16th European Signal Processing Conference (EUSIPCO 2008), Lausanne, Switzerland, August 25-29, 2008, copyright by EURASIP

INVERTED-REPEATS-AWARE FINITE-CONTEXT MODELS FOR DNA CODING

Armando J. Pinho, António J. R. Neves and Paulo J. S. G. Ferreira

Signal Processing Lab, DETI / IEETA
University of Aveiro, 3810–193 Aveiro, Portugal
ap@ua.pt / an@ua.pt / pjf@ua.pt

ABSTRACT

Finite-context models have been used for DNA sequence compression as secondary, fall back mechanisms, the generalized opinion being that models with order larger than two or three are inappropriate. In this paper we show that finite-context models can also be used as the main encoding method, and that they are effective for model orders at least as higher as thirteen. Moreover, we propose a new model updating scheme that takes into account inverted repeats, a common characteristic in DNA sequences.

1. INTRODUCTION

The human genome is determined by approximately 3 000 million bases [1], whereas the genome of the wheat has about 16 000 million [2]. This means that, being DNA a language written with an alphabet of four different symbols (usually known as nucleotides or bases), namely, Adenine (A), Cytosine (C), Guanine (G), and Thymine (T), it takes approximately 750 MBytes to store the human genome (using $\log_2 4 = 2$ bits per symbol) and 4 GBytes to store the genome of the wheat.

Frequently, the main motivation for studying data compression algorithms is the need for efficient storage or transmission of information. This is also an important motivation in the case of DNA, because of the rapid increase in the number of sequenced genomes. However, there is another aspect of paramount interest that is associated to compression algorithms. Every compression method has an underlying data model. The better the compression method is, the better the model. Therefore, looking for DNA compression algorithms is also a form of finding models that describe the information source associated to DNA.

In a recent work [3, 4], we proposed a three-state finite-context model for DNA protein-coding regions, i.e., for the parts of the DNA that carry information regarding how proteins are synthesized. Basically, this three-state model proved to be better than a single-state model, given additional evidence of a phenomenon that is common in those protein-coding regions.

In this paper, we investigate the performance of finite-context models for unrestricted DNA, i.e., DNA including coding and non-coding parts. We show that finite-context models are able to give very competitive compression results, specially in long sequences, such as those of the human chromosomes. Moreover, we show that it is possible to incorporate in these models a characteristic usually found in DNA sequences, which is the occurrence of inverted repeats, i.e., sequences of bases that are the reversed complement of other sequences.

This work was supported in part by the FCT (Fundação para a Ciência e Tecnologia) grant PTDC/EIA/72569/2006.

2. DNA COMPRESSION METHODS

The first method designed specifically for compressing DNA sequences was proposed by Grumbach *et al.* in 1993 and was named *Biocompress* [5]. This technique is based on the sliding window algorithm proposed by Ziv and Lempel, also known as LZ77 [6]. According to this universal data compression technique, a sub-sequence is encoded using a reference to an identical sub-sequence that occurred in the past. *Biocompress* uses a characteristic usually found in DNA sequences which is the occurrence of inverted repeats. These are sub-sequences that are both reversed and complemented ($A \leftrightarrow T$, $C \leftrightarrow G$). The second version of *Biocompress*, *Biocompress-2*, introduced an additional mode of operation, based on a second order finite-context arithmetic encoder [7].

Rivals *et al.* proposed another compression technique based on exact repetitions, *Cfact*, which relies on a two-pass strategy [8, 9]. In the first pass, the complete sequence is parsed using a suffix tree, producing a list of the longest repeating sub-sequences that have a potential coding gain. In the second pass, those sub-sequences are encoded using references to the past, whereas the rest of the symbols are left uncompressed.

