EXPLORING DEEP MARKOV MODELS IN GENOMIC DATA COMPRESSION USING SEQUENCE PRE-ANALYSIS

Diogo Pratas and Armando J. Pinho

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

ABSTRACT

The pressure to find efficient genomic compression algorithms is being felt worldwide, as proved by several prizes and competitions. In this paper, we propose a compression algorithm that relies on a pre-analysis of the data before compression, with the aim of identifying regions of low complexity. This strategy enables us to use deeper context models, supported by hash-tables, without requiring huge amounts of memory. As an example, context depths as large as 32 are attainable for alphabets of four symbols, as is the case of genomic sequences. These deeper context models show very high compression capabilities in very repetitive genomic sequences, yielding improvements over previous algorithms. Furthermore, this method is universal, in the sense that it can be used in any type of textual data (such as quality-scores).

***Index Terms*—** Genomic data compression, hash-tables, finite-context models

1. INTRODUCTION

Genomic DNA sequences are usually represented by elements from an alphabet of four different symbols (called nucleotides or bases), namely, Adenine (A), Cytosine (C), Guanine (G), and Thymine (T), that store basic information of the living organisms. Nowadays, the genomics sequencing centers and the scientific community are being flooded with genomic data [1]. In spite of the possibility that a transformative breakthrough in storage technology occurs in the following years, the \$1000 genome milestone is most likely to arrive before the \$100 petabyte hard disk, mainly because the cost of disk storage is steadily decreasing over time, not matching the dramatic change in the cost and volume of sequencing. Therefore, compression is a key to cope with this problem, specially when the sequences share high homology rates.

Work supported in part by National Funds through FCT - Foundation for Science and Technology, in the context of the project PEst-OE/EEI/UI0127/2014, and by the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement No. 305444 "RD-Connect: An integrated platform connecting registries, biobanks and clinical bioinformatics for rare disease research".

Moreover, the study of data compression algorithms, besides the immediate aim of obtaining data reduction, provides a means for discovering the structure of the data [2,3]. In fact, the compression methods have underlying models that represent the data more efficiently. Hence, the better the compression, the better these models describe the information source associated to the data [4].

Genomic sequences, besides being very heterogeneous and non-stationary, have specific properties, such as inverted repeats [5], that led general purpose algorithms to be substituted by several special purpose methods. However, they only have started to be truly used almost after two decades of the first special propose algorithm, Biocompress [6], has been developed. Since Biocompress [6], several reference-free genomic compression algorithms have been proposed (see, for example, [7–18], and [19] for a recent review). With the advances of sequencing techniques, specific file formats emerged: FASTA (adding headers), FASTQ (mostly adding headers and quality-scores), and others, creating the need for specific compression tools [20–23]. Most of them rely on a combination of existing models, such as Markovian and Lempel-Ziv variants to handle diverse information channels.

From all the reference-free pure genomic compressors, the most successful, in compression ratio, seems to be XM [16], POMA [17] and DNAEnc3 [18]. Although XM and POMA present good compression results, they are impractical on current biological sequences (containing hundreds or thousands of MB), mainly because their memory and time requirements are too high to run on common computers. DNAEnc3 presents a better balance between running resources and compression results, being able to easily compress sequences with hundreds of MB in a few minutes and using less than 3 GB of memory. The method is based on multiple Markov models, with variable context orders, that compete to encode every block of the genomic sequence.

In this paper, we follow the line of DNAEnc3, exploring two competing Markov models with a low and high context order. However, unlike DNAEnc3, the proposed approach rely on deep context orders and on a preprocessing analysis to identify low complexity regions of the data. This strategy allows the reduction of memory usage and, consequently, allows to use deeper contexts that positively impact the compression gain. Moreover, we adopt a variable input alphabet, allowing this compressor to run in all textual sequence data (i.e., with alphabets until 256 symbols). Furthermore, it is very flexible, since it allows a variable multi-thread approach, defined by the user, as well as the possibility of compressing with hash-tables and decompressing with regular tables (or vice-versa).

The remainder of this paper is organized as follows. In Section 2, we present the method. In Section 3, we present the results for compression of a few genomic sequences. Finally, we draw some conclusions.

