Variable Order Finite-Context Models in DNA Sequence Coding

Daniel A. Martins*, António J. R. Neves, Armando J. Pinho

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

Abstract. Being an essential key in biological research, the DNA sequences are often shared between researchers and digitally stored for future use. As these sequences grow in volume, it also grows the need to encode them, thus saving space for more sequences. Besides this, a better coding method corresponds to a better model of the sequence, allowing new insights about the DNA structure. In this paper, we present an algorithm capable of improving the encoding results of algorithms that depend of low-order finite-context models to encode DNA sequences. To do so, we implemented a variable order finite-context model, supported by a predictive function. The proposed algorithm allows using three finite-context models at once without requiring the inclusion of side information in the encoded sequence. Currently, the proposed method shows small improvements in the encoding results when compared with same order finite-context models. However, we also present results showing that there is space for further improvements regarding the use variable order finite-context models for DNA sequence coding.

1 Introduction

The complete human genome has about 3 000 million bases [1] and the genome of the wheat has nearly 16 000 million [2]. These two examples show that DNA sequences can have very large sizes when stored or transmitted without any kind of encoding.

DNA is 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). Therefore, without compression, it takes approximately 750 MBytes to store the human genome (using $\log_2 4 = 2$ bits per base) and 4 GBytes to store the genome of the wheat.

Besides this motivation, DNA compression can play an important role in providing new insights regarding the underlying information source. This is due to the models associated to each compression technique: the closer the model to the DNA information source, the better the resulting encoding will be.

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

In the last years, finite-context models (FCM) have been used for DNA sequence compression as secondary, fallback, mechanisms [3,4,5,6,7,8]. The FCM's used in these cases are usually of low-order (typically, 1 to 3) and are used only when the main encoding algorithm produces bad results.

In order to improve the encoding results that depend on these low-order FCM's, we developed an algorithm capable of choosing between models of different orders. This method allows to encode DNA sequences using FCM's with different orders as it was a single model. The improvements attained with this approach are due to the use of multiple models of different orders, without needing to include side information. This reduces the average number of bits per base (bpb) necessary for encoding the sequences, therefore improving the encoding result.

This paper contains six sections. We start by introducing the finite-context models and their relevance in current DNA encoding techniques (Section 2). In Section 3, we briefly explain the concept of variable order finite-context models (VOFCM) and we introduce the proposed prediction function. The tree representation of the VOFCM is presented in Section 4. Finally, we present experimental results in Section 5 and draw some conclusions in Section 6.

2 Low Order Finite-Context Models

Consider an information source that generates symbols, s, from a finite alphabet $\mathcal{A}$. At time t, the sequence of outcomes generated by the source is $x^t = x_1 x_2 \dots x_t$. The proposed algorithm relies on a VOFCM that generates probability estimates that are then used for driving an arithmetic encoder [9]. The VOFCM collects statistical information from a context of depth M_{vofcm}, which is also the maximum order of the VOFCM. At time t, we represent the conditioning outcomes by $c^t = x_{t-M_{\text{vofcm}}+1}, \dots, x_{t-1}, x_t$. Note that the total number of conditioning states of a VOFCM with maximum context depth M_{vofcm} is $\frac{|\mathcal{A}|^{M_{\text{vofcm}}+1}-1}{|\mathcal{A}|-1}$. In the case of DNA, $|\mathcal{A}| = 4$. In Fig 1 and 2 we show an example of a FCM and a VOFCM, respectively.

From the most recent algorithms for DNA compression, we can conclude that there is a common interest in using several competing encoding techniques. This is due to the fact that different regions in the DNA sequences are better encoded with different algorithms. Knowing this property, it is possible to use a fallback algorithm when the primary algorithm produces bad results.

Regardeless of the main encoding method, the fallback algorithm is most of the cases a low-order FCM. Some of the DNA compression algorithms that use FCM's as a fallback mechanism are:

- *Biocompress-2* [3];
- *GenCompress* [4,5];
- *DNA2* and *DNA3* [6];
- *GeNML* [7];
- *DNAEnc* [8].