The idea of using repeating sub-sequences was also exploited by Chen *et al.* [10, 11]. The authors proposed a generalization of this strategy such that approximate repeats of sub-sequences and of inverted repeats could also be handled. In order to reproduce the original sequence, the algorithm, named *GenCompress*, uses operations such as replacements, insertions and deletions. As in *Biocompress*, *GenCompress* includes a mechanism for deciding if it is worthwhile to encode the sub-sequence under evaluation using the substitution-based model. If not, it falls back to a 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*, [12]. 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, *CTW+LZ*, was proposed by Matsumoto *et al.* [13]. Basically, long repeating sub-sequences or inverted repeats, exact or approximate, are encoded by a LZ-type algorithm, whereas short sub-sequences are compressed using CTW.

One of the main problems of techniques based on sub-sequence matching is the time taken by the search operation.

Manzini *et al.* [14] addressed this problem and proposed a fast, although competitive, DNA encoder, based on fingerprints. Basically, in this approach, small sub-sequences are not considered for matching. Instead, the algorithm focus on finding long matching sub-sequences (or inverted repeats). Like most other methods, this technique also uses fall back mechanisms for the zones where matching fails, in this case, finite-context arithmetic coding of order two (*DNA2*) or three (*DNA3*).

Tabus *et al.* proposed a sophisticated DNA sequence compression method based on normalized maximum likelihood discrete regression for approximate block matching [15]. This work, later improved for compression performance and speed [16] (*GeNML*), encodes fixed-size blocks by referencing a previously encoded sub-sequence with minimum Hamming distance. Only replacement operations are allowed for editing the reference sub-sequence which, therefore, always have the same size as the block, although may be located in an arbitrary position inside the already encoded sequence. Fall back modes of operation are also considered, namely, a finite-context arithmetic encoder of order one and a transparent mode in which the block passes uncompressed.

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 inverted repeats, and either CTW or order-2 arithmetic coding for non-repeating regions [17]. Moreover, *DNAPack* relies on dynamic programming techniques for choosing the repeats, instead of greedy approaches as others do.

3. INVERTED-REPEATS-AWARE FINITE-CONTEXT MODEL

Consider an information source that generates symbols, s, from an alphabet $\mathscr{A}$. At time t, the sequence of outcomes generated by the source is $x^t = x_1 x_2 \ldots x_t$. A finite-context model (see Fig. 1) 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) [18–20]. 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 $|\mathscr{A}|^M$, which dictates the model complexity or cost. In the case of DNA, since $|\mathscr{A}| = 4$, an order-M model implies 4^M conditioning states.

In practice, the probability that the next outcome x_{t+1} is s, where $s \in \mathscr{A} = \{A, C, G, T\}$, is obtained using the estimator

$$P(x_{t+1} = s|c^t) = \frac{n_s^t + 1}{\sum_{a \in \mathscr{A}} n_a^t + 4}, \tag{1}$$

where n_s^t represents the number of times that, in the past, the information source generated symbol s having c^t as the conditioning context. This estimator can be viewed as the extension of Laplace's estimator for the case of a four-symbol alphabet. Note that initially, when all counters are zero, the symbols have probability $1/4$, i.e., they are assumed equally probable. The 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 1 shows an example of how a finite-context model is typically implemented. In this example, an order-5 finite-context model is presented (as that of Fig. 1). 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 "*ATAGA*", i.e., $c^t = ATAGA$, then the model communicates the following probability estimates to the arithmetic encoder:

$$P(A|ATAGA) = (16+1)/(58+4),$$
$$P(C|ATAGA) = (6+1)/(58+4),$$
$$P(G|ATAGA) = (21+1)/(58+4)$$

and

$$P(T|ATAGA) = (15+1)/(58+4).$$

The block denoted "Encoder" in Fig. 1 is an arithmetic encoder. It is well known that practical arithmetic coding generates output bit-streams with average bitrates almost identical to the entropy of the model [18–20]. The theoretical bitrate average (entropy) of the finite-context model after encoding N symbols is given by

$$H_N = -\frac{1}{N} \sum_{t=0}^{N-1} \log_2 P(x_{t+1} = s|c^t) \quad \text{bps}, \tag{2}$$

