMPRESSION VALUES (IN BITS PER BASE). THE VALUES INSIDE PARENTHESIS INDICATE THE VALUE OF M

Schizosaccharomyces pombe							
Sequence	Bases	Three-state			Single-state		
		$M_{\text{opt}}-1$	M_{opt}	$M_{\text{opt}}+1$	$M_{\text{opt}}-1$	M_{opt}	$M_{\text{opt}}+1$
Chr-I	2 996 109	1.889 (3)	1.889 (4)	1.893 (5)	1.941 (3)	1.939 (4)	1.939 (5)
Chr-II	2 399 394	1.890 (3)	1.889 (4)	1.894 (5)	1.941 (3)	1.940 (4)	1.940 (5)
Chr-III	1 169 991	1.897 (2)	1.895 (3)	1.896 (4)	1.944 (3)	1.943 (4)	1.947 (5)

performed better. Finally, for the *Arabidopsis thaliana*, the three-state finite-context model falls short in comparison to *DNACompress* and *Dna3*.

In Table III, we present, for the case of *Schizosaccharomyces pombe*, the variation of the bit-rate when M varies somewhat around the optimal value, M_{opt}. From this example, it can be observed that the variation of the bit-rate is usually small for variations of M of ± 1 around M_{opt}. This shows that small variations of M are not expected to originate large variations in the performance of the method.

We note that the approach proposed in this paper is not a complete DNA compression technique: It explores only one property of certain regions of DNA, the protein-coding regions, and tries to show that their properties deserve careful study. The results obtained using the compression methods included in Table II are intended to put ours into perspective. In fact, those compression methods are much more sophisticated, and explore important features such as the coding of repetitions, whereas the presented encoder uses only the three-base periodicity.

VI. Discussion

The results of Table II show clearly that, from a compression point of view, it is advantageous to take into account the three-base periodicity of the DNA protein-coding regions. Moreover, for some of the organisms tested, this approach provided better results than state-of-the-art DNA compression techniques. Since *DNACompress* and *Dna3* rely strongly on subsequence repetition (exact or approximate), as in fact most of the DNA compression techniques do, we believe that the reason why in some sequences the three-state approach is better is because in those sequences repetitions are relatively rare.

Obviously, the opposite also holds, i.e., without the support of complementing techniques, the three-state model falls short in sequences containing repetitions that are better exploited by other algorithms. This is precisely what happens with the *Arabidopsis thaliana* genome. In fact, it is common for plants to have a great amount of DNA repetition [31]. Particularly, it is know that the *Arabidopsis thaliana* plant has extensive gene repetition [32].

Our view is supported by the data presented in the last column of Table II, where the percentages of bases that have been encoded by *Dna3* using references to past subsequences are indicated. As can be observed, for the *Arabidopsis thaliana* these percentages are over 10%. More generally, when the percentage of bases encoded using the repeat strategy is over 3%, the three-state model alone cannot capture all relevant structure of the data. Therefore, if compression performance is the goal, then the three-state model has to be complemented with a method able to exploit these repetitive patterns.

Another interesting observation that we can extract from Table II is how the bit-rates of the three states compare. For the results presented, state 0 corresponds to the first base of the codon, state 1 to the second and state 3 to the last base of the codon. Therefore, the values denoted by "bpb0", "bpb1," and "bpb2" indicate the average number of bits required by the encoder to represent, respectively, the first, the second and the third bases of the codon. For the *Haemophilus influenzae, Schizosaccharomyces pombe*, and *Arabidopsis thaliana*, the first base is the most hard to compress, then comes the second, and, finally, the third. From a point of view of information theory, this means that the first base is the one conveying the largest fraction of the codon information, followed by the second base.

