DNA Sequences Compression Algorithm Based on Extended-ASCII Representation

Bacem Saada, Jing Zhang

***Abstract*— The large amount of DNA sequences stored in databases has led researchers to propose compression algorithms for DNA sequences. The properties of the DNA sequences offer the opportunity to use a LOSSLESS algorithm. In this paper, we will present a two phases algorithm based on the binary representation of DNA sequences. In the first phase, we will compress the DNA sequences using the Extended-ASCII encoding through which one character encode four nucleotides. Thereafter, we will apply the Run Length Encoding technique to further enhance the compression of entire genomes. The simple way to implement the algorithm and its remarkable compression ratio make it interesting to be used.**

***Index Terms*—Extended-ASCII code, DNA compression, horizontal compression, vertical compression;**

I. INTRODUCTION

Nowadays, with the technological developments, the internet and the use of computers and devices connected together, the flow of data stored and transmitted between the different terminals has significantly increased. This boom has now incited researchers to talk about the Internet of Things. The IOT can be defined as "things belonging to the Internet to supply and access all of real-world information. Billions of devices are expected to be associated into the system and that shall require of huge distribution networks as well as the process of transforming raw data into Meaningful inferences". [1]

Thus, each day, a phenomenal amount of information is created, used, shared and analyzed by the different actors of the digital world. We are, now, in the era of the Big Data with a data flow that exceeds 8000 Exabytes and will reach around 40000 Exabytes in 2020.

This large amount of data requires powerful computers to properly analyze them and large databases to store them. Consequently, this has arisen two major problems. First, the encoding of the data that defines how it is stored and, second, the time required to process them. As a result, many data compression methods emerged to attempt to reduce data sizes. Compressors such as JPEG, MPEG and AVI are lossy compressors that try to remove some redundant information that humans cannot notice in images or in videos. Lossless compressors, on the second hand, compress data without loss. Therefore, they are used with text files and thus for the DNA sequences.

Technological evolution has led to the birth of the bioinformatics discipline which processes and analyzes the data of different living beings. The essential element in achieving these treatments is the Deoxyribonucleic Acid or DNA, which is a biomolecule present in all cells and in many viruses. This biomolecule contains all the genetic information called genotype that allows the functioning and development of all living beings. Each monomer that constitutes it is a nucleotide, which is composed of a nitrogenous base; adenine (A), cytosine (C), guanine (G) or thymine (T). GenBank is a free access database that contains a large amount of DNA sequences whose size increases exponentially. This database, which is managed by the International Nucleotide Sequence Database Collaboration, stores DNA sequences in raw format and may contain redundant data. For this reason, it is important to propose DNA sequences compression algorithms that reduce the size and so thoroughly analyze and choose the pertinent data that will be stored there.

In this article, we will start with a review of DNA sequences compression algorithms (Section II). In section III, we will present our approach to the compression of DNA sequences and explain how it can help to detect similarity regions between several sequences. Finally, in section IV, we will illustrate the achieved experiments and we will draw a comparison between our algorithm and other existing algorithms.

II. DNA SEQUENCES COMPRESSION ALGORITHMS

The compression of DNA sequences is based on the algorithms designed for text compression. The difficulty in applying those algorithms on DNA sequences is that first, the DNA sequences contain only 4 nucleotide bases {A, C, G, T} and second the existing regularity between these nucleotide bases in the sequence.

The researchers proved that conventional text compression algorithms are not suitable for DNA sequences compression. Based on the standard benchmark data [2] GZIP tool [3] for

Manuscript received July 01, 2015; revised July 21, 2015.

Bacem Saada, Ph.D. Student with Harbin Engineering University, College of Computer Science and Technology, Harbin, China, (email:basssoum@gmail.com).

Jing Zhang, Ph.D. Professor with Harbin Engineering University, College of Computer Science and Technology, Harbin, China, (email: zhangjing@hrbeu.edu.cn).

example has a compression ratio of 2.217 BpB. However a compression tool can only be considered good if the BpB is lower than two as there are only 4 nucleotides [4] and that can be represented by two bits. The Hoffman coding [5] is also not applicable because it is built on the basis of the probability of text language alphabet occurrences probability while the probability of nucleotides occurrence is almost identical.

There are two major classes of DNA sequences compression. The algorithms for DNA compression in horizontal mode and the algorithms for DNA Compression in vertical mode. The first is based on the compression of a single sequence based on its genetic information. For example, Biocompress [6] seeks repetitions and palindromes in a sequence. BIocompress-2 [7] uses a Markov model of order 2 to compress non-repetitive regions of a sequence. By applying these algorithms to the standard benchmark data, the compression ratio is 1.85 BpB for Biocompress and 1.78 BPB for biocompress-2. Therefore, they are better than conventional Lossless compression algorithms since the BpB rate is below two.