2. PROPOSED METHOD

2.1. Multiple finite-context models

Consider an information source that generates symbols, s, from a finite alphabet $\mathcal{A} = \{s_1, s_2, \ldots, s_{|\mathcal{A}|}\}$, where $|\mathcal{A}|$ denotes the size of the alphabet. In the case of DNA data, $\mathcal{A} = \{\mathrm{A}, \mathrm{C}, \mathrm{G}, \mathrm{T}\}$ and, therefore, $|\mathcal{A}| = 4$. Also, consider that the information source has already generated the sequence of n symbols $x^n = x_1 x_2 \ldots x_n,\ x_i \in \mathcal{A}$. A Markov model, also known as Finite-Context Model (FCM), assigns probability estimates to the symbols of the alphabet, regarding the next outcome of the information source, according to a conditioning context computed over a finite and fixed number, $k > 0$, of the most recent past outcomes $c = x_{n-k+1} \ldots x_{n-1} x_n$ (order-k FCM, with $|\mathcal{A}|^k$ conditioning states) [24–26].

The probability estimates, $P(X_{n+1} = s|c), \forall_{s \in \mathcal{A}}$, are usually calculated using symbol counts that are accumulated while the sequence is processed, which makes them dependent not only of the past k symbols, but also of n (i.e., these probability estimates are time varying).

The theoretical per symbol information content average provided by the FCM after having processed n symbols is given by

$$H_n = -\frac{1}{n} \sum_{i=0}^{n-1} \log_2 P(X_{i+1} = x_{i+1}|c) \quad \text{bps}, \qquad (1)$$

where "bps" stands for "bits per symbol".

Applying an estimator α, we address the information content estimation process under the form

$$P_\alpha(X_{n+1} = s|c) = \frac{n_s^c + \alpha}{n^c + \alpha|\mathcal{A}|}, \qquad (2)$$

where n_s^c represents the number of times that, in the past, the information source generated symbol s having c as the conditioning context and where n^c is the total number of events that has occurred so far in association with context c. Parameter α allows balancing between the maximum likelihood estimator and an uniform distribution. Note that when the total number of events, n, is large, the estimator behaves as a maximum likelihood estimator. For $\alpha = 1$, (2) is the Laplace estimator.

Genomic sequences are non-stationary. In fact, one of the reasons why most DNA encoding algorithms use a mixture of two methods, one based on repetitions and the other relying on low-order FCM, is to try to cope with the non-stationary nature of the data. We also follow this line of reasoning, using a low order FCM, typically 4, and a high order FCM that can be, in the case of DNA data (4 symbols), up to 32, where the high order context size depends on the size and repetitiveness of the sequence as shown in the results section.

Accordingly, we explore an approach based on two FCMs of different orders (low and high) that compete for encoding the data, allowing a better handling of DNA regions with diverse characteristics. Figure 1 shows an example where the two FCMs are used. In this example, each model collects statistical information from a context of depth $k_1 = 5$ and $k_2 = 11$, respectively. At time n, the two conditioning contexts are $c_1 = x_{n-k_1+1} \ldots x_{n-1} x_n$ and $c_2 = x_{n-k_2+1} \ldots x_{n-1} x_n$.

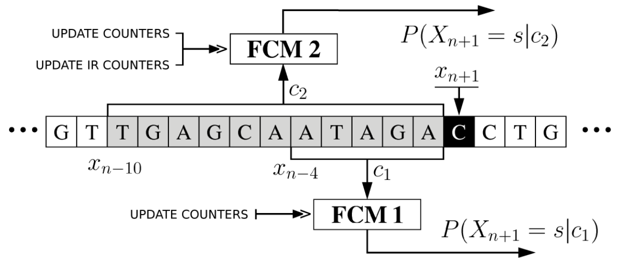

Fig. 1. Architecture of the two competitive FCMs. The probability of the next outcome, X_{n+1}, is conditioned by the k_1 or k_2 last outcomes, depending on the FCM chosen for handling a particular block. In this example, $k_1 = 5$ and $k_2 = 11$.

The competition between the two FCMs is held in the evaluation process for non-overlapping blocks of fixed size, such as 100 symbol blocks, which are then encoded by the best estimated FCM. The binary stream with the information of the respective FCM used in the compression of each block is encoded using an adaptive order-0 model followed by arithmetic coding.

2.2. Exploring high-order models using pre-analysis

Although the symbol counters for the low order FCM are constantly accumulated in a table with 16 bits of precision, the high order FCM requires two key approaches to maintain reasonable memory resources and the possibility of exploring deeper orders that might provide a better compression.