Using a FCM, we can model the DNA sequence by counting the occurrences of each symbol given a certain past (see Fig. 1). Encouraged by the overall acceptance as a fallback encoding algorithm, we went from this fixed low-order FCM to a variable low-order FCM, as described in Section 3.

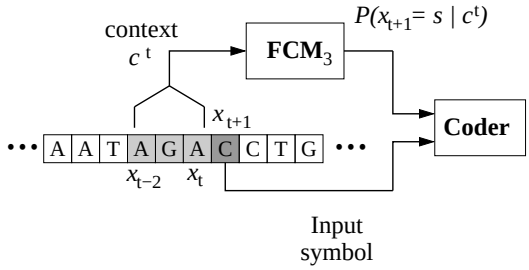

Fig. 1. Example of an order-3 finite-context model, used for estimating the probabilities. The probability of the next outcome, x_{t+1}, is conditioned by the last 3 outcomes.

3 Variable Order Finite-Context Models

In this paper, we present a technique capable of improving the compression rates produced by low-order FCM's. This is achieved by allowing the encoding system to use a FCM with variable order (VOFCM). In fact, this VOFCM is a group of three fixed low-order FCM's, being the order of the FCM decided automatically, avoiding the need to pass information (overhead) of the chosen FCM to the decoder. This way, the proposed algorithm acts like a single FCM model, making its use transparent to the encoder and decoder. This VOFCM is also known as a *Variable Length Markov Chain* (VLMC) [10].

In order to choose which FCM order is used, we developed a function that automatically tries to select the best order for each symbol being encoded. The developed function, called *Efficiency*, is in fact a simple prediction system that decides which FCM to use, taking into account the following statistics:

- the entropy associated to the FCM;
- the probability associated to the last outcome symbol in the FCM;
- the FCM associated *Bonus*.

The *Bonus* is a function with range $[0, 1]$, that increases if the last occurred symbol was the most probable to appear and decreases otherwise. Next we define the proposed *Efficiency* function (1), and the *Bonus* function (2):

$$E(k,t) = \frac{P(x_t|\mathrm{FCM_k})B(k,t)}{H(k,t)}, \tag{1}$$

$$B(k,t) = \begin{cases} B(k,t-1) - (1 - B(k,t-1)) \log P(x_{\mathrm{t}}|\mathrm{FCM_k}) \\ B(k,t-1) + B(k,t-1) \log P(x_{\mathrm{t}}|\mathrm{FCM_k}) \end{cases}, \qquad (2)$$

where the first member of (2) is used when

$$P(x_{\mathrm{t}}|\mathrm{FCM_k}) = \max P(x|\mathrm{FCM_k}), \quad x \in \{A, C, T, G\},$$

and where k represents the order of the FCM, $H(k,t)$ is the entropy of $\mathrm{FCM_k}$ at time t, and $P(x_{\mathrm{t}}|\mathrm{FCM_k})$ is the conditional probability of the last symbol calculated according to a FCM of order k.

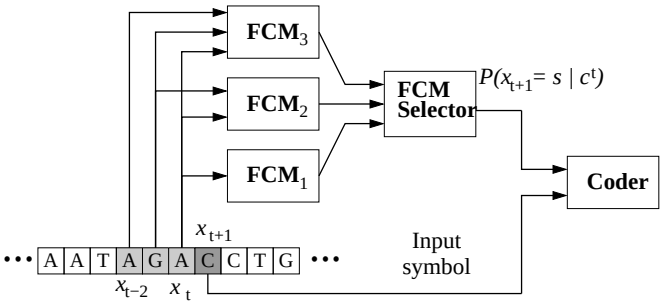

Fig. 2. Example of the proposed approach, where the variable low-order FCM is composed by 3 fixed low-order FCM of order 1, 2 and 3. The next input symbol probability is conditioned by the last 3 outcomes and the *Efficiency* function.

