Vertical DNA Sequences Compression Algorithm Based on Hexadecimal Representation

Bacem Saada, Jing Zhang

***Abstract*— The number of DNA sequences stored in databases is growing exponentially. Thus, decreasing the Data storage costs for the creation and the analysis of DNA sequences has become a necessity. Standard compression algorithms failed to get a good compression ratio. In this paper, we describe a LOSSLESS algorithm that compresses the DNA sequence in its equivalent in hexadecimal representation. In addition, by using a mathematical operation, it identifies the regions of similarity between two sequences.**

Index Terms—*DNA, encode, Hexadecimal Representation, Big Data, Horizontal compression, Vertical compression;*

I. Introduction

Bioinformatics is a field that combines mathematics, statistics, physics, chemistry and biology. It solves the problems that biologists face in the agricultural, medical science and study of the living world. The most important element in bioinformatics is the study of biomolecules present in all living cells that are the DNA sequences. DNA contains all the genetic information, called genotype, enabling the development and functioning of living beings. Each element constituting it is a nucleotide, which is formed of a nitrogenous base - adenine (A), cytosine (C), guanine (G) or thymine (T).

The scientific community has developed data banks where biologists can store, analyze and search for similarities between the DNA sequences and the strains they identified as well as classify DNA sequences by degree of similarity. GenBank is a free access database that contains a large amount of DNA sequences whose size increases exponentially. This database, which is headed by the International Nucleotide Sequence Database Collaboration [1], stores DNA sequences in their raw format and may contain redundant data. It is thus imperative to propose compression algorithms of DNA sequences to reduce the size and to well analyze and select which data should be stored in the database.

Since the 2000s, the reduction in sequencing costs and the growth of data storage discs sizes resulted in the implementation of several projects on the sequencing of entire genomes. Among these projects, we can mention the 1000 Genomes Project and the Human Arabidopsis thaliana Project in 1001 [2]. The genome size is very large; to humans it reached 3 billion nucleotides and can even reach more than 100 billion nucleotides for certain amphibian species [3].

This large amount of data requires powerful computers to properly analyze them and large databases to store them. This induces us to have two main problems. The first is the encoding of the data that defines how it is stored. The second is the time required to process them. This has led to the development of many data compression methods to attempt to reduce their sizes. Compression techniques known as lossless compressors compress data without any loss. Therefore, they are used to compress text files and thus the DNA sequences.

In this article, in Section II, we will present the DNA sequences compression algorithms. In Section III, we will present our approach to DNA sequences compression and explain how it is useful in comparing DNA sequences. Finally, in section IV, we will present the experiments made and we will draw a comparison with existing algorithms.

II. DNA sequences Compression algorithms

Compression of DNA sequences uses algorithms made for compressing text and alphabets. The difficulty in the application of the algorithms on the DNA sequences is that the DNA sequences contain only 4 nucleotide bases {A, C, G, T} for which the probability of the occurrence in the DNA sequence is substantially identical .

There are two major classes of DNA sequences compression. The compression algorithms for DNA in horizontal mode and the Compression Algorithms for DNA in vertical mode. The first class compresses a single sequence based on its genetic information. Biocompress [4], for instance, compresses the repetitions and palindromes in a sequence. BIocompress-2 [5] is based on a Markov model to compress the non-repetitive

Manuscript received July 01, 2015; revised July 20, 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).

regions of a sequence. To calculate the efficiency of an algorithm, it is necessary to calculate the compression ratio of a nucleotide-by-bit. The rate is calculated as follows: Ratio = Bits / Bases. By applying these algorithms to the standard benchmark data [6], the compression ratio is of 1.85 BPB for Biocompress and 1.78 BPB for biocompress-2.

Some compression algorithms are based on the binary representation of DNA sequences (eg, A = 00, C = 01, G = 10, T = 11). For example, GENBIT [7] divides a sequence into blocks of 8 bits and introduces a 9th bit at the end of this sequence. If the block is identical to the previous one, the 9th bit will be equal to 1, otherwise 0. The compression ratio reaches 1.125 / BpB. DNABIT [8] divides the blocks in sequences and compresses them while taking into account if they have been previously detected or not.

The second type of algorithms processes a vertical compression of DNA sequences that is a compression of a set of sequences by analyzing all their genetic information in order to detect one of these sequences that will be representative of the whole. LZ77 [9] proposes a compression algorithm of several genomes belonging to the same genus. DNAZIP package [10] has a series of algorithms that divide a genome into a set of blocks and compress them later.

III. Our Proposed Algorithm

A. Description of the algorithm

Our algorithm is based on the binary representation of nucleotides. It compresses the nucleotides in two bits. Thereafter, to reduce the size of the sequence, the bits will be converted into a hexadecimal representation whose character codes two nucleotides. Finally, our algorithm will be used to detect regions of similarity between several DNA sequences.