The first one is to use *sequence pre-analysis*, in order to classify the data blocks into low or high information content, reducing the number of them that otherwise would "pollute" the hash or counter table with incorrect statistics. Therefore, by applying this block-by-block analysis to the sequence, we are able to determine the blocks with low and high information content, and hence only update the high order model in the low information content blocks. As such, in the compression process, we can spend more memory than in the decompression process. If we do not have the resources on the compression side, it is also possible to estimate these low and high information content blocks by using a smaller context order, only marginally reducing the performance. However, we recall that the importance of resources is on the side of the decompression phase and this is ensured to be light on memory and time.

The second one, enhanced by pre-analysis, is to use high-order hash-tables (indexes up to 2^{64}), mainly because when implementing the FCM using simple tables the memory requirements grow exponentially with k. For DNA data, and considering 16 bits counters, this would imply about 39.4 zettabytes of memory for implementing an order-32 model. However, this table would also be very sparse, because the maximum number of different words of size k that can be found in a sequence of length n is upper bounded by n. Therefore, using hash-tables, it is possible to explore large order FCMs having an approximate increase of memory proportional to the size of the sequence if data are random. Repetitive data mitigate the memory consumption, which is usually the case for genomic data, given its repetitive nature (due to homologous genes, transposons, centromeres, telomeres, among others).

2.3. The encoding process

The symbolic sequence is processed from left to right (LR), in order to create a binary sequence representing the blocks that correspond to low information regions, as depicted in Fig. 2. After that, from right to left (RL), the block sequence is updated only when a block is marked as a low information region and when the block sequence had the previously index marked in the LR as a high information block. This corresponds to the maximum of the LR and RL cases.

Finally, the sequence block information is used to compress the sequence with the high-order FCM, which is associated to the lower information content region blocks, and the low-order FCM, that is associated to the higher information content region blocks. In practice, the high-order FCM is used to compress regions that can be evaluated as low information content regions by orders lower or equal to its own.

DNA sequences are characterized by a property known as the inverted repeats (IR) [5]. An inverted repeat is a subsequence of nucleotides that is the reversed complement of another sub-sequence. For example, the sub-sequence “ATAGAC” inverted repeat is known as “GTCTAT”. This particularity of DNA sequence data is used by most of the DNA specific compression methods, in spite of additional modeling performance (specially in high orders). Therefore, the IR are also explored in the high FCM (in the analysis and compression stages), where after encoding a symbol the respective sub-sequence IR counters are also updated in the same model. Specifically, the high FCM is constituted by two complete chains, the *regular* chain and the IR chain.

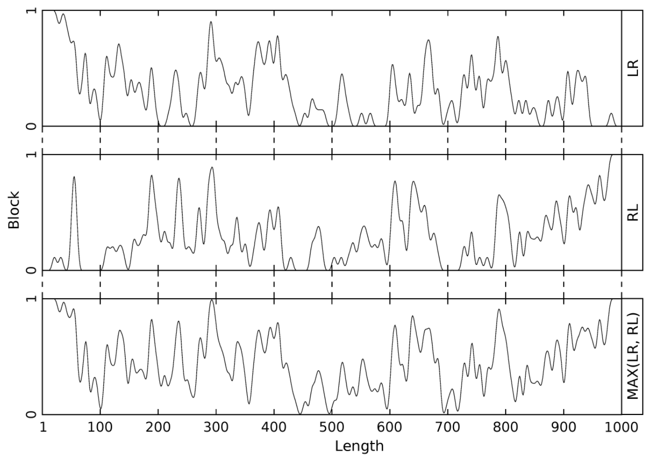

Fig. 2. Block sequence profiles from 100 KBases of the human chromosome 22, processed in left-to-right mode (LR) and right-to-left mode (RL), and computing the maximum of LR and RL. Block value 0 indicates a low-order FCM (order-4) while 1 a high-order FCM (order-16). The block size used was 100. Filtering: Blackman window of size 10.

3. EXPERIMENTAL RESULTS

The experiments have been performed on a Linux server running Ubuntu with 16 Intel(R) Xeon(R) CPU E7320 at 2.13 GHz and with 256 GB of RAM. The implementation, in the C programming language, is publicly available at `http://bioinformatics.ua.pt/software/highfcm/`. Three datasets have been used: Escherichia and Salmonella (bacterial) collections from NCBI and a collection of 20 human chromosomes 22 (very repetitive) from the 1000 genomes project.

We have estimated only the best high order, as shown in Fig. 3, where the compression ratios using different context orders of the dataset are presented. As depicted, the high context orders have a fundamental role in the compression, namely in the larger sequence, since the best compression ratio is achieved with the higher order (supported by the current implementation).