This observation is not surprising, due to the degenerated nature of the genetic code. In fact, most of the amino-acids can be represented by more than one triplet and, for some of them, the third base is even irrelevant. In our opinion what is surprising is finding out that for all chromosomes of the *Saccharomyces cerevisiae* the second base seems to carry less information than the third one. Moreover, although marginally, for the *Escherichia coli K12* the second base seems to carry, at least, as much information as the first base. Currently, we are unable to provide an explanation for this behavior, but we believe that it justifies further consideration and analysis also from the biological perspective. In fact, as discussed in Section II, this organism-dependence is not totally unexpected: Codon preference may vary dramatically across species, and warmer environment organisms have a C/G content higher than A/T, because it is harder to break a $C-G$ bond, a fact that may also induce preferences in the sets of synonyms.

A final note concerning the computational complexity of the three-state model. The memory requirements are directly related to the implementation of the finite-context model. From the example presented in Table I, it can be observed that to implement an order-M finite-context model for an alphabet of size $|\mathcal{A}|$ we need to store $|\mathcal{A}|^M$ probability models (i.e., the number of lines of Table I) which require about $|\mathcal{A}|^{M+1}$ counters. Therefore, for example, if $M=6$, this implies about 4^7 counters that can be stored in about 64 Kbytes (considering four bytes for each counter). Regarding the computational time, we provide some indicative values because our current implementation has not been optimized for speed. For example, for the largest sequence of the test, i.e., chromosome I of *Arabidopsis thaliana*, our implementation requires, for a given M, about 12 s on a P4 Mobile@2 GHz with 512 Mbytes of RAM. For the same sequence, *DnaC* required 57 s whereas *Dna3* needed 14 s.

VII. Conclusion

In this paper, we addressed a characteristic of DNA protein-coding regions known as the three-base periodicity. We studied the compression performance of a three-state finite-context model and concluded that, in those regions, it always behaves better than the single-state counterpart. For some organisms it also surpasses state-of-the-art DNA compression techniques.

Probably, the most relevant feature of this model is the possibility of analyzing how information is distributed among the three bases of the codon. Although, in this paper, we have not tried to go deeper into this subject, we pointed out some aspects that we do believe deserve further study. Particularly, the observation that for the *Saccharomyces cerevisiae* the third base of the codon carries, on average, more information than the second base, appears to be an interesting finding.

Acknowledgment

The author would like to thank X. Chen and G. Manzini for providing implementations of their DNA compression algorithms.

References

[1] L. Rowen, G. Mahairas, and L. Hood, "Sequencing the human genome," *Science*, vol. 278, pp. 605–607, Oct. 1997.
[2] C. Dennis and C. Surridge, "A. *thaliana* genome," *Nature*, vol. 408, p. 791, Dec. 2000.
[3] M. Z. Ludwig, "Functional evolution of noncoding DNA," *Curr. Opin. Genetics Develop.*, vol. 12, no. 6, pp. 634–639, 2002.
[4] N. V. Dokholyan, S. V. Buldyrev, S. Havlin, and H. E. Stanley, "Distribution of base pair repeats in coding and noncoding DNA sequences," *Phys. Rev. Lett.*, vol. 79, no. 25, pp. 5182–5185, Dec. 1997.

