American Journal of Bioinformatics Research 2013, 3(3): 72-81
DOI: 10.5923/j.bioinformatics.20130303.04

DNA Lossless Compression Algorithms: Review

Nour S. Bakr[1,*], Amr A. Sharawi[2]

[1]Department of Biomedical Engineering, Higher Technological Institute, 10th of Ramadan City, Egypt
[2]Department of Systems and Biomedical Engineering, Cairo University, Cairo, Egypt

Abstract Modern DNA sequencing instruments are able to generate huge amounts of genomic data. Those huge volumes of data require effective storage, fast transmission, provision of quick access to any record, and superior functionality. Data storage costs have an appreciable proportion of total cost in the creation and analysis of DNA sequences. In particular, the increase in the DNA sequences is highly manageable due to a tremendous increase in the disk storage capacity. Standard compression techniques failed to compress these sequences. Recently, new algorithms have been introduced specifically for this purpose. In this paper, we comparatively survey the main ideas and results of lossless compression algorithms that have been developed for DNA sequences.

Keywords DNA, Codon, High Throughput Sequencing, Re-sequencing, Horizontal Compression, Vertical Compression

1. Introduction

DNA strings are composed of an alphabet of four characters: A, T, G, and C. Each three letter sub-strings within the DNA sequences are called codons (e.g. AGC and GCT). There are 64 recognized codons. These produce 20 different amino acids because different codons can produce the same amino acid[1]. An organism has in each one of its cells the same DNA sequences making up the individual's genome, which for higher life forms are organized in many chromosomes[2]. In a human DNA structure there are so many unknown nucleotides. These unknown nucleotides are represented by N (space) so, the human genome structure consists of five characters A, C, G, T, and N[3].

Some characteristics of DNA sequences show that they are not random sequences. If these sequences were totally random, the most efficient and logical way to store them would be using two bits per base. However, DNA is used for the expression of proteins in living organisms, and thus must contain some logical organization[4]. There are many repeats within a given DNA sequence (e.g. AAACGT), which may occur more than once in a given DNA sequence. However, there may exist an approximate repeat of AAACGT in this DNA sequence (e.g. AAACGG and ACGCGT). In DNA A and T are complements of each other. So do G and C. The complement of the DNA sequence AAACGT would be TTTGCA. If we then reversed TTTGCA we would get ACGTTT. ACGTTT is defined as being the reverse complement or "palindrome" of AAACGT.

* Corresponding author:
bioeng_nour@ieee.org (Nour S. Bakr)
Published online at http://journal.sapub.org/bioinformatics

DNA sequences within the same species are both large and highly repetitive. Consider that only approximately 0.1% of the 3GB human genome is specific; the rest is common to all humans[5].

The development of new DNA sequencing technologies, such as next-generation sequencing (NGS) and single molecule sequencing, has enabled the research of genomics and functional genomics to advance to new levels[6, 7]. Due to the dramatic reduction of sequencing cost and increase of sequencing efficiency, these new high-throughput sequencing technologies have led to a large number of whole genome DNA sequencing projects. Some of these projects, such as the Genome 10K project, take advantage of the improved sequencing speeds and costs to obtain genomes of species that are nowadays un-sequenced. Still other projects focus on re-sequencing, where individual genomes from a given species are sequenced to understand variation between individuals. Examples are the 1000 Genomes project for humans and the 1001 Genomes project for the plant Arabidopsis thaliana[8]. The size of the genomes varies wildly in range; viruses may have only few thousand symbols, bacteria millions of symbols, humans about 3 billion and some amphibian species have even 120 billion bases[2].

Sequencing labs submit their data to big archival databases such as GenBank at the National Center for Biotechnology Information (NCBI)[9], the European Bioinformatics Institute EMBL database[10], the DNA Data Bank of Japan (DDBJ)[11], the Short Read Archive (SRA)[12], the Gene Expression Omnibus (GEO)[13], and the microarray database Array Express[14]. These databases maintain, organize, and distribute the sequencing data. The three partners (GenBank, DDBJ, and EMBL Bank) of the International Nucleotide Sequence Database Collaboration

(INSDC) set out to capture, preserve, and present the permanent scientific record for nucleic acid sequencing and associated information[15, 16]. Records in these databases are synchronized according to guidelines from the INSDC [17], which assures that each database contains a copy of the others.

Collecting and understanding genomic data for all organisms and their species has become the corner stone of modern life science research. In basic research, genomics data leads to better understanding of the cellular functions and organism evolution[18]. In biomedical applications, it is used for studying the molecular and genetic basis of diseases and their variations, for discovering and designing modern therapeutics as well as for developing modern diagnostic and prognostic tools[19]. In agriculture and food studies, it is used to help breed better species of plants and livestock as well as for studying pathogen interactions[20, 21].

The assembled sequences from the sequencing projects can range from terabytes to petabytes in size[8]. Approximately 154,192,921,011 bases (154.2 billion) from 167,295,840 (167.3 million) reported sequences were there in the GenBank database in August 2013[22], the database size getting two or three times bigger annually. Therefore efficient lossless compression techniques and data structures to efficiently store, access, communicate, and search these large datasets are necessary.

The standard compression techniques cannot compress the biological sequences well. These algorithms do not use special structures of biological sequences. Recently, several algorithms have been proposed for the compression of DNA sequences based on DNA sequence special structures.

Grumbach and Tahi in[23] propose two modes for the compression of DNA sequences:

- Horizontal mode: one is given a biological sequence, which is compressed by making use of information contained only in the sequence, typically by making reference only to its substrings. Evaluation of compression methods is usually performed in this mode.
- Vertical mode: One is given a set of biological sequences, and each sequence is compressed by making use of information contained in the entire set, typically the substrings of the set. It may lead to an organization of the data related similarities between the species.

In the following sections we will discuss the algorithms for compression of DNA sequences in the two modes.