Relatively to benchmark results, as shown in Table 1, in the second and third sequences, the proposed approach has the best compression ratio, compared to other existing techniques, while it stands out in the last sequence (eukaryotic genomic collections) with almost 50% reduction relatively to DNAEnc3. Moreover, the memory spent and time usage, in all sequences, seems to be reasonable since the maximum is only slightly higher than 3 GB. General purpose

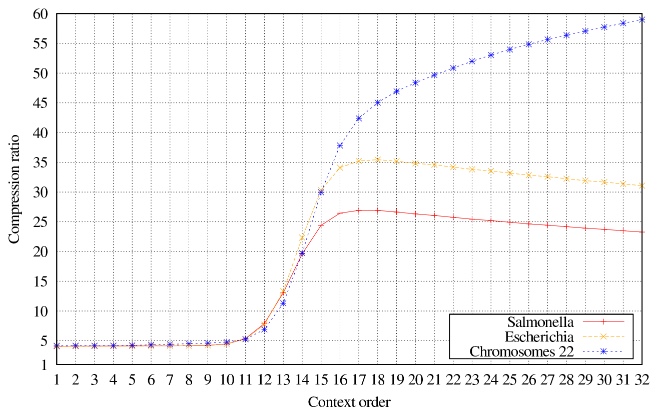

Fig. 3. Compression ratios as a function of the variation of the order of the deeper context, for all dataset sequences (static low order: 4).

algorithms (Gzip, Bzip2, lzma) are not capable of handling efficiently this type of sequences, while specific FASTA tools (MFCompress and Deliminate) seem to be between general purpose and pure reference-free algorithms (DNAEnc3, XM and HighFCM). Although the XM method had the best compression result in the first sequence (the smallest one), it spent huge amounts of memory and time (both in compression and decompression). Moreover, it was not able to handle larger sequences, returning a runtime error.

4. CONCLUSION

We have seen that applying preprocessing analysis techniques before compression can substantially improve the savings in memory resources, particularly in the decompression process. Moreover, these savings, together with appropriate high-order hash-tables, yield tremendous improvements in the compression ratio, specially in highly repetitive genomic sequences.

The proposed asymmetric compressor is also able to handle any textual sequences (up to 256 symbols). Furthermore, it is able to parallelize the tasks, allowing faster compression modes, at the expense of a small reduction in the compression ratio.

5. REFERENCES

[1] B. Berger, J. Peng, and M. Singh, "Computational solutions for omics data," *Nature Reviews Genetics*, vol. 14, pp. 333–346, May 2013.

[2] T. I. Dix, D. R. Powell, L. Allison, J. Bernal, S. Jaeger, and L. Stern, "Comparative analysis of long DNA sequences by per element information content using different contexts," *BMC Bioinformatics*, vol. 8, no. Suppl. 2, pp. S10, 2007.

[3] A. J. Pinho, D. Pratas, P. J. S. G. Ferreira, and S. P. Garcia, "Symbolic to numerical conversion of DNA sequences using finite-context models," in *Proc. of the 19th European Signal Processing Conf., EUSIPCO-2011*, Barcelona, Spain, Aug. 2011.

[4] M. Li, X. Chen, X. Li, B. Ma, and P. M. B. Vitányi, "The similarity metric," *IEEE Trans. on Information Theory*, vol. 50, no. 12, pp. 3250–3264, Dec. 2004.

[5] F. Lillo, S. Basile, and R. Mantegna, "Comparative genomics study of inverted repeats in bacteria," *Bioinformatics*, vol. 18, no. 7, pp. 971–979, 2002.

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

[7] 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.

[8] D. Loewenstern and P. N. Yianilos, "Significantly lower entropy estimates for natural DNA sequences," in *Proc. of the Data Compression Conf., DCC-97*, Snowbird, Utah, Mar. 1997, pp. 151–160.

[9] 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.

[10] 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.

[11] 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.

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

[13] 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.

[14] B. Behzadi and F. Le Fessant, "DNA compression challenge revisited," in *Combinatorial Pattern Matching: Proc. of CPM-2005*, Jeju Island, Korea, June 2005, vol. 3537 of *LNCS*, pp. 190–200, Springer-Verlag.

[15] G. Korodi and I. Tabus, "Normalized maximum likelihood model of order-1 for the compression of DNA

Table 1. Compression and decompression benchmarks. The memory has been estimated with *valgrind*, using *massif*, while running time with the *time* Linux program. It was not possible to obtain two results using XM, due to a program error (# stands for "Array index out of bounds").