4 Tree-Structured Representation of VOFCM

Inspired by the work of J. Rissanen [11], also considered in [10], we implemented a tree representation for the statistical information collected from the DNA sequence. Note that we do not use the *Context Gather* algorithm described in Section III of [11], but, instead, we developed a greedy algorithm to create the tree.

Basically, each level of the tree corresponds to a position of a context symbol. For example, a tree with a depth 3 corresponds to an order-3 context model. Each node of the tree is constructed the first time that the symbol appears in the corresponding position of the context. Note that by *context* we refer to a ordered group of outcome symbols.

Each node of the tree contains the probabilistic information about a specific model and each branch represents a past outcome symbol. For example, the root node has 4 branches, each one representing the last outcome symbol. Furthermore, with this representation we have easy access to all the FCM's. In fact, as we visit the tree, from the root to the leaves, we are visiting the different FCM's from a specific context.

Fig. 3 shows an example of the tree-structured representation of the VOFCM. In this example, it is possible to see that the tree has not yet reached its full extension. The *TREE* block has the tree basic information like number of nodes and maximum FCM order (*maxDepth*). From the *TREE* block it grows two nodes. On the left node, we have the next outcome (*NEXT*) and the current context (*CONTEXT*). The right node (*NODE R*) is the root node of the tree. In each node, we have the statistical information about each symbol, the *Bonus* and the *Efficiency* values. The efficiency value is presented in the first row of each node. Still in Fig. 3, the yellow and green nodes represent the current context, the green node being the one that presents higher *Efficiency*.

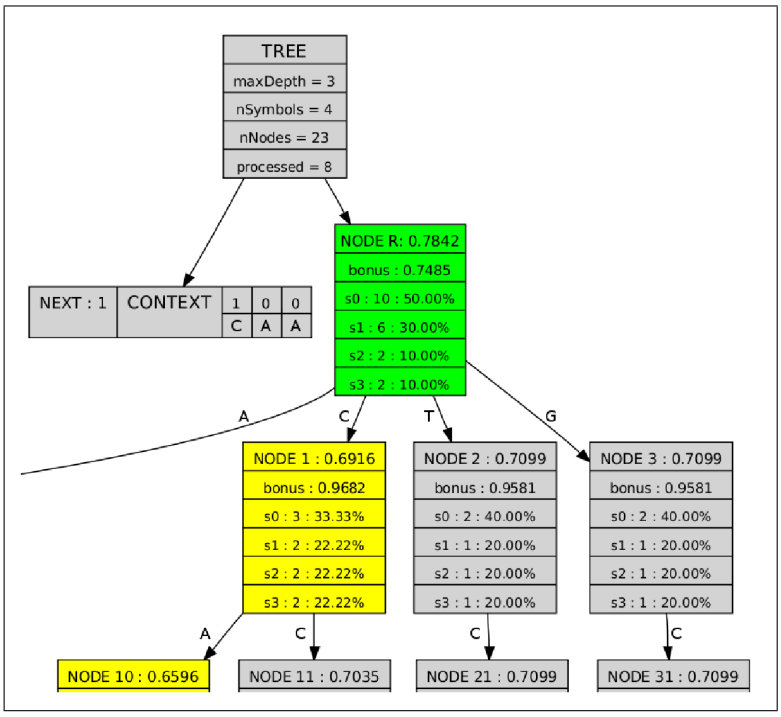

Fig. 3. Detail of a tree-structured VOFCM. In the figure, it is possible to see the tree details along with the context and the root node.

5 Experimental results

In order to present a first approach to the lower bound of encoding using low-order FCM's, we compare the difference between encoding using a fixed order FCM and encoding using a variable order FCM (always choosing the best order

for each symbol being encoded). Afterwards, we tested the proposed method using the *Efficiency* function presented in Section 3.

The results are presented in Table 1 in the *Test 1* columns. The first and second columns of the table contain the name and size (in number of bases) of the DNA sequences. In the *FCM* columns it is presented the encoding results (in bpb) of the DNA sequences using the fixed low-order FCM, for orders 1, 2 and 3.