There have been recent publications by Giancarlo et al.[24] and Nalbantoglu et al.[25], which also examine some of the literature reviewed here. However, the papers take somewhat different approaches to the literature. Giancarlo et al. examine the various problems of bioinformatics and describe compression tools that have been used to address these problems. Nalbantoglu et al. examine the different concepts in compression and look at how these concepts can be used to understand and resolve issues in computational biology and bioinformatics.

In this paper we survey the main issues and results of lossless compression algorithms developed for DNA sequences with a somewhat comparative view. The current paper would be beneficial to individuals interested in DNA compression tools for keeping up on their field and may be most useful for those who are not experts in this field.

The content of the paper is organized as follows: In section 2 we report the general compression algorithms and its performance with DNA sequences. In section 3 we present the horizontal mode of special algorithms for DNA compression and its performance, while the vertical mode is presented in section 4. Finally, in section 5, we conclude several research ideas to be explored further relating to DNA sequences compression.

2. Standard Algorithms for DNA Compression

The compression of a DNA sequence is a difficult task for general compression algorithms because these algorithms are designed mainly for English text compression, while the regularities in DNA sequences are very small.

DNA sequences contain only four bases {A, C, T, G}. Thus, each base (symbol) can be represented by two bits. However, the standard text compression tools, such as compress (Lempel–Ziv–Welch "LZW"), gzip (Lempel–Ziv "LZ" + Huffman) and bzip2 (Burrows–Wheeler transform "BWT" + Move-to-Front "MTF" + Huffman), cannot compress these DNA sequences[4]. Using standard benchmark data[1], the average compression ratio is 2.185 bpb "bits per base" for compress, 2.271 bpb for gzip, and 2.138 bpb for bzip2 except the sequence "HUMGHCSA", where the average compression ratio is 1.729 bpb[26].

Huffman's code also fails badly on DNA sequences both in the static and adaptive model, because the probabilities of occurrence of the four symbols are not very different[27]. Prediction with Partial Match (PPM) cannot compress DNA sequences having less than two bits per symbol either[27]. On the other hand, arithmetic coding and Context Tree Weighting (CTW) are known to compress the DNA data well[27]. In spite of the good compression ratio, arithmetic coding and CTW have disadvantages, such as low decompression speed[28].

3. Algorithms for DNA Compression in Horizontal Mode

This mode uses the information contained only in the sequence, typically by making reference only to its substrings.

[1] Most DNA compression algorithms use the standard benchmark data in[23]. These standard sequences, come from a variety of sources and include the complete genomes of two mitochondria (MPOMTCG, PANMTPACGA "also called MIPACGA"), two chloroplasts (CHNTXX and CHMPXX "also called MPOCPCG"), five sequences from humans (HUMDYSTROP, HUMGHCSA, HUMHBB, HUMHDABCD and HUMHPRTB), and finally the complete genome from two viruses (VACCG and HEHCMVCG "also called HS5HCMVCG").

Most of the compression methods used today including DNA compression falls into the following categories:

3.1. Substitutional Based Methods

This algorithm compresses data by replacing long sequences by short pointer information to the same sequences in a dictionary.

The first special purpose DNA compression algorithm is BioCompress developed by Grumbach and Tahi[23] in 1993. This technique is based on the sliding window algorithm LZ. First, it detects exact and reverse complement repeats in the DNA and stores it using 4-ary tree, and then it encodes them by the repeat length and the position of a previous repeat occurrence. For non-repeat regions, it is encoded by 2 bpb. The improved version, BioCompress 2[29] is similar to BioCompress, but uses order-2 arithmetic coding to encode non-repeat regions. The results showed that both algorithms compressed the standard benchmark data with an average compression ratio of 1.850 bpb for Biocompress and 1.783 bpb for Biocompress 2, compared to the general-purpose algorithms compact and compress, which used more than 2 bpb.

The Cfact[30] is similar to Biocompress and uses a two-pass algorithm to search for the longest exact and reverse complement repeats. For this purpose, it builds the suffix tree of the sequence in the first pass and does the actual encoding using LZ in the second pass. Non-repeat regions are also encoded by 2 bpb. There are no compression results about this algorithm; therefore it is difficult to compare. The results may be similar to Biocompress, but they take more compression time, since Cfact uses two passes, and constructs a suffix tree.

A similar substitution approach by Chen et al. is used in GenCompress[31], except the use of approximate repetitions. An inexact repeat subsequence is encoded by a pair of integers, as for BioCompress-2, and a list of edit operations for mutations. It exists in two variants: GenCompress-1, which uses the Hamming distance (only substitutions) for the repeats and GenCompress-2, which uses the edition distance (deletion, insertion and substitution) for the encoding of the repeats. While GenCompress obtains better compression ratios (with average 1.742 bpb - standard benchmark data) than BioCompress-2 and Cfact, it is not suitable for compressing large sequences.

Chen et al. made further a modification of GenCompress that led to a two-pass algorithm DNACompress[32]. During the first pass, GenCompress finds all approximate repeats including complementary palindromes, using specific software called PatternHunter[33]. Then the approximate repeats and the non-repeat regions are encoded using the LZ compression scheme. DNACompress produces a slightly better compression ratio (with average 1.725 bpb - standard benchmark data) with faster compression than GenCompress and CTW + LZ algorithms. DNACompress was able to deal with large sequences (e.g. E. coli with about 4.6 Megabases) in about a minute, where GenCompress required nearly about half an hour.

Searching for approximate repeats takes a long time and requires a large amount of memory. For this reason Manzini and Rastero in DNA-X algorithm[34] found it natural to investigate how much compression they can achieve working only with exact and reverse complement repeats. It is a single-pass algorithm and operates in a so called LZ77 manner. The algorithm only encodes exact repeats, but in designing the code words that represent exact repeats they use the knowledge that some of these exact repeats are "fragments" of larger approximate repeats. Like most of the other methods, this technique also uses fall back mechanisms for the regions where matching fails, in this case finite-context arithmetic coding of order-2 (DNA2) or order-3 (DNA3)[35]. This algorithm is faster than any other algorithm and achieves a compression ratio very close to the best DNA compressors. The algorithm is memory efficient and occupies small space, which leads to the compression of hundreds of megabytes.