[5] E. N. Trifonov and J. L. Sussman, "The pitch of chromatin DNA is reflected in its nucleotide sequence," *Proc. Nat. Acad. Sci. USA*, vol. 77, no. 7, pp. 3816–3820, Jul. 1980.
[6] P. P. Vaidyanathan, "Genomics and proteomics: A signal processor's tour," *IEEE Circuits Syst. Mag.*, vol. 4, no. 4, pp. 6–29, Fourth Quarter 2004.
[7] E. M. Slimko and H. A. Lester, "Codon optimization of *Caenorhabditis elegans* GluCl ion channel genes for mammalian cells dramatically improves expression levels," *J. Neurosci. Meth.*, vol. 124, pp. 75–81, 2003.
[8] J. W. Fickett, "Recognition of protein coding regions in DNA sequences," *Nucleic Acids Res.*, vol. 10, no. 17, pp. 5303–5318, 1982.
[9] V. R. Chechetkin and A. Y. Turygin, "Size-dependence of three-periodicity and long-range correlations in DNA sequences," *Phys. Lett., A*, vol. 199, pp. 75–80, 1995.
[10] S. Tiwari, S. Ramachandran, A. Bhattacharya, S. Bhattacharya, and R. Ramaswamy, "Prediction of probable genes by Fourier analysis of genomic sequences," *Bioinformatics*, vol. 13, pp. 263–270, 1997.
[11] B. Issac, H. Singh, H. Kaur, and G. P. S. Raghava, "Locating probable genes using Fourier transform approach," *Bioinformatics*, vol. 18, no. 1, pp. 196–197, 2002.
[12] D. Kotlar and Y. Lavner, "Gene prediction by spectral rotation measure: A new method for identifying protein-coding regions," *Genome Res.*, vol. 13, pp. 1930–1937, 2003.
[13] V. Afreixo, P. J. S. G. Ferreira, and D. Santos, "Spectrum and symbol distribution of nucleotide sequences," *Phys. Rev., E*, vol. 70, p. 031910, 2004.
[14] ——, "Fourier analysis of symbolic data: A brief review," *Digital Signal Process.*, vol. 14, no. 6, pp. 523–530, Nov. 2004.
[15] S. Grumbach and F. Tahi, "Compression of DNA sequences," in *Proc. Data Compression Conf. (DCC-93)*, Snowbird, UT, 1993, pp. 340–350.
[16] J. Ziv and A. Lempel, "A universal algorithm for sequential data compression," *IEEE Trans. Inf. Theory*, vol. IT-23, pp. 337–343, May 1977.
[17] S. Grumbach and F. Tahi, "A new challenge for compression algorithms: Genetic sequences," *Inf. Process. Manage.*, vol. 30, no. 6, pp. 875–886, 1994.
[18] E. Rivals, J.-P. Delahaye, M. Dauchet, and O. Delgrange, A guaranteed compression scheme for repetitive DNA sequences LIFL, Université des Sciences et Technologies de Lille, Tech. Rep. IT-95-285, Nov. 1995.
[19] ——, "A guaranteed compression scheme for repetitive DNA sequences," in *Proc. Data Compression Conf. (DCC-96)*, Snowbird, UT, 1996, p. 453.
[20] X. Chen, S. Kwong, and M. Li, "A compression algorithm for DNA sequences and its applications in genome comparison," in *Proc. 10th Workshop Genome Informatics 1999:* , K. Asai, S. Miyano, and T. Takagi, Eds., Tokyo, Japan, 1999, pp. 51–61.
[21] ——, "A compression algorithm for DNA sequences," *IEEE Eng. Med. Biol. Mag.*, vol. 20, pp. 61–66, Jul.–Aug. 2001.
[22] X. Chen, M. Li, B. Ma, and J. Tromp, "DNACompress: Fast and effective DNA sequence compression," *Bioinformatics*, vol. 18, no. 12, pp. 1696–1698, 2002.
[23] T. Matsumoto, K. Sadakane, and H. Imai, "Biological sequence compression algorithms," in *Proc. 11th Workshop Genome Informatics 2000*, A. K. Dunker, A. Konagaya, S. Miyano, and T. Takagi, Eds., Tokyo, Japan, 2000, pp. 43–52.
[24] G. Manzini and M. Rastero, "A simple and fast DNA compressor," *Software—Practice Experience*, vol. 34, pp. 1397–1411, 2004.
[25] I. Tabus, G. Korodi, and J. Rissanen, "DNA sequence compression using the normalized maximum likelihood model for discrete regression," in *Proc. Data Compression Conf. (DCC-2003)*, Snowbird, UT, 2003, pp. 253–262.
[26] G. Korodi and I. Tabus, "An efficient normalized maximum likelihood algorithm for DNA sequence compression," *ACM Trans. Inf. Syst.*, vol. 23, no. 1, pp. 3–34, Jan. 2005.
[27] B. Behzadi and F. Le Fessant, "DNA compression challenge revisited," in *Proc. CPM-2005 Combinatorial Pattern Matching*, Jeju Island, Korea, Jun. 2005.
[28] T. C. Bell, J. G. Cleary, and I. H. Witten*, Text Compression*. Upper Saddle River, NJ: Prentice-Hall, 1990.
[29] D. Salomon*, Data Compression—The Complete Reference*, 2nd ed. Berlin, Germany: Springer-Verlag, 2000.
[30] K. Sayood*, Introduction to Data Compression*, 2nd ed. San Mateo, CA: Morgan Kaufmann, 2000.
[31] S. Ouyang and C. R. Buell, "The TIGR plant repeat databases: A collective resource for the identification of repetitive sequences in plants," *Nucleic Acids Res.*, vol. 32, pp. D360–D363, 2004.
[32] V. Walbot, "A green chapter in the book of life," *Nature*, vol. 408, pp. 794–795, Dec. 2000.