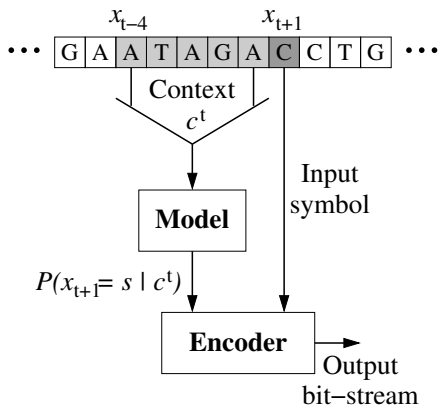

Figure 1: Finite-context model: the probability of the next outcome, x_{t+1}, is conditioned by the M last outcomes. In this example, $M = 5$.

Table 1: 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^t	n_C^t	n_G^t	n_T^t	$\sum_{a \in \mathscr{A}} n_a^t$
AAAAA	23	41	3	12	79
⋮	⋮	⋮	⋮	⋮	⋮
ATAGA	16	6	21	15	58
⋮	⋮	⋮	⋮	⋮	⋮
GTCTA	19	30	10	4	63
⋮	⋮	⋮	⋮	⋮	⋮
TTTTT	8	2	18	11	39

Table 2: Table 1 updated after encoding symbol "C" according to context "ATAGA" (see example of Fig. 1).

Context, c^t	n_A^t	n_C^t	n_G^t	n_T^t	$\sum_{a\in\mathscr{A}} n_a^t$
AAAAA	23	41	3	12	79
⋮	⋮	⋮	⋮	⋮	⋮
ATAGA	16	$\boxed{7}$	21	15	59
⋮	⋮	⋮	⋮	⋮	⋮
GTCTA	19	30	10	4	63
⋮	⋮	⋮	⋮	⋮	⋮
TTTTT	8	2	18	11	39

Table 3: Table 1 updated after encoding symbol "C" according to context "ATAGA" (see example of Fig. 1) and taking the inverted repeats property into account.

Context, c^t	n_A^t	n_C^t	n_G^t	n_T^t	$\sum_{a\in\mathscr{A}} n_a^t$
AAAAA	23	41	3	12	79
⋮	⋮	⋮	⋮	⋮	⋮
ATAGA	16	$\boxed{7}$	21	15	59
⋮	⋮	⋮	⋮	⋮	⋮
GTCTA	19	30	10	$\boxed{5}$	64
⋮	⋮	⋮	⋮	⋮	⋮
TTTTT	8	2	18	11	39

where "bps" stands for "bits per symbol". When dealing with DNA bases, the generic acronym "bps" is sometimes replaced with "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.

According to the example of Fig. 1 and Table 1, the next symbol to encode, "C", would require, theoretically, $-\log_2(6+1)/(58+4) \approx 3.15$ bits. Note that this is more than two bits because, in this example, "C" is the least probable symbol and, therefore, needs more bits to be encoded than the more probable ones. After encoding this symbol, the counters will be updated according to Table 2.

As we mentioned previously, DNA sequences frequently contain sub-sequences that are reversed and complemented copies of some other sub-sequences. These sub-sequences are named "inverted repeats". As described in Section 2, this particularity of DNA is used by most of the DNA compression methods that have been proposed and that rely on the sliding window searching paradigm. However, as far as we know, this property has never been incorporated in finite-context approaches.

For exploring the inverted repeats of a DNA sequence, besides updating the corresponding counter after encoding a symbol, we also update another counter that we determine in the following way. Consider the example given in Fig. 1, where the context is the string "ATAGA" and the symbol to encode is "C". Reversing the string obtained by concatenating the context string and the symbol, i.e., "ATAGAC", we obtain the string "CAGATA". Complementing this string ($A \leftrightarrow T$, $C \leftrightarrow G$), we get "GTCTAT". Now we consider the prefix "GTCTA" as the context and the suffix "T" as the symbol that determines which counter should be updated. Therefore, according to this procedure, for taking into consideration the inverted repeats, after encoding symbol "C" of the example in Fig. 1, the counters should be updated according to Table 3.