Lee et al. introduced a DNAC algorithm in 2004[36] as an update of the Cfact algorithm, but it works in four phases: During the first phase, it builds a suffix tree to locate exact repeats. During the second phase, all exact repeats are extended into approximate repeats by dynamic programming. In the third phase, it extracts the optimal non-overlapping repeats from the overlapping ones, and in the last phase, it uses the Fibonacci encoding method to encode the repeats in a self-delimited way. The authors compared experimental results with the GenCompress algorithm, since the GenCompress outperforms the Cfact algorithm and shows that the DNAC algorithm outperforms the GenCompress algorithm by several orders of magnitude. Moreover, it can be used to compress long sequences with millions bases or more.

Behzadi and Le Fessant in DNAPack[4] found a better set of repeats than those found by GenCompress and DNACompress by using dynamic programming instead of greedy approaches as others do. It uses the Hamming distance (only substitutions) for the repeats and inverted repeats. Non-repeat regions are encoded by the best choice from an order-2 arithmetic coding, context tree weighting, and naive 2 bits per symbol methods. DNAPack consistently performed better than the Biocompress-2, GenCompress, CTW+LZ and DNA Compress in terms of average compression ratio (1.714 bpb - standard benchmark data). It is expected that the algorithm will be slow due to the expensive calculation, and will not be suitable for long sequences.

3.2. Statistical Based Methods

There are many statistical methods for DNA compression, such as CDNA, Approximate Repeats Model (ARM), expert-Model (XM), and the finite-context model algorithm by replacing a more popular symbol by a shorter code.

Loewenstern and Yianilos introduced the first algorithm to combine statistical compression with approximate repeat

for DNA compression called CDNA algorithm[37].

It is based on the probability distribution of each symbol that is obtained by approximate partial matches from history. Each approximate match is applied to a previous subsequence having a small Hamming distance to the context preceding the symbol to be encoded. Predictions are combined using a set of weights, which are learnt adaptively.

The ARM algorithm is also a pure statistical DNA compressor proposed by Allison et al. in 1998[38] that forms the probability of a subsequence by summing the probabilities over all explanations of how the subsequence is generated. The ARM and CDNA algorithms yield significantly better compression ratios than those in the substitutional classes and can also produce information content sequences.

The last pure statistical DNA compressor is the XM algorithm introduced by Cao et al. in 2007[39], which relies on a mixture of experts for providing symbol by symbol probability estimates which are then used for driving an arithmetic encoder. The algorithm comprises three types of decision-support systems: (1) order-2 Markov models; (2) order-1 context Markov models, i.e., Markov models that use statistical information only of a recent past (typically, the 512 previous symbols); (3) the copy expert that considers the next symbol as part of a copied region from a particular offset. The probability estimates provided by the set of experts are then combined using Bayesian averaging and sent to the arithmetic encoder. The compression results of XM are excellent (average compression ratio 1.714 bpb - standard benchmark data).

All these algorithms are computationally intensive, which makes them practical only for compressing small sequences.

Armando et al. in[35] provide a finite-context model[2] for DNA compression, which is based on two finite-context models of different orders that compete for encoding the sequence. In this model several aspects have been addressed, such as the inclusion of mechanisms for handling inverted repeats and the use of multiple finite-context models that compete for encoding the data. This algorithm provides significant results, although not as expressive as those provided by methods such as NML-1[2] or XM[39]. Nevertheless, the experimental results show that this approach can outperform methods of similar computational complexity, such as the DNA3 coding method[34]. No doubt, this algorithm has proved to give good returns in terms of compression gains, but normally at the cost of long compression times.

3.3. Substitutional and Statistical Based Methods

Several DNA compression algorithms combine substitution and statistical styles. An inexact repeat is encoded using a pointer to a previous occurrence, and the probabilities of symbols are copied, changed, inserted or deleted.

[2] The finite-context model takes a few immediately preceding symbols into account to make a prediction.

The Off-line algorithm introduced by Apostolico and Lonardi in 2000[40-41], uses repeated regions for compression. Only exact repeats are considered during each iteration, and the algorithm selects a substring that leads to the largest contraction suffix tree used to find the substring with the maximum possible number of non-overlapping occurrences. The relatively poor compression achieved by Off-line (average compression ratio 1.938 bpb - standard benchmark data) is attributed to its consideration of exact repeats only. Indeed, we can say that Off-line is not a DNA-specific compressor but rather a general purpose compressor which works reasonably well also for DNA sequences[34]. Off-line is a time consuming algorithm because of its need for building the suffix tree.

Matsumoto et al. produced a new algorithm called CTW + LZ[27] that is based on the context tree weighting method, which uses a weighting of multiple models to determine the next symbol probabilities. The algorithm detects approximate repeats using dynamic programming then encodes long exact and approximate repeats using an LZ77-type encoding. Short repeats and non-repeats are encoded using a CTW. The algorithm performs better than Biocompress-2 and GenCompress (average compression ratio 1.738 bpb - standard benchmark data). Although they obtained good compression ratios, its execution time is too high to be used for long sequences. Chen et al.[32] show that a sequence of just 229 K bases required many hours to compress.

Tabus et al. in[42] proposed a sophisticated DNA sequence compression method based on normalized maximum likelihood (NML) discrete regression for approximate block matching. NML algorithm separates the input into fixed-size blocks. To encode a block, the algorithm finds a "regressor", which is a substring that occurred earlier in the input that has the minimum Hamming distance from the current block, and encodes the block as a reference to the earlier occurrence and uses a bit mask to represent the differences between the current block and the regressor. The bit mask is encoded using an order-0 NML model between the current block and the regressor.