In the *FCM Limit* column it is presented a lower bound of the encoding results using the variable low-order FCM. These results were obtained choosing the best from the three models in each symbol to be encoded. Because this choice was made by comparing the encoding results from the three models, the results provided in the column *FCM Limit* are to be taken as a lower limit comparison, as they don't include the necessary overhead to signal which was the selected FCM for each symbol. Doing so would make the encoder unusable, as the necessary overhead would increase the resulting bpb value drastically.

Finally, the column *FCM Eff* presents the results from the proposed algorithm using the *Efficiency* function to decide which order of the FCM to use. Note that in columns 6 and 7 we use a VOFCM with maximum order 3 as this is only intended to use in fallback low-order FCM algorithms.

In [8] it is presented some results using arithmetic encoding with FCM's as primary and fallback methods. Using the same primary encoding method, i.e., a FCM with the same order than the presented in [8], we tested the proposed VOFCM method as the fallback method. To do so, we replaced the fixed order FCM used in [8] (an order-3 FCM) as fallback encoding by the VOFCM proposed in this paper. The results are presented in Table 1 in the *Test 2* columns.

The *DNA3* column shows the results of the DNA3 compressor of Manzini *et al.* [6], to allow some comparison with other existing methods. The *FCM-IR* columns show the order of the primary FCM that was obtained using a "brute force" approach [8] and the corresponding encoding results in bpb. The columns *VOFCM* show the order of the primary FCM (that is equal to the used in *FCM-IR*) and the encoding results in bpb. In the *Total* row it is possible to notice that the results obtained in *VOFCM* are globally slightly better.

6 Conclusions

According to Table 1, we can see the benefits of using the proposed method, as the bpb necessary to the encoding reduce. In columns *Test 1* of Table 1, we show the potential gain of using the proposed method versus the use of a fixed order FCM. Besides the improvements of the encoding results, the proposed method allows using three FCM's with different orders at the same time. Recall that the FCM model is chosen by the *Efficiency* function on an symbol by symbol basis.

From columns *Test 2*, we conclude that it is possible to bring improvements to the overall encoding results, replacing the fallback mechanism by the proposed VOFCM with the *Efficiency* function. Furthermore, we can take advantage of

using more than one FCM without having to conduct a "brute force" search for the FCM with best encoding results.

In *Test 1*, the improvements provided by the use of the VOFCM are more visible than in *Test 2*, because, in the latter, the proposed algorithm is only used when the primary encoding method fails, making the improvements less notable in the results. Nevertheless, the proposed method brought better overall results to the DNA encoding technique that uses low-order FCM's. Moreover, the results from column *FCM Limit* show that the use of VOFCM with a predictive system can bring even better encoding results. To explore this potentiality, we pretend to further explore the use the VOFCM using other predictive and cost functions and apply them to higher order FCM's.

References

1. Rowen, L., Mahairas, G., Hood, L.: Sequencing the human genome. Science **278** (October 1997) 605–607
2. Dennis, C., Surridge, C.: *A. thaliana* genome. Nature **408** (December 2000) 791
3. Grumbach, S., Tahi, F.: A new challenge for compression algorithms: genetic sequences. Information Processing & Management **30**(6) (1994) 875–886
4. Chen, X., Kwong, S., Li, M.: A compression algorithm for DNA sequences and its applications in genome comparison. In Asai, K., Miyano, S., Takagi, T., eds.: Genome Informatics 1999: Proc. of the 10th Workshop, Tokyo, Japan (1999) 51–61
5. Chen, X., Kwong, S., Li, M.: A compression algorithm for DNA sequences. IEEE Engineering in Medicine and Biology Magazine **20** (2001) 61–66
6. Manzini, G., Rastero, M.: A simple and fast DNA compressor. Software—Practice and Experience **34** (2004) 1397–1411
7. Korodi, G., Tabus, I.: An efficient normalized maximum likelihood algorithm for DNA sequence compression. ACM Trans. on Information Systems **23**(1) (January 2005) 3–34
8. Pinho, A.J., Neves, A.J.R., Ferreira, P.J.S.G.: Inverted-repeats-aware finite-context models for DNA coding. In: Proc. of the 16th European Signal Processing Conf., EUSIPCO-2008, Lausanne, Switzerland (August 2008)
9. Salomon, D.: Data compression - The complete reference. 2nd edn. Springer (2000)
10. Bühlmann, P., Wyner, A.J.: Variable length Markov chains. The Annals of Statistics **27**(2) (1999) 480–513
11. Rissanen, J.: A universal data compression system. IEEE Trans. on Information Theory **29**(5) (September 1983) 656–664