Dataset	Method	Mode / parameters	Compression			Decompression	
			Size (B)	Time (s)	Memory (MB)	Time (s)	Memory (MB)
Collection of Salmonella	*Gzip*	-9 (best)	38,026,737	190	**6**	8	**6**
	Bzip2	-9 (best)	36,707,571	39	15	24	11
	Lzma	-9 (best)	6,285,003	364	383	**7**	47
size: 130 MB	*MFCompress*	-3 (best)	5,572,546	184	2,329	126	2,329
	Deliminate	-	11,199,769	**15**	8	19	8
	XM	500 experts	**3,480,673**	12,413	12,632	13,149	12,632
	DNAEnc3	1:4, 1:16 1/20, -ir	5,461,468	325	1,144	234	1,144
	HighFCM	4, 17 1/100 -ir	5,051,170	679	2,106	243	1,792
Collection of Escherichia	*Gzip*	-9 (best)	85,356,853	399	**6**	**9**	**6**
	Bzip2	-9 (best)	82,325,477	70	15	50	11
	Lzma	-9 (best)	14,240,205	2,150	383	12	47
size: 291 MB	*MFCompress*	-3 (best)	10,247,079	920	2,329	651	2,329
	Deliminate	-	19,156,665	**47**	656	45	68
	XM	500 experts	#	#	#	#	#
	DNAEnc3	1:4, 1:16 1/20, -ir	9,560,314	689	1,378	647	1,378
	HighFCM	4, 18 1/100 -ir	**8,615,784**	1,688	3,166	586	2,105
Collection of 20x human chromosomes 22	*Gzip*	-9 (best)	183,918,347	875	**6**	**18**	**6**
	Bzip2	-9 (best)	175,272,623	**138**	15	112	11
	Lzma	-9 (best)	151,390,802	2,594	107	35	46
	MFCompress	-3 (best)	29,983,772	973	2,329	639	2,329
size: 664 MB	*Deliminate*	-	39,775,033	531	711	95	67
	XM	500 experts	#	#	#	#	#
	DNAEnc3	1:4, 1:16 1/20, -ir	22,232,663	1,848	1,550	1,324	1,550
	HighFCM	4, 32 1/1000 -ir	**11,330,125**	1,821	3,087	625	1,770

sequences," in *Proc. of the Data Compression Conf., DCC-2007*, Snowbird, Utah, Mar. 2007, pp. 33–42.

[16] M. D. Cao, T. I. Dix, L. Allison, and C. Mears, "A simple statistical algorithm for biological sequence compression," in *Proc. of the Data Compression Conf., DCC-2007*, Snowbird, Utah, Mar. 2007, pp. 43–52.

[17] Z. Zhu, J. Zhou, Z. Ji, and Y. Shi, "DNA sequence compression using adaptive particle swarm optimization-based memetic algorithm," *IEEE Trans. on Evolutionary Computation*, vol. 15, no. 5, pp. 643–658, 2011.

[18] A. J. Pinho, P. J. S. G. Ferreira, A. J. R. Neves, and C. A. C. Bastos, "On the representability of complete genomes by multiple competing finite-context (Markov) models," *PLoS ONE*, vol. 6, no. 6, pp. e21588, 2011.

[19] Sebastian Wandelt and Ulf Leser, "Fresco: Referential compression of highly similar sequences," *IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB)*, vol. 10, no. 5, pp. 1275–1288, 2013.

[20] M. H. Mohammed, A. Dutta, T. Bose, S. Chadaram, and Sharmila S. Mande, "DELIMINATE - a fast and efficient method for loss-less compression of genomic sequences," *Bioinformatics*, vol. 28, no. 19, pp. 2527–2529, 2012.

[21] A. J. Pinho and D. Pratas, "MFCompress: a compression tool for fasta and multi-fasta data," *Bioinformatics*, Oct. 2013.

[22] Daniel C. Jones, Walter L. Ruzzo, Xinxia Peng, and Michael G. Katze, "Compression of next-generation sequencing reads aided by highly efficient de novo assembly," *Nucleic Acids Research*, vol. 40, no. 22, pp. e171, 2012.

[23] J. K. Bonfield and M. V. Mahoney, "Compression of FASTQ and SAM format sequencing data," *PLoS ONE*, vol. 8, no. 3, pp. e59190, Mar. 2013.

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

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

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