This work was later improved in[43] to produce GeNML 2005, which encodes fixed-size blocks (equal to 24, 32, 40, or 48)[3] by referencing a previously encoded sub-sequence (size equal to 218)[3] with a minimum Hamming distance. Only replacement operations are allowed for editing the reference sub-sequence, which therefore always has the same size as the block, although it 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-1 and a transparent model in which the block passes uncompressed. The algorithm performs better than Biocompress-2, GenCompress, CTW + LZ, and DNAPack (average compression ratio 1.69 bpb - standard benchmark data, except HUMHBB" because the authors were unable to find the correct version"). Also, GeNML improves genome size compression and speed. For example, for the Gbase H.

sapiens genome, a compression ratio of 1.535 bpb with a compression time of 3.5 hours.

Further improvements to GeNML[2] include using an order-1 NML model to encode the bitmask, where the current bit is encoded based on the previous bit, and finding the best regressor block by means of first-order dependencies (these dependencies were not considered in the previous approach). With these improvements this algorithm produced better compression results for the human genome.

Mishra et al. presented a DNA Sequence Compressor (DNASC) in 2010[3]. This algorithm compresses the DNA sequence horizontally first by using extended Lempel-Ziv style representation for the 5 basic symbols A, C, G, T, and N. Then the sequence is vertically compressed by taking a block size equals to 6 and a window size equals to 128. To compress every 2 digits of a block they used 7 bits. Therefore, each block is compressed by using only 21 bits, because each block has 6 digits and each 2 digits are represented by 7 bits. The DNASC algorithm gives better compression results in comparison to Biocompress2, Gencompress, CTW+LZ, DNA compress, and GeNML 2005. By comparing the results of DNASC algorithm with GeNML 2005, the former algorithm has an average compression ratio equals 1.54 bpb - for standard benchmark data, except "HUMHBB sequence". The DNASC algorithm also compressed the human genome.

3.4. Transformational Based Methods

Any sequence is subject to transformations before the actual compression takes place to achieve a better compression ratio, such as Burrows–Wheeler Transform (BWT). Based on the transformation from a DNA sequence to a codon sequence there are three methods that specialize in the compression of biological sequences.

Bao et al. introduced a new algorithm based on fixed length LUT and LZ77[44]. First, they created a fixed size look up table which contains a combination of three characters (without 'N', 'space' or unknown nucleotide) forming 64 combinations (codons), resulting in a fixed size of the table. Also they encountered a series of successive N's, between two actual DNA strings in the destination file. Later these combinations were replaced by the symbols which are assigned in the lookup table, hence resulting in encoding using LZ77 algorithm. Applying LZ77 to the pre-coded file only improves very slightly the compression ratio. Compared with Chen's algorithm[32], the compression ratio of this algorithm is 0.2 bpb, higher in general. But the cost of the tiny 0.2 bpb, improvement was very high. Nonetheless, this algorithm runs almost 103 times faster than DNAcompress[32].

The Differential Direct Coding algorithm (2D) by Vey in[45] proposes that compression strategies must accommodate large data sets and consist of multiple sequences and auxiliary data. The set of expected symbols for the 2D model are (A, T, G, C, and U), which removes the burden of explicit declaration of sequence type like DNA or RNA. The 2D model accommodates a total of 125 different triplets according to any of the nucleotide bases at any of the three triplet positions, such that the set of codons {AAA, AAC, . . . , UUT, UUU}. These instances are accommodated in order to provide simplified arithmetic translation. Also, 128 different ASCII symbols are supported as extra symbols and a single unknown flag is included to denote a symbol that belongs to neither set. By using several bacterial genomes, the results show that gzip provided the best compression ratios while 2D had the fastest execution times. If 2D were applied and followed immediately with gzip, this would provide the best compression ratios and at the same time execution times that are still faster than gzip alone. 2D is suitable for any type of sequence data, including very large data sets, such as meta-genomes.

In 2011 Bharti and Singh[46] proposed a two phase compression algorithm based on a Look up Table (LUT) using a complementary palindrome of fixed size. During the first phase of the algorithm the algorithm searches for all palindromes in a specific length (3 Base Character). This is carried out by checking all the possible places in the sequence. Then the algorithm finds a palindrome that correlates with its demands and prints it to the output. In second phase they apply the LUT base variable length compression algorithm. This proposed algorithm has a high compression ratio compared to other existing Biological Sequence Compression algorithms, such as GenCompress [31], DNACompress[32], 2D[45], and Fixed LUT[44]. Also it uses less memory compared to the other existing algorithms and is easy to implement.

Later in 2012 Roy et al. proposed a finite LUT[47] in two algorithms with a four-step coding rule. The first step is based on the LUT, which comprises a combination of three characters among 'A', 'T', 'C, 'G' or 'N' (algorithm-1) and a combination of four characters ('A', 'T', 'C' and 'G') (algorithm-2). The second step is based on ASCII characters (in the form of algorithm-1 which is based on 125 chosen ASCII character 8 bits each and algorithm-2 which includes 256 ASCII characters 8 bit each. The third step is based on tandem repeats only for algorithm-1. The fourth step is based on a segment which consists of 1 or 2 characters. The authors used different data sets, three homo sapiens mRNAs, five funguses, Rhizopus oryzae, glucoamylase A precursor gene, partial cds, and four Zea mays mRNA were used to test the algorithms. Results suggest that these proposed compression algorithms perform better than most general compression. This technique also requires less memory and less coding effort compare to the other algorithms. By taking a block size of three/four characters and using tandem repeats in the sequence line, it is easier to make sequence alignment and sequence analysis between compressed sequences.

3.5. Grammar Based Method

These algorithms infer context-free grammars to represent the input data. The grammar is then transformed into a symbol stream and finally encoded in binary.