Armando J. Pinho (S'87–M'95) was born in Lisbon, Portugal, in July 1964. He received the electronics and telecommunications engineering degree from the University of Aveiro, Aveiro, Portugal, in 1988, the Master degree in electrical and computers engineering from the Technical University of Lisbon (IST), Portugal, in 1991, and the Ph.D. degree in electrical engineering from the University of Aveiro, in 1996.

He is currently an Associate Professor at the Departamento de Electrónica, Telecomunicações e Informática (DETI), University of Aveiro, and a Researcher at the Signal Processing Laboratory of the Institute of Electronics and Telematics Engineering of Aveiro (IEETA). His research interests include image and video coding, data compression and bioinformatics.

António J. R. Neves (S'03) was born in Torres Novas, Portugal, in July 1979. He received the electronics and telecommunications engineering degree from the University of Aveiro, Aveiro, Portugal, in 2002. He is currently working toward the Ph.D. degree in electrical engineering at the University of Aveiro.

He is currently a Researcher with the Signal Processing Laboratory, the Institute of Electronics and Telematics Engineering of Aveiro (IEETA). His research interests include image and video coding; in particular, image compression.

Vera Afreixo received the B.Sc. and M.Sc., degrees in mathematics from University of Aveiro, Aveiro, Portugal, in 2000 and 2003. She is currently working toward the Ph.D. degree in electrical engineering at the University of Aveiro.

She is with the Signal Processing Laboratory/the Institute of Electronics and Telematics Engineering of Aveiro (IEETA) since 2004. Since 2002, she cooperates with the Bioinformatics group at the University of Aveiro/IEETA.

Carlos A. C. Bastos (S'87–M'99) received the electronics and telecommunications engineering degree from the University of Aveiro, Aveiro, Portugal, in 1989, the M.Sc. degree in electronic engineering from the University College of North Wales, Bangor, U.K., in 1991, and the Ph.D. degree in electrical engineering from the University of Aveiro in 1999.

In 1993, he joined the Departamento de Electrónica, Telecomunicações e Informática of the University of Aveiro, where he is an Assistant Professor; he is also with the Institute of Electronics and Telematics Engineering of Aveiro (IEETA), Aveiro. His main research interests are biomedical signal processing and signal modelling, and DNA data processing.

Paulo J. S. G. Ferreira (M'99) was born in Torres Novas, Portugal, and is Full Professor at the Departamento de Electrónica, Telecomunicações e Informática/IEETA, University of Aveiro, Aveiro, Portugal. His current research interests include bioinformatics and symbolic signals, coding, sampling and signal reconstruction. He was an Associate Editor of the IEEE TRANSACTIONS ON SIGNAL PROCESSING. He co-edited, with J. J. Benedetto, the book *Modern Sampling Theory: Mathematics and Applications* (Birkhauser, 2001).

Dr. Ferreira is currently a member of the editorial board of the *Journal of Applied Functional Analysis* and an Editor-in-Chief of *Sampling Theory in Signal and Image Processing*.