Some DNA sequences compression algorithms are based on the binary representation of the nucleotides (e.g. A = 00, C = 01, G = 10, T = 11). For example, GENBIT [8] divides sequences in a set of 8 bits and subsequently makes a 9th bit. If the block is identical to the above, the 9th bit is equal to 1, otherwise to 0. The compression ratio reaches then 1.125 BpB. DNABIT [9] divides the sequence into small blocks and compresses them while taking into consideration if they existed previously or not.

The second major class of DNA sequences compression algorithms analyzes the information existing in several sequences in order that one of these sequences will be representative of the whole set. For instance, DNAZIP package [10] has a series of algorithms that divide a genome into many blocks and compress them. LZ77 [11] proposes a compression technique for several genomes belonging to the same genus.

In other way, Biji, C. L., Madhu, M. K., and Vishnu, V. used parallel computing platform to compress large genomic datasets [12].

III. Our proposed algorithm

A. Description of the algorithm

Our algorithm is one of the algorithms that are based on the binary representation of nucleotides. It compresses the nucleotide bases in two bits. Thereafter, to reduce the size of the sequence, the bits will be converted to Extended ASCII coding which have an 8-bit character code. Finally, we will use our algorithm to detect regions of similarity between several DNA sequences.

B. Presentation of the algorithm

Through this section and in order to illustrate our algorithm's approach, throughout this section, we will use the following sequence as an example:

AGAA ATGT GACC GACC ATCT AGGC CAAT CGTT CACC ATCT

1.Encoding phase

a) Conversion to binary digit

The four nucleotides {A, C, G, T} will be encoded as follows:

A=00, C=01, G=10, T=11

The result of our example encoding will be as follows:

AGAA ATGT GACC GACC ATCT AGGC CAAT CGTT CACC ATCT

00100000 00111011 10000101 10001010 00110111 00101001 01000011 01101111 01000101 00110111

b) Conversion to Extended-ASCII code

In this step, the algorithm converts the series of binary numbers to decimal numbers. Thereafter the algorithm codes each number into its equivalent in ASCII code as shown in this figure (fig 1).

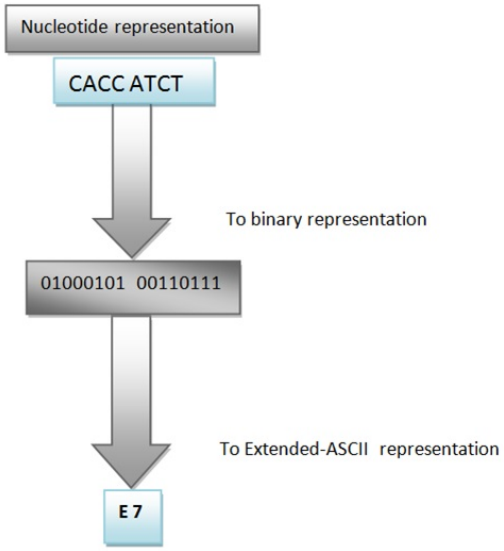

Fig. 1. Conversion to Extended ASCII code

Our example will be converted to Extended ASCII-coding as follows:

00100000 00111011 10000101 10001010 00110111 00101001 01000011 01101111 01000101 00110111

32 59 133 138 55 41 67 111 69 55

;…Š7)CoE7

ISBN: 978-988-14047-2-5
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

2. Decoding phase

The decoding phase is the inverse of the coding phase.

From an Extended-ASCII encoding, we will build an integer number that will be subsequently translated into a binary sequence. This bit stream will allow the building of the DNA sequence (Fig. 2).

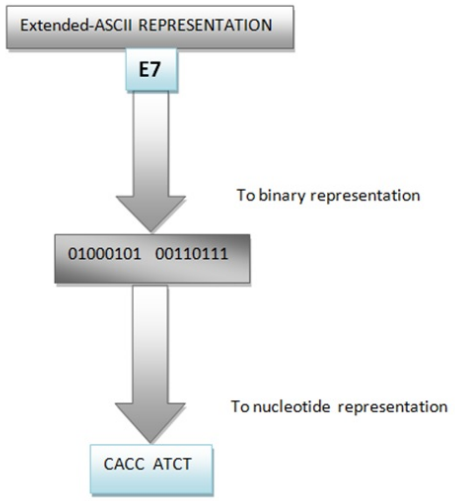

Fig. 2. Conversion to nucleotide representation

3. *The Use of the Run–Length Encoding algorithm*

The DNA sequences may have repeated sequences of nucleotides. To better compress the sequence, we apply the technique of Run-Length Encoding that detects similar adjacent regions and keeps only one instance of this block. However, an additional data structure is needed to keep the occurrence of these characters and the number of its repetition (fig.3).