DNASequitur[48] is a grammar-based compression algorithm explored by Cherniavsky and Ladner for DNA sequences, which infers a context-free grammar to represent the input sequence. The novelty of DNASequitur is its ability to recognize reverse complements when creating rules and during substitutions beside the exact repeats. Results show that other methods achieve better compression ratios. Grammar-based compression is ideal for detecting repeats that occur across multiple sequences in a collection. Another advantage is the ability to support random access and search in the compressed collection. Therefore, grammar-based DNA compression should not be ruled out.

3.6. Two bits Based Methods

These algorithms make first bit-preprocessing by assigning 4 unique two bits (A = 00, G = 01, C = 10, and T = 11) to different four DNA bases before the encoding process. A DNA sequence is represented in smaller segments before the encoding process to compress both repetitive and non-repetitive DNA sequences. The input sequence is divided into fragments, where each fragment is four 8 bit-characters long. The following algorithms are similar in the bit-preprocessing stage but different in the coding stage.

Rajeswari and Apparao proposed the GENBIT Compress tool[49] for DNA sequences based on a novel concept of assigning binary bits. After the bit-preprocessing stage in this coding scheme, 256 combinations can be represented. Hence every DNA segment containing four bases is replaced by an 8 bit binary number "XXXXXXXX". If the consecutive fragments are the same, then a specific bit "1" is introduced as a 9th bit. If the consecutive fragments are different, then a specific bit "0" is introduced as a 9th bit to the 8 bit unique number. It takes an n-fragment long input of a DNA sequence, and divides into n/4 fragments. The left out individual bases (fragment length < 4) are assigned 4 unique "2" bits. The authors assume DNA sequences with a length of 64 bases to test this algorithm, not real biological sequences. They got a compression ratio for the best case (maximum repetitive fragments) of 1.125 bpb, while it was 1.727 bpb for the average case (random input which is not given by either the worst-case or the best-case efficiency), and 2.238 bpb for the worst case (there are no repetitive fragments), even for larger genomes (nearly 2,000,000 characters). Also, the GENBIT Compress tool program significantly improves the running time of all previous DNA compressors but expand the DNA sequence (with average = 2.23 bpb - standard benchmark data, except MPOMTCG, HEHCMVCG, HUMGHCSA and HUMHDABCD).

Rajeswari et al. introduced HUFFBIT compress[60] for DNA sequences. In this algorithm a bit-preprocessing stage to the sequence takes place. Then the extended binary tree principle is applied to derive a special class of variable length codes that satisfies the prefix property (Huffman Codes). The authors assume DNA sequences with length 1000 bases to test this algorithm, not real biological sequence. They got a better compression ratio than GENBIT. For the best case they got 1.006 bpb, for the average case they got 1.611 bpb, and for the worst case they got 2.109 bpb.

Rajeswari and Apparao presented the DNABIT Compress tool (DBC)[51] that assigns binary bits "in the bit-preprocessing stage" to exact and reverse repeats fragments of DNA sequences. This a unique concept introduced in this algorithm for the first time in DNA compression. DNABIT achieves the best compression ratio for DNA sequences for larger genomes. It significantly improves the running time and achieves better compression (with an average of 1.58 bpb - standard benchmark data, except MIPACGA and HUMGHCSA). Results show that DBC is the best among the remaining compression algorithms (Biocompress[29], CTW+LZ[27], GenCompress [31], DNA Compress[32], and DNAPack[4]).

GenCodex[52] introduced by Satyanvesh et al. yields a better compression ratio at a high throughput by using graphical processing units (GPUs) and multi-cores in two phases. In the first phase, bit-preprocessing occurs. In the next phase, the fragments are represented using either one or two bytes. If a fragment does not appear consecutively, then a single byte is allocated using its 8-bit unique representation. If a fragment is repeating two or more times, then the simple 8-bit representation is put in the first byte and the number of repetitions is represented in the second byte. For every eight bytes of the compressed data, an extra byte is used as a code byte in which we set the corresponding bit to 1 if there is a repetition. The compression ratio of GenCodex is better than GENBIT[49] and DNABIT[51] algorithms for the best and average cases. Results show that this method achieves a good compression ratio along with a better throughput compared to other existing methods such as GENBIT[49], GenCompress[31], and DNA Compress[32].

Prasad and Kumar in the DNACRAMP tool[53] took a sequence of DNA for bit-preprocessing. Then they used basic procedural language to perform the encoding and decoding process with the help of a two-stage index bounded linear array data structure. This technique is applicable on repetitive and non-repetitive sequences of DNA. DNACRAMP obtains better compression ratios (with an average of 1.143 bpb - standard benchmark data) than DNABIT[51], DNAPack[4], CTW + LZ[27], and DNASC [3].

Prasad then introduced PGBC "Partitioned group binary compression"[54] that improves the worst-case scenario of the previous algorithms. After bit-preprocessing the encoding process starts. Every six fragments can be grouped as a partition, which contains two sub- partitions. Afterwards, the algorithm substitutes the fragment by its equivalent bits, before making sub partitions, and later the fragment is grouped into a single main partition set. These algorithms are far better than existing ones (e.g., HUFFBIT[60], GENBIT [49]) and suitable for non-repetitive DNA sequences of genomes (they encode every base by 1.33 bits in the worst-case scenarios). In addition to that, existing techniques use dynamic programming to compress the sequence which is complex in implementation and time consuming, but these algorithms are simple and fast.

4. Algorithms for DNA Compression in Vertical Mode

This mode uses the information between two sequences typically by making one of them a reference sequences. We can say that vertical mode compression nowadays is growing faster because of the development of new DNA sequencing technologies, such as next-generation sequencing (NGS). These new high-throughput sequencing technologies have led to a large number of whole genome DNA sequencing projects. Vertical mode compression is more suitable for sequences with large size.

There has been considerable work on compression of sequencing data with some research in vertical mode DNA compression in many data formats e.g. FASTQ[55] and SAM/BAM[56] formats.