4. EXPERIMENTAL RESULTS

For evaluating the finite-context model that we described in the previous section we used the same DNA sequences as Manzini *et al.* in [14], which are available from `www.mfn.unipmn.it/~manzini/dnacorpus`. This corpus contains sequences from four organisms: yeast (*Saccharomyces cerevisiae*, chromosomes 1, 4, 14 and the mitochondrial DNA), mouse (*Mus musculus*, chromosomes 7, 11, 19, x and y), arabidopsis (*Arabidopsis thaliana*, chromosomes 1, 3 and 4) and human (*Homo sapiens*, chromosomes 2, 13, 22, x and y).

Each of the sequences was encoded using finite-context models with orders ranging from four to thirteen, with and without the inverted repeats updating mechanism. As in most of the other DNA encoding techniques, we also provided a fall back method that is used if the main method produces worse results. This is checked on a block by block basis, where each block is composed of one hundred DNA bases. As in the DNA3 version of Manzini's encoder, we used an order-3 finite-context model as fall back method [14]. Note that, in our case, both the main and fall back methods rely on finite-context models.

Table 4 presents the results of compressing the DNA sequences with the "normal" finite-context model (FCM) and with the model that is aware of the inverted repeats (FCM-IR). Besides the bitrate, the order of the model that provided the best results is also indicated. For comparison, we included the results of the DNA3 compressor of Manzini *et al.* [14].

As can be seen from the results presented in Table 4, the finite-context models using the updating mechanism for inverted repeats (FCM-IR) always provide better results than the "normal" finite-context models (FCM). This confirms that the finite-context models can be modified according to the proposed scheme to exploit inverted repeats.

It can also be seen that finite-context models attain compression performances that seem to differ from organism to organism. In fact, they could not beat DNA3 in the case of yeast and arabidopsis, but they attained better results for the mouse and human sequences. This is an interesting aspect, that deserves further investigation.

Figure 2 shows how the finite-context models perform for various model orders, from order-4 to order-13. As can be seen in these examples of the "y-1" and "h-y" sequences, the inverted repeats mechanism produces increasingly better results as the order of the model increases. It can also be observed that, for the case of "h-y", it would be reasonable to expect that further increase in the order of the model would provide increased compression performance. In fact,

this phenomenon shows up in most of the larger sequences, suggesting the possibility of additional improvements. However, our current implementation of the finite-context models requires 4^{M+1} counters for an order-M model, which, for two-byte counters, implies 2^{2M+3} bytes of memory. For an order-13 model, this corresponds to 512 MBytes, which is still reasonable, specially if considering that the implementation is quite fast. Finding alternative ways of implementing these models that could reduce the memory requirements is of key importance to explore deeper contexts.

5. CONCLUSION

Finite-context modeling has been used for DNA compression as a secondary, fall back method. The general opinion has been that finite-context models of orders higher than two or three are inappropriate for this problem. However, as far as we are aware of, no systematic study of this issue has been carried out. In this paper, we have shown that finite-context models of orders higher than four (we tested models with orders up to thirteen) are indeed able to attain significant performance, at least for some organisms. An obvious example is the human DNA, for which the finite-context models could easily beat the DNA3 technique of Manzini *et al.* [14]. Although not the best method available in terms of compression performance, DNA3 is a well-balanced approach, with reasonable computation time requirements. Other methods, such as GeNML [16], attain better compression results but at the cost of much longer processing times.

We introduced an updating mechanism in the finite-context models that permits capturing information regarding the inverted repeats usually found in DNA sequences. Although common in DNA compression methods based on sub-sequence matching, this is the first time that it is proposed in association to finite-context models. Moreover, the experimental results have been consistently better when this updating mechanism was used, showing its appropriateness.