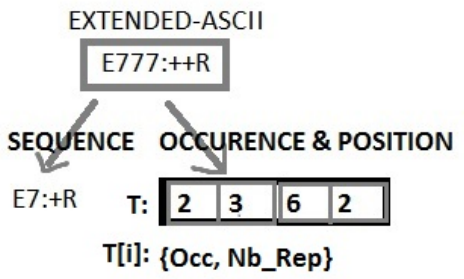

Fig. 3. RLE data structure

4. *Detection of similar blocks between multiple sequences*

The strength of the ASCII compression encoding is that one character code four nucleotides. By applying a searching process for similarities regions in the Extended- ASCII encoded DNA sequence, a common character between the compressed sequences means the existence of 4 common nucleotides (Fig. 4). The detection of similarity zones will be faster and more significant.

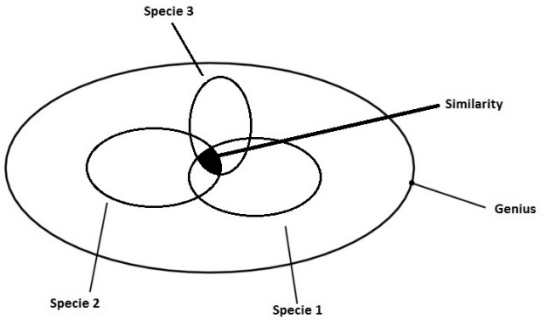

Fig. 4. Detection of similarity between species

IV. Experimental Results

A. Evaluation Metrics

To measure the performance of our algorithm, we used two types of data:

• Entire genomes: in order to calculate the contribution of our algorithm in terms of compression ratio to the genomes which have a large number of nucleotides.

• DNA sequences belonging to the same genus: this will, in addition to the compression of sequences, detect regions of similarity between the sequences after applying the EXTENDED-ASCII encoding.

B. Performance in terms of data compression

To achieve our experimental study, we used 11 species that belong to the genus Bacillus. The species used are amyloliquefaciens Anthracis, Azotoformans, Badius, Cereus, Circulans, coagulans, licheniformis, megaterium, mycoides, Psychrosaccharolyticus and pumilus. The size of a DNA sequence is about 1500 nucleotides. We also used the Mitochondrial genome (MPOMTCG) and the Vaccinia Virus genome (VACCG) whose size is about 190000 nucleotides.

Compression using the Extended ASCII coding has reduced the DNA sequence to a quarter of its original size. As indicated in table I, applying the RLE algorithm had allowed a gain up to 4% of the original size of the sequence. However, this gain was only for large size genomes.

We also compared our approach with existing DNA sequences compression algorithms in terms of binary representation rate per nucleotide.

Compression ratios shown in Table II demonstrate that most of the algorithms have a compression ratio higher than 1.7 BpB. Our algorithm provides better results and has a compression ratio equal to 1.65 BpB for the compression of the genome MPOMTCG.

ISBN: 978-988-14047-2-5
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

TABLE I. PERFORMANCE OF OUR ALGORITHM ON DIFFERENT DNA SEQUENCES AND GENOMES

Sequence Name	Number of Nucleotides	Sequence Size (Bytes)	Size after the Extended-ASCII Compression	Size after the RLE compression
Bacillus Subtilis	1538	1560	390	390
Bacillus Anthracis	1459	1480	370	370
Bacillus Cereus	1501	1522	381	381
MPOMTCG	186609	189274	47319	47287
VACCG	191737	194476	48620	48618

TABLE II. Comparison with other algorithms

Sequence	Base Pair	GZIP	DNA Compress	DNA Pack	CTW+LZ	XM	ASC-RLE
Subtilis	1538	2.30	1.92	1.91	1.92	1.87	1.68
MPOMTCG	186609	2.21	1.90	1.89	1.90	1.86	1.65
VACCG	191737	2.17	1.75	1.76	1.76	1.72	1.74

C. Experiments in Time execution

To measure the execution time of our algorithm, we used a computer with an Intel i3-2375M processor cadenced at 1.5 Ghz and a 4GB Ram memory.