B. Overview of the algorithm

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

B.1 Encoding phase

STEP 1: Conversion to binary digit

The four nucleotides {A, C, G, T} will be coded as follows: A=00, C=01, G=10, T=11

The result of the encoding of our example will be as follows:

AGAA ATGT GACC GACC ATCT AGGC CAAT CGTT CACC ATCT

00100000 00111011 10000101 10001010 00110111 00101001 01000011 01101111 01000101 00110111

STEP 2: Hexadecimal conversion

In this second step, the algorithm converts the result of the binary numbers in hexadecimal numbers. (Fig. 1).

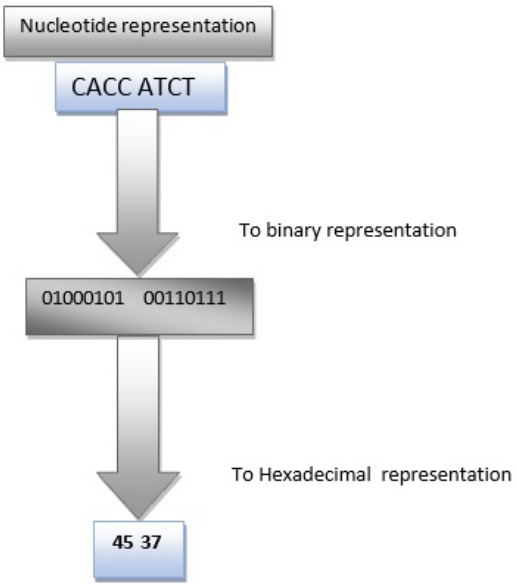

Fig. 1. Conversion to Hexadecimal representation

Our example will be converted to hexadecimal representation as follows:

00100000 00111011 10000101 10001010 00110111 00101001 01000011 01101111 01000101 00110111

20 3B 85 8A 37 29 43 6F 49 37

B.2 Decoding phase

The decoding phase is the inverse of the encoding phase.

From a hexadecimal representation, a hexadecimal number will be converted into a binary sequence. This bit sequence will build the DNA sequence (Fig. 2).

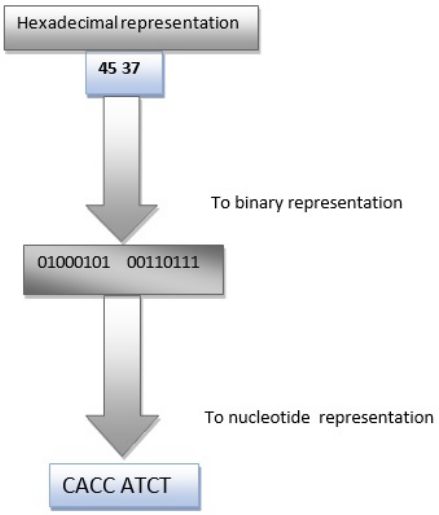

Fig. 2. Conversion to nucleotide representation

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

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

The DNA sequences may have repetitive blocks of nucleotides. To better compress the sequence, it is possible to apply the technique of Run-Length Encoding that detects similar adjacent areas and keeps only one copy of this block. All the same, an additional data structure must be used to keep the occurrence of these characters and the number of repetition (fig.3).

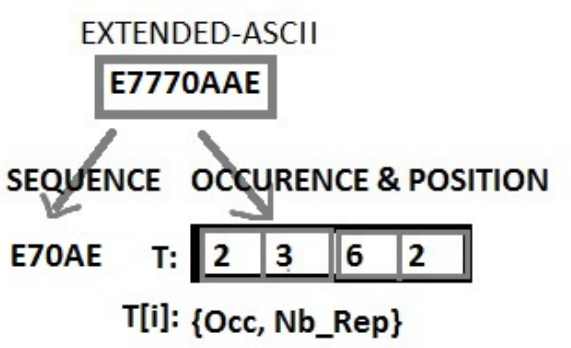

Fig. 3. RLE data structure

B.4 Detection of similar blocks between multiple sequences

By applying a searching process for similar regions on multiple DNA sequences encoded in binary representation, it is possible to detect similarities between them by using an hexadecimal representation. It is easy to make a subtraction operation between these sequences. The obtained result is then converted to binary coding. Consequently, the bits equal to zero represent the regions of similarity between the sequences (Fig. 4).

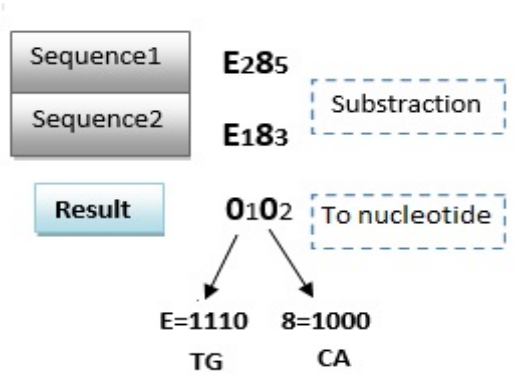

Fig. 4. Detection of similarity between multiple species

IV. Experimental results

A. Evaluation Metrics

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

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

• The DNA sequences belonging to the same genus: this will, in addition to the compression of sequences, detect regions of similarities between the sequences after applying the Hexadecimal encoding.