Christley et al. present a DNAzip package[57], which is a series of techniques that in combination reduces a single genome to a size small enough to be sent as an email attachment.

Tembe et al. in[58] presented a Huffman coding-based sequencing which reads specific representation schemes that compress data without altering the relative order.

Daily et al. in[59] developed data structures and compression algorithms for high-throughput sequencing data. A processing stage maps short sequences to a reference genome or a large table of sequences. Then the integers representing the short sequence's absolute or relative addresses, their length, and the substitutions they may contain, are compressed and stored using various entropy coding algorithms.

Afify et al. in[60] exploited a differential compression model based on the alignment of two similarity sequences or more, which compress one sequence by comparing it with another sequence and using renewal entropy estimation.

Grabowski and Deorowicz in[5] present an LZ77-style compression scheme for relative compression of multiple genomes of the same species to improve runtime and compression performance.

Kuruppu et al. proposed the RLZ algorithm in[61] and the improved RLZ in[62], which are algorithms that provide effective compression in a single pass over the collection, and the final compressed representation allows rapid random access to arbitrary substrings using a simple greedy technique, akin to LZ77 parsing.

The main difficulty of relative compression is in selecting an appropriate reference sequence. Kuruppu et al. in[8], explore using the dictionary of repeats generated by Comrad [63], Re-pair[64] and Dna-x[34] algorithms as applied to reference sequences for relative compression.

Deorowicz and Grabowski present DSRC[65], which is a specialized compression algorithm for genomic data in FASTQ format and which dominates its competitor, G-SQZ.

In CRAM[66], Fritz et al. present a new reference-based compression method that efficiently compresses DNA sequences for storage. This approach works for re-sequencing experiments that target well-studied genomes. They align new sequences to a reference genome and then encode the differences between the new sequence and the reference genome for storage.

Sakib et al. present SAMZIP[67], a specialized encoding scheme, for sequence alignment data in SAM format, which improves the compression ratio of existing compression tools available.

Wang and Zhang present a novel compression tool for storing and analyzing genome re-sequencing data, named GRS[68]. GRS is able to process the genome sequence data without the use of the reference single-nucleotide polymorphism (SNPs) and other sequence variation information and automatically rebuilds the individual genome sequence data using the reference genome sequence.

As biologists move their analyses to high-performance systems with greater I/O bandwidth, low-throughput compression becomes a limiting factor. Mark Howison addresses this gap by a new storage model called SeqDB[69], which offers high-throughput compression of sequence data with minimal sacrifice in compression ratio.

Pinho et al. describe GReEn[70] (Genome Re-sequencing Encoding), a tool for compressing genome re-sequencing data using a reference genome sequence. It overcomes some drawbacks of the proposed tool GRS,

Jones et al. present Quip[71], a lossless compression algorithm for next-generation sequencing data in the FASTQ and SAM/BAM formats using reference-based compression.

Popitsch and Haeseler present NGC[72], a tool for the compression of mapped short read data stored in the wide-spread SAM format. NGC introduces two novel ideas: first, a way to reduce the number of required code words by exploiting common features of reads mapped to the same genomic positions; second, a highly configurable way for the quantization of per-base quality values, which takes their influence on downstream analyses into account.

5. Conclusions

Research in bioinformatics largely depends on storage and manipulation of huge amounts of data. We believe that efficient DNA "the code of life" compression remains a challenging problem and a rather difficult task.

In substitution algorithms searching for some kinds of repeats such as exact, reverse repeats, complemented repeats, and complemented palindromes, then encode using an appreciable algorithm, take more compression time. The Lempel-Ziv compression algorithm has become a reasonable default choice for compression of DNA.

Most of statistical algorithms are effective for DNA sequence compression and computationally intensive in practice. Compression algorithms with combined substitution and statistical modules provide better results over substitution algorithms only.

Combining a lookup table pre-coding transformation making use of three or four bases with other compression algorithms improve its performance. Also it is easy to make

sequence alignment and sequence analysis between compressed sequences.

Grammar-based compression is ideal for detecting repeats and has the ability to support random access and search in the compressed collection but with poor compression ratio. In the future, we may explore the use of edit grammars for DNA sequences such as insert, delete, or replace a character with another to improve the compression ratio.

Bit-based compression is compressing both repetitive and non-repetitive DNA sequences. It is implemented without dynamic programming, so it is simple and fast.

Finally, vertical mode compression provides an efficient storage model for the data produced by new DNA sequencing technologies, such as next-generation sequencing. Combining vertical mode compression and statistical compression may improve the compression ratio by determining other factors to choose a reference sequence.

ACKNOWLEDGEMENTS

I would like to send special thanks to all whom were supported me throughout this work.

REFERENCES

[1] Pinho, A. J., Neves, A. J. R., Afreixo, V., et al., 2006, A Three-State Model for DNA Protein-Coding Regions, IEEE Trans. Biomed Eng., 53(11), 2148 –2155.

[2] 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.

[3] Mishra, K. N., Aaggarwal, A., Abdelhadi, E., et al., 2010, An Efficient Horizontal and Vertical Method for Online DNA Sequence Compression, International Journal of Computer Applications, 3(1), 39-46.

[4] A. Postolico, et al., Eds., DNA Compression Challenge Revisited: A Dynamic Programming Approach, Lecture Notes in Computer Science, Island, Korea: Springer, 2005, vol. 3537, 190–200.

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

[6] Horner, D. S., Pavesi, G., Castrignanò, T., et al. , 2010, Bioinformatics approaches for genomics and post genomics applications of next-generation sequencing, Briefings in Bioinformatics, 11(2), 181–197.

[7] Pushkarev, D., Neff, N. F., and Quake, S. R., 2009, Single-molecule sequencing of an individual human genome, Nature Biotechnology, vol. 27, 847–852.