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, pp. 791, Dec. 2000.

[3] P. J. S. G. Ferreira, A. J. R. Neves, V. Afreixo, and A. J. Pinho, "Exploring three-base periodicity for DNA compression and modeling," in *Proc. of the IEEE Int. Conf. on Acoustics, Speech, and Signal Processing, ICASSP-2006*, Toulouse, France, May 2006, vol. 5, pp. 877–880.

[4] A. J. Pinho, A. J. R. Neves, V. Afreixo, Carlos A. C. Bastos, and P. J. S. G. Ferreira, "A three-state model for DNA protein-coding regions," *IEEE Trans. on Biomedical Engineering*, vol. 53, no. 11, pp. 2148–2155, Nov. 2006.

[5] S. Grumbach and F. Tahi, "Compression of DNA sequences," in *Proc. of the Data Compression Conf., DCC-93*, Snowbird, Utah, 1993, pp. 340–350.

[6] J. Ziv and A. Lempel, "A universal algorithm for sequential data compression," *IEEE Trans. on Information Theory*, vol. 23, pp. 337–343, 1977.

[7] S. Grumbach and F. Tahi, "A new challenge for compression algorithms: genetic sequences," *Information Processing & Management*, vol. 30, no. 6, pp. 875–886, 1994.

[8] E. Rivals, J.-P. Delahaye, M. Dauchet, and O. Delgrange, "A guaranteed compression scheme for repetitive DNA sequences," Tech. Rep. IT–95–285, LIFL, Université des Sciences et Technologies de Lille, Nov. 1995.

[9] E. Rivals, J.-P. Delahaye, M. Dauchet, and O. Delgrange, "A guaranteed compression scheme for repetitive DNA sequences," in *Proc. of the Data Compression Conf., DCC-96*, Snowbird, Utah, 1996, p. 453.

[10] X. Chen, S. Kwong, and M. Li, "A compression algorithm for DNA sequences and its applications in genome comparison," in *Genome Informatics 1999: Proc. of the 10th Workshop*, K. Asai, S. Miyano, and T. Takagi, Eds., Tokyo, Japan, 1999, pp. 51–61.

[11] X. Chen, S. Kwong, and M. Li, "A compression algorithm for DNA sequences," *IEEE Engineering in Medicine and Biology Magazine*, vol. 20, pp. 61–66, 2001.

[12] 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.

[13] T. Matsumoto, K. Sadakane, and H. Imai, "Biological sequence compression algorithms," in *Genome Informatics 2000: Proc. of the 11th Workshop*, A. K. Dunker, A. Konagaya, S. Miyano, and T. Takagi, Eds., Tokyo, Japan, 2000, pp. 43–52.

[14] G. Manzini and M. Rastero, "A simple and fast DNA compressor," *Software - Practice and Experience*, vol. 34, pp. 1397–1411, 2004.

[15] I. Tabus, G. Korodi, and J. Rissanen, "DNA sequence compression using the normalized maximum likelihood model for discrete regression," in *Proc. of the Data Compression Conf., DCC-2003*, Snowbird, Utah, 2003, pp. 253–262.

[16] G. Korodi and I. Tabus, "An efficient normalized maximum likelihood algorithm for DNA sequence compression," *ACM Trans. on Information Systems*, vol. 23, no. 1, pp. 3–34, Jan. 2005.

[17] B. Behzadi and F. Le Fessant, "DNA compression challenge revisited," in *Combinatorial Pattern Matching: Proc. of CPM-2005*, Jeju Island, Korea, June 2005, Lecture Notes in Computer Science, Springer.

[18] T. C. Bell, J. G. Cleary, and I. H. Witten, *Text compression*, Prentice Hall, 1990.

[19] D. Salomon, *Data compression - The complete reference*, Springer, 2nd edition, 2000.

[20] K. Sayood, *Introduction to data compression*, Morgan Kaufmann, 2nd edition, 2000.