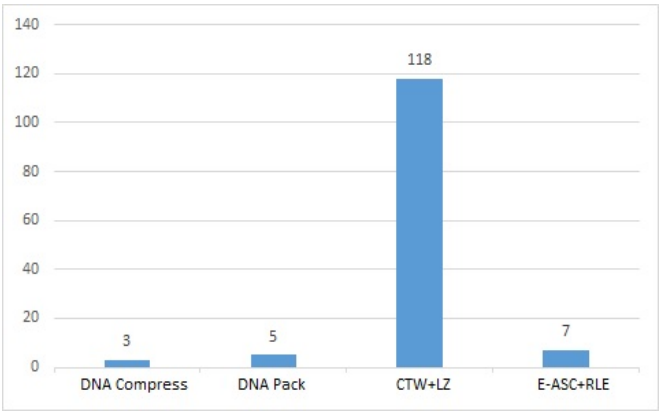

Fig. 5. Execution time comparison between our approach and other algorithms

The previous figure (fig.5) presents the execution time by applying the algorithms on the VACCG genome. It shows that our algorithm has an execution time better than the CTW+LZ algorithm. moreover, the execution time of our approach is greater than DNA Pack and DNA Compress. To better reduce our algorithm's execution time it is possible to use a data grid by parallelizing the execution of the algorithm.

D.Performance in terms of similarities percentages between sequences of the same genus

We applied our algorithm on the species of the genus Bacillus and the Phylum Firmicutes sequences. Subsequently, we looked for the longest common string between the sequences.

TABLE II. Longest common chain for a set of species

Sequences Species from	Length of the longest chain	length after compression
Bacillus genus	140	18
Firmicutes Phylum	85	11

From the results in Table III, we can say that in the case of applying our algorithm on sequences of the same genus, we will have a considerable gain in terms of data storage. In the case described in the table, 18 characters will suffice to describe the longest common string of 11 species of the genus Bacillus.

V. Conclusion and Future Work

The advantages of our algorithm is that it allows to have a compression ratio per base better than other compression algorithms and lower than 1.7 BpB. The algorithm is also very easy to implement and has an interest in the fact that a character in Extended-ASCII can encode four nucleotides. In the future, we will try to associate our algorithm to vertical compression algorithms based on statistical approaches to represent a set of DNA sequences or entire genomes in order to compress them with a rate higher than the rate of the current algorithms.

REFERENCES

[1] A survey of Internet-of-Things: Future Vision, Architecture, Challenges and Services

[2] S. G rumbach and F. Tahi, "Compression of DNA Sequences," in Proc. of the Data Compression Conf., (DCC '93), 1993, 340–350.

[3] Pierzchala, S. (2004). Compressing Web Content with mod—gzip and mod—deflate. Linux Journal, 1-10.

[4] Matsumoto, T., Sadakane, K., Imai, H., et al., 2000, Can General-Purpose Compression Schemes Really Compress DNA Sequences?, Computational Molecular Biology, Universal Academy Press, 76–77.

[5] Huffman, D. A. (1952). A method for the construction of minimum redundancy codes. Proceedings of the IRE, 40(9), 1098-1101.

[6] Grumbach S. and Tahi F.: Compression of DNA Sequences. In Data compression conference, pp 340-350. IEEE Computer Society Press, 1993.

[7]]Korodi, G., Tabus, I., Rissanen, J., et al., 2007, DNA Sequence Compression Based on the normalized maximum likelihood model, Signal Processing Magazine, IEEE, 24(1), 47-53.

[8] Grumbach, S., Tahi, F.: A new Challenge for compression algorithms: genetic sequences. Journal of Information Processing and Management 30, 866–875 (1994)

[9] A.AppaRao, "DNABIT compress-compression of DNA sequences," in Proc. the Bio medical Informatics, 2011.

[10] Ahmed, S., Brickner, D. G., Light, W. H., Cajigas, I., McDonough, M., Froyshteter, A. B., ... & Brickner, J. H. (2010). DNA zip codes control an ancient mechanism for gene targeting to the nuclear periphery. Nature cell biology, 12(2), 111-118.

ISBN: 978-988-14047-2-5
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

[11] Ahmed, S., Brickner, D. G., Light, W. H., Cajigas, I., McDonough, M., Froyshteter, A. B., ... & Brickner, J. H. (2010). DNA zip codes control an ancient mechanism for gene targeting to the nuclear periphery. Nature cell biology, 12(2), 111-118.

[12] Biji, C. L., Madhu, M. K., & Vishnu, V. (2015). Compression of Large genomic datasets using COMRAD on Parallel Computing Platform. Bioinformation, 11(5), 267.

ISBN: 978-988-14047-2-5
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)