[8] R. Grossi et al., Eds., Reference Sequence Construction for Relative Compression of Genomes, Lecture Notes in Computer Science, Pisa, Italy: Springer, 2011, vol. 7024, 420-425.

[9] Benson, D. A., Karsch-Mizrachi, I., Lipman, D. J., et al., 2005, GenBank, Nucleic Acids Research, vol. 33, 34-38.

[10] Brooksbank, C., Cameron, G., and Thornton, J., 2010, The European Bioinformatics Institute's data resources, Nucleic Acids Research, vol. 38, 17-25.

[11] Sugawara, H., Ogasawara, O., Okubo, K., et al., 2008, DDBJ with new system and face, Nucleic Acids Research, vol. 36, 22-24.

[12] Shumway, M., Cochrane, G., and Sugawara, H., 2010, Archiving next generation sequencing data, Nucleic Acids Research, vol. 38, 870-871.

[13] Barrett, T., Troup, D. B., Wilhite, S. E., et al. , 2009, NCBI GEO: archive for high-throughput functional genomic data, Nucleic Acids Research, vol. 37, 885-890.

[14] Kapushesky, M., Emam, I., Holloway, E., et al. , 2010, Gene expression atlas at the European bioinformatics institute, Nucleic Acids Research, 38(1), 690-698.

[15] Ahmed A., Hisham G., Moustafa G., et al., 2010, EGEPT: Monitoring Middle East Genomic Data, Proc., 5th Cairo International Biomedical Engineering Conf., Egypt, 133-137.

[16] Karsch-Mizrachi, I., Nakamura, Y., and Cochrane, G., 2012, The International Nucleotide Sequence Database Collaboration, Nucleic Acids Research, 40(1), 33–37.

[17] International nucleotide sequence database collaboration, (2013),[Online]. Available: http://www.insdc.org.

[18] Celniker, S. E., Dillon, L. A. L., Gerstein, M. B., et al. , 2009, Unlocking the secrets of the genome, Nature, 459(7249), 927–930.

[19] Guttmacher, A. E., and Collins, F. S., 2005, Realizing the promise of genomics in biomedical research, JAMA: Journal of the American Medical Association, 294(11), 1399–1402.

[20] Joosen, R. V., Ligterink, W., Hilhorst, H. W., et al., 2009, Advances in genetical genomics of plants, Current Genomics, 10(8), 540–549.

[21] Womack, J. E., 2005, Advances in livestock genomics: opening the barn door, Genome Research, 15(12), 1699–1705.

[22] Genbank size, (2013),[Online]. Available: http://ftp.ncbi.nih.gov/genbank/gbrel.txt

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

[24] Giancarlo, R., Scaturro, D., and Utro, F., 2009, Textual data compression in computational biology: a synopsis, Bioinformatics, 25(13), 1575–1586.

[25] Nalbantoğlu, Ö. U., Russell, D.J., and Sayood, K., 2010, Data Compression Concepts and Algorithms and their Applications to Bioinformatics, Entropy, 12(1), 34-52.

[26] 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.

[27] Matsumoto, T., Sadakane, K., and Imai, H., 2000, Biological Sequence Compression Algorithms, Genome Informatics, vol. 11, 43–52.

[28] Sato, H., Yoshioka, T., Konagaya, A., et al., 2001, DNA Data Compression in the Post Genome Era, Genome Informatics, vol. 12, 512–514.

[29] Grumbach, S., and Tahi, F., 1994, A new challenge for compression algorithms: genetic sequences, Information Processing & Management, 30(6), 875–886.

[30] E. Rivals, M. Dauchet, J-P. Delahaye, et al., "Fast Discerning Repeats in DNA Sequences with a Compression Algorithm," The 8th Workshop on Genome and Informatics, (GIW97), 1997, vol. 8, 215-26.

[31] X. Chen, S. Kwong and M. Li, "A Compression Algorithm for DNA Sequences and It's Applications in Genome Comparison," The 10th Workshop on Genome and Informatics, (GIW99), 1999, vol. 10, 51-61.

[32] Chen, X., Li, M., Ma, B., et al., 2002, DNACompress: fast and effective DNA sequence Compression, Bioinformatics, 18(12), 1696–1698.

[33] Ma, B., Tromp, J. and Li, M., 2002, Pattern Hunter: faster and more sensitive homology search, Bioinformatics, 18(3), 440–445.

[34] Manzini, G., and Rastero, M., 2004, A Simple and Fast DNA Compressor, Software: Practice and Experience, 34(14), 1397–1411.

[35] A. J. Pinho, A. J. R. Neves, D. A. Martins, et al., Finite-Context Models for DNA Coding, Signal Processing Lab, DETI/IEETA, S. Miron , Ed., University of Aveiro, Portugal, Chapter 6, 117-130, 2010.

[36] A. J. T. Lee, C. Chang and C. Chen, "DNAC: An Efficient Compression Algorithm for DNA Sequences," National Taiwan University, Taipei, Taiwan 10617, R.O.C., 2004.

[37] D. Loewenstern, and P. N. Yianilos, "Significantly lower entropy estimates for natural DNA sequences," in Proc. of the Data Compression Conf., (DCC '97), 1997, 151–160.

[38] Allison, L., Edgoose, T., and Dix, T. I., 1998, Compression of strings with approximate repeats, Proc. ISMB, 8–16.

[39] M. D. Cao, T. I. Dix, L. Allison, et al., "A Simple Statistical Algorithm for Biological Sequence Compression," in Proc. of the Data Compression Conf., (DCC '07), 2007, 43–52.

[40] A. Apostolico, and S. Lonardi, Compression of biological sequences by Greedy Off-Line Textual Substitution, in Proc. of the Data Compression Conf., (DCC '00), 2000, P. 143.

[41] Apostolico, A., and Lonardi, S., 2000, Off-Line Compression by Greedy Textual Substitution, Proc. IEEE, 88(11), 1733–1744.