Table 4: Compression values, in bits per base (bpb), regarding a number of DNA sequences. The "DNA3" column shows the results obtained by Manzini *et al.* [14]. Columns "FCM" and "FCM-IR" contain the results, respectively, of the "normal" finite-context models and of the finite-context models equipped with the inverted repeats updating mechanism. The order of the model that provided the best result is indicated under the columns labeled "Order".

Name	Size	DNA3	FCM		FCM-IR	
		bpb	Order	bpb	Order	bpb
y-1	230 203	1.871	10	1.935	11	1.909
y-4	1 531 929	1.881	12	1.920	12	1.910
y-14	784 328	1.926	9	1.945	12	1.938
y-mit	85 779	1.523	6	1.494	7	1.479
Average	–	1.882	–	1.915	–	1.904
m-7	5 114 647	1.835	11	1.849	12	1.835
m-11	49 909 125	1.790	13	1.794	13	1.778
m-19	703 729	1.888	10	1.883	10	1.873
m-x	17 430 763	1.703	12	1.715	13	1.692
m-y	711 108	1.707	10	1.794	11	1.741
Average	–	1.772	–	1.780	–	1.762
at-1	29 830 437	1.844	13	1.887	13	1.878
at-3	23 465 336	1.843	13	1.884	13	1.873
at-4	17 550 033	1.851	13	1.887	13	1.878
Average	–	1.845	–	1.886	–	1.876
h-2	236 268 154	1.790	13	1.748	13	1.734
h-13	95 206 001	1.818	13	1.773	13	1.759
h-22	33 821 688	1.767	12	1.728	12	1.710
h-x	144 793 946	1.732	13	1.689	13	1.666
h-y	22 668 225	1.411	13	1.676	13	1.579
Average	–	1.762	–	1.732	–	1.712

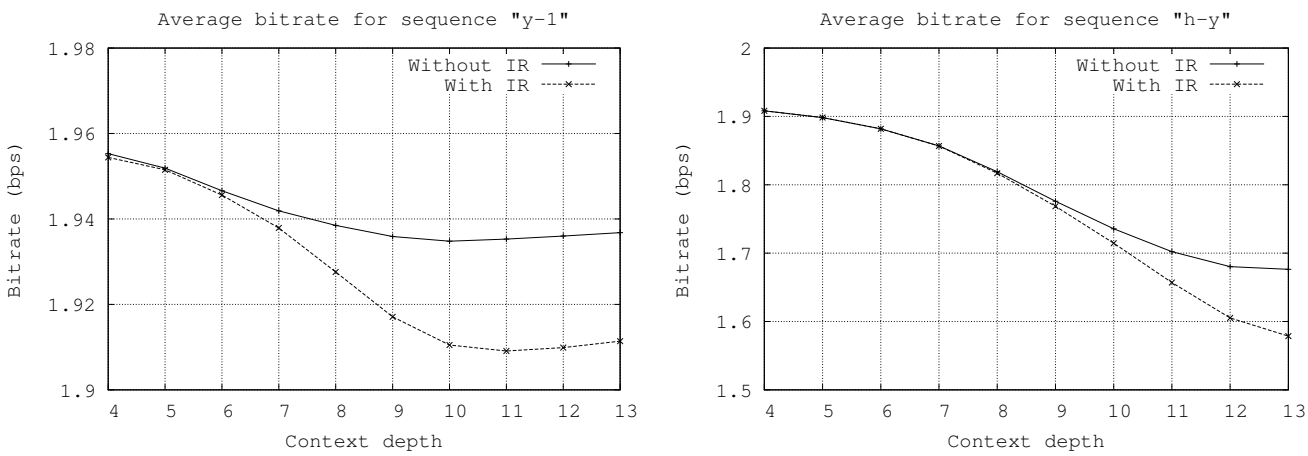

Figure 2: Performance of the finite-context model as a function of the order of the model, with and without the updating mechanism for inverted repeats (IR), for sequences "y-1" and "h-y".