Table 1. Test 1: Encoding results using low-order FCM with fixed and variable order. In column *FCM* the encoding is done using fixed order FCM, in *FCM Limit* it is chosen the best order for each encoding symbol (used as a lower bound for comparison) and in *FCM Eff* we have the results using the proposed VOFCM. The best average results are presented in bold. **Test 2**: Comparison of encoding results using the proposed VOFCM as fallback method. Column *DNA3* and *FCM IR* show results from [6] and [8], respectively. Column *VOFCM* shows the result of the same method applied in *FCM-IR*, but using the proposed VOFCM as fallback method.

Name	Size	Test 1					Test 2				
		FCM			FCM	FCM	DNA3	FCM-IR		VOFCM	
		1	2	3	Limit	Eff	bps	Order	bpb	Order	bpb
y-1	230 203	1.958	1.954	1.950	1.862	1.951	1.871	11	1.909	11	1.913
y-4	1 531 929	1.947	1.944	1.939	1.867	1.944	1.881	12	1.910	12	1.918
y-14	784 328	1.953	1.949	1.945	1.870	1.950	1.926	12	1.938	12	1.946
y-mit	85 779	1.590	1.568	1.542	1.411	1.605	1.523	7	1.479	7	1.478
Average	–	1.938	1.934	**1.929**	1.852	1.935	**1.882**	–	1.904	–	1.912
m-7	5 114 647	1.933	1.924	1.922	1.837	1.917	1.835	12	1.835	12	1.833
m-11	49 909 125	1.935	1.925	1.923	1.828	1.918	1.790	13	1.778	13	1.776
m-19	703 729	1.944	1.932	1.928	1.828	1.920	1.888	10	1.873	10	1.869
m-x	17 430 763	1.921	1.913	1.911	1.827	1.911	1.703	13	1.692	13	1.691
m-y	711 108	1.918	1.910	1.909	1.826	1.907	1.707	11	1.741	11	1.742
Average	–	1.931	1.922	1.920	1.828	**1.916**	1.772	–	1.762	–	**1.761**
at-1	29 830 437	1.928	1.923	1.920	1.843	1.919	1.844	13	1.878	13	1.881
at-3	23 465 336	1.931	1.927	1.923	1.846	1.922	1.843	13	1.873	13	1.875
at-4	17 550 033	1.929	1.925	1.921	1.844	1.919	1.851	13	1.878	13	1.880
Average	–	1.929	1.925	1.921	1.844	**1.920**	**1.845**	–	1.876	–	1.879
h-2	236 268 154	1.924	1.916	1.913	1.819	1.910	1.790	13	1.734	13	1.731
h-13	95 206 001	1.918	1.910	1.907	1.816	1.906	1.818	13	1.759	13	1.760
h-22	33 821 688	1.942	1.929	1.922	1.806	1.915	1.767	12	1.710	12	1.703
h-x	144 793 946	1.922	1.914	1.911	1.821	1.910	1.732	13	1.666	13	1.665
h-y	22 668 225	1.918	1.910	1.908	1.812	1.900	1.411	13	1.579	13	1.577
Average	–	1.923	1.915	1.912	1.818	**1.909**	1.762	–	1.712	–	**1.710**
Total	–	1.925	1.917	1.914	1.822	**1.911**	1.772	–	1.735	–	**1.734**