[42] 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. (DCC2003), 2003, 253–262.

[43] Korodi, G., and Tabus, I., 2005, An Efficient Normalized Maximum Likelihood Algorithm for DNA Sequence Compression, ACM Trans. on Information Systems, 23(1), 3–34.

[44] Bao, S., Chen, S., Jing, Z., et al., 2005, A DNA Sequence Compression Algorithm Based on LUT and LZ77, Signal Processing and Information Technology, Proc., 50th IEEE International Symposium, 23–28.

[45] Vey, G., 2009, Differential direct coding: a compression algorithm for nucleotide sequence data, Database, Oxford University Press, vol. 2009, ID bap013.

[46] Bharti, R. K., and Singh, R. K., 2011, A Biological Sequence Compression based on Look up Table (LUT) using Complementary Palindrome of Fixed Size, International Journal of Computer Applications, 35(11), 55-58.

[47] Roy, S., Khatua, S., Roy, S., et al., 2012, An Efficient Biological Sequence Compression Technique Using LUT and Repeat in the Sequence, IOSR Journal of Computer Engineering (IOSRJCE), 6(1), 42-50.

[48] N. Cherniavsky and R. Ladner, "Grammar-based Compression of DNA Sequences," UW CSE Technical Report (TR2007-05-02), presented at the DIMACS Working Group, 2004.

[49] Rajeswari, P. R., and Apparao, A., 2010, Genbit Compress Tool (GBC): A 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.

[50] Rajeswari, P. R., Apparao, A., and Kumar, R. K., 2010, HUFFBIT COMPRESS – Algorithm to compress DNA sequences using extended binary tree, Journal of Theoretical and Applied Information Technology, 13(2), 101-106.

[51] Rajeswari, P. R., and Apparao, A., 2011, DNABIT Compress – Genome compression algorithm, Bioinformation, 5(8), 350-360.

[52] Satyanvesh, D., Balleda, K., Padyana, A., et al., 2012, GenCodex - A Novel Algorithm for Compressing DNA sequences on Multi-cores and GPUs, Proc. IEEE, 19th International Conf. on High Performance Computing (HiPC), Pune, India, No 37.

[53] Prasad, V. H., and Kumar, P. V., 2012, A New Revised DNA Cramp Tool Based Approach of Chopping DNA Repetitive and Non-Repetitive Genome Sequences, International Journal of Computer Science Issues (IJCSI), 9(6), 448-454.

[54] Prasad, V. H., 2013, A new revisited compression technique through innovation partition group binary compression: a novel approach, International Journal of Computer Engineering & Technology (IJCET), 4(2), 94-101.

[55] Cock, P. J. A., Fields, C. J., Goto, N., et al., 2010, The sanger FASTQ format for sequences with quality scores, and the Solexa/fillumina FASTQ variants, Nucleic Acids Research, 38(6), 1767–1771.

[56] Li, H., Handsaker, B., Wysoker, A., Fennell, T., et al., 2009, The Sequence Alignment/Map format and SAMtools, Bioinformatics, 25 (16), 2078-2079.

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

[58] Tembe, W., Lowey, J., and Suh, E., 2010, G-SQZ: compact encoding of genomic sequence and quality data, Bioinformatics, 26(17), 2192-2194.

[59] Daily, K., Rigor, P., Christley, S., et al., 2010, Data structures and compression algorithms for high-throughput sequencing technologies, BMC Bioinformatics, vol. 11.

[60] Afify, H., Islam, M., Abdel-Wahed, M., et al., 2010, Genomic Sequences Differential Compression Model, Proc., 27th National Radio Science Conf., Egypt.

[61] E. Chavez and S. Lonardi, Eds., Relative Lempel-Ziv Compression of Genomes for Large-Scale Storage and Retrieval: Springer 2010, vol. 6393, 201–206.

[62] S. Kuruppu, S. J. Puglisi, and J. Zobel, "Optimized Relative Lempel-Ziv Compression of Genomes," in Proc. Australasian Computer Science Conf. (ACSC'11), 2011, vol. 113, 91-98.

[63] Kuruppu, S., Smith, B. B., Conway, T., et al., 2012, Iterative dictionary construction for compression of large DNA datasets, Computational Biology and Bioinformatics, IEEE/ACM, 9(1), 137–149.

[64] N. J. Larsson, and A. Moffat, "Offline dictionary-based compression," in Proc. Data Compression Conf. (DCC'99), 1999, 296-305.

[65] Deorowicz, S., and Grabowski, S., 2011, Compression of genomic sequences in FASTQ format, Bioinformatics, 27(6), 860-862.

[66] Fritz, M. H., Leinonen, R., Cochrane, G., et al., 2011, Efficient storage of high throughput DNA sequencing data using reference-based compression, Genome Research, vol. 21, 734-740.

[67] Sakib, M. N., Tang, J., Zheng, W. J., et al., 2011, Improving Transmission Efficiency of Large Sequence Alignment/Map (SAM) Files, PLOS ONE, 6(12), e28251.

[68] Wang, C., and Zhang, D., 2011, A novel compression tool for efficient storage of genome resequencing data, Nucleic Acids Research, 39(7), e45.

[69] Howison, M., 2013, High-Throughput Compression of FASTQ Data with SeqDB, Computational Biology and Bioinformatics, IEEE/ACM, 10(1), 213 – 218.

[70] Pinho, A. J., Pratas, D., and Garcia, S. P., 2012, GReEn: a tool for efficient compression of genome resequencing data, Nucleic Acids Research, 40(4), e27.

[71] Jones, D. C., Ruzzo, W. L., Peng, X., et al., 2012, Compression of next-generation sequencing reads aided by highly efficient de novo assembly, Nucleic Acids Research, 40(22), e17.

[72] Popitsch, N. and Haeseler, A., 2013, NGC: lossless and lossy compression of aligned high-throughput sequencing data, Nucleic Acids Research, 41(1), e27.