B. Performance in terms of data compression

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

The compression using the hexadecimal coding has reduced the DNA sequence's original size to the half. As indicated in table I, applying the RLE algorithm has permitted a gain of up to 4% of the sequence's original size. However, this gain is only perceived in the genomes' compression.

We also compared our approach to existing DNA sequences compression algorithms in terms of binary representation rate per nucleotide. Compression ratios presented in Table II show

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

Sequence Name	Number of Nucleotides	Sequence Size (Bytes)	Size after the Hexadecimal Compression	Size after the RLE compression
Bacillus Subtilis	1538	1560	780	780
Bacillus Anthracis	1459	1480	740	740
Bacillus Cereus	1501	1522	761	761
MPOMTCG	186609	189274	94637	94601
VACCG	191737	194476	97238	97237

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

TABLE II. Comparison with other algorithms

Sequence	Base Pair	GZIP	DNA Compress	DNA Pack	CTW+LZ	XM	Hex-RLE
Subtilis	1538	2.30	1.92	1.91	1.92	1.87	2
MPOMTCG	186609	2.21	1.90	1.89	1.90	1.86	1.98
VACCG	191737	2.17	1.75	1.76	1.76	1.72	1.99

that the majority of algorithms have a compression ratio greater than 1.7 BpB including our algorithm which has a compression rate equal to 2 BpB.

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.

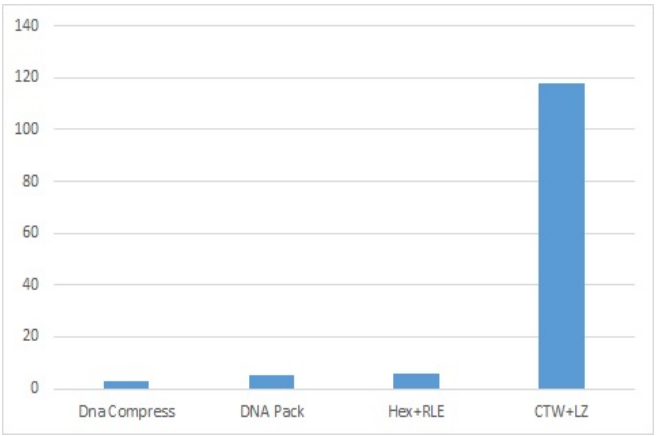

Fig. 5. Execution time comparison between our approach and other algorithms

The previous figure (fig.5) represents the execution time of algorithms applied on the VACCG genome. It shows that our algorithm has a running time better than the CTW + LZ algorithm. All the same, the execution time is greater than that of DNA and DNA Pack Compress. However, it is possible to reduce the execution time of our algorithm if its execution is parallelized.

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

Our algorithm permits an easy search of regions of similarity of a set of DNA sequences. Indeed the transformation of the sequences to the hexadecimal representation is followed by a simple subtraction operation and a conversion of the result into binary representation and a detection of adjacent zero suites that represent the regions of similarity between the sequences. To test our algorithm, we applied it on Bacillus Anthracis and Bacillus Subtilis sequences (Tab III)

TABLE III. Number of similarity between Sequences

Number of similarity (Nucleotide)	331
Number of similarity (Hexadecimal characters)	134
the longest common subchain (nucleotide)	190
the longest common subchain (Hexadecimal characters)	95

Our algorithm has detected the longest subchain between the two sequences.
Second, we have 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 IV. Longest common chain for a set of species

Sequences from	Length of the longest chain	Length after compression
Bacillus genius	140	70
Firmicutes Phylum	85	42

From the results in Table IV, we can say that our algorithm has allowed a considerable gain in terms of data storage. In addition, 70 Hexadecimal characters are only used to describe the longest common chain of 11 species of the genus Bacillus used in this experiment.

V. Conclusion and Future Work

The main strength of our algorithm is that it allows, using a simple arithmetic operation, to compare a set of DNA sequences and identify regions of similarity between them. The algorithm is also very easy to implement and his usage is advantageous because a hexadecimal character code two nucleotides. In the future, we will try to associate our algorithm to other vertical compression algorithms based on statistical approaches in order to choose the sequence that can best represent a set of DNA sequences.

REFERENCES

[1] A.AppaRao, "DNABIT compress-compression of DNA sequences," in Proc. the Bio medical Informatics, 2011.
[2] Rajeswari, P. R., and Apparao, A., 2010, Genbit Compress Tool (GBC)Java-Based Tool To Compress DNA Sequences and Compute Compression Ratio (BITS/BASE) Of Genomes, International Journal of Computer Science and Information Technology, 2(3), 181-191.
[3] Pierzchala, S. (2004). Compressing Web Content with mod—gzip and mod—deflate. Linux Journal, 1-10.

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

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

[5] 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.

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

[7] Rajeswari, P. R., and Apparao, A., 2010, Genbit Compress Tool (GBC): A Java-Based Tool To Compress DNA Sequences and Compute

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

[9] Deorowicz, S., and Grabowski, S., 2011, Robust relative compression of genomes with random access, Bioinformatics, 27(21), 2979–2986.

[10] Christley, S., Lu, Y., Li, C., et al., 2009, Human genomes as email attachments, Bioinformatics, 25(2), 274-275

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