IJCSE International Journal of Computer Sciences and Engineering Open Access
Survey Paper Vol.-6, Issue-8, Aug 2018 E-ISSN: 2347-2693

A Survey of Genome Compression Methodology

Rituparna Mitra[1], Subhankar Roy[2*]

[1]Dep. of Computer Science, Kirtipur Nabin Chandra High School (H.S.), Kirtipur, Kolkata-700128, W.B., India
[2]Dep. of Computer Science and Engineering, Academy of Technology, G. T. Road, Aedconagar, Hooghly-712121, W.B., India

[*]*Corresponding Author: subhankar.roy07@gmail.com, Tel.: +919681011727*

Available online at: www.ijcseonline.org

Accepted: 18/Aug/2018, Published: 31/Aug/2018

Abstract- Storing the information about human nucleotide is become essential now a day for various medical research purposes. A human genome consists of almost 3.2 billion nucleotides. It is unmanageable to store, access and retrieve the desired information from the massive bulk of unprocessed data. So the possible solution is genome compression. By compression we mean that we are restricting on the data storage. Existing data compression methodology is not suitable to deal with this massive data. In this paper we provide a survey analysis on various types of genome compression and read compression algorithm which are specially designed to handle this voluminous raw DNA information. To extract the unique non repeated information from the whole sequence is actually a tough challenge .These compression algorithms not only save time but also provide high compression rate. We have discussed all the types of compression algorithm with their distinctive approach. Each of them having some benefit over other. We also briefly discuss on various file formats used while compression.

Keywords— Genome compression, Read compression, Data formats

I. INTRODUCTION

The cost of ordering the nucleotides within a genome has dropped in such a way that it is more tending to produce larger and longer reads by various sequencing platforms which results in a massive amount of unmanageable data. It is not easy to store, access, retrieve or transfer this amount of data as it is a time consuming and costly process. Storing these large amount of data need a sufficient storage space which is beyond the possibility after a certain level. Accessing this data took lot of time to retrieve the required information. Transferring this data needs a large bandwidth and also most of the time it cause network congestion. One easily achievable solution is to remove the sequences form storage after manipulating it or to store the sequence in a compressed form for future reusability. The second method is more reliable and scientific. This compression of genome sequencing can be acquired by the use of any genome compression algorithm. These compression algorithms are complicated as they give importance to find out the raw non repeated sub-sequences from the huge amount of data.

In this paper we are first going to discuss about some DNA compression algorithm in section II. Sequencing the genome is very crucial to properly analyse the genome information. The production of short reads were done while sequencing. In the section III of our paper we have mentioned about various read compression algorithm. Following this we talk about various research works done on genome compression recently in section IV. After that in the next segment i.e. in section V we converse about various file formats specifically required to support genomic compressed data. Finally we conclude our paper in section VI by providing our opinion on various compression techniques which we have discussed in this paper. In this paper we have done a survey analysis on various genome compressions and read compression techniques so that anyone could get an overview of different methodology which will help in their future research works and also those who are unaware of this domain of bioinformatics could get the basic concept about genome and read compression.

II. GENOME COMPRESSION

There are various type of genome compression algorithm are there each of them having their own unique logical structure. This can be categorized into following way-

2.1. Bit operation oriented compression

Each DNA sequence consists of chain of genome bases which are also known as nucleotides, they are adenine (A), guanine (G), cytosine (C) and thymine (T). In this technique each bases are represented using binary code(like A->00,C-

>01,G->10,T->11) and two or more bases are put into one set. In this way the whole compression technique took place. GenBit Compress (GBC) [1] algorithm use this methodology for compression. The algorithm based on 2 bit compression and run length encoding technique. Using similar approach Rajendra Kumar Bharti et al. presents a sequence compression algorithm in 2011 where they put three bases in a set to encode the genome and then compress it using LZ77 [2]. In 2010 Ateet Mehta and Bankim Patel incorporates hash functions with bit operation to compress genome sequences [3]. Piyuan Lin, Shaopeng Liu, Lixia Zhang, et al in 2009 make revolutionary change in approach by applying pattern matching technique for compression [4]. In 2011 Pothuraju Rajeswari and Allam Apparao [5] give attention not to compress the whole genome sequence but to figure out the non-repeated portion in a genome and hence produces different bit patterns for compression of repeated and non-repeated part in the genome.

2.2. Dictionary oriented compression

In this method the input sequence is scanned from left to right to check for the repeated and non-repeated substrings within the given sequence. After scanning the non-repeated parts is stored for future reference but the redundant repeated parts are not stored in place of that a reference is stored which then links with a dictionary where all the repeated substrings are stored. This method of compression does not depend on the given input sequence. Hence this approach focuses on the reduction on the storage consumption made by the genome sequences. Biocompress [6], Biocompress-2[7] and Cfact [8] these algorithms follow this approach, but compression rate for these are very poor. To improve the performance approximate repeated sequences are being given preference and in 2000, Chen et al. developed GenCompress [9] with better result. This algorithm is further improved by incorporating palindrome scheme in DNACompress [10] by Chen et al. in 2002. This algorithm achieves higher compression rates for large sequence as well. CWT+LZ developed by Matsumoto et al.[11] follow the same procedure as GenCompress but it is capable of encoding only smaller repeated sub sequences, the longer sequences are handled by substitution method and to store the non-repeated parts it uses a context tree weighting technique. COMRAD [12] algorithm creates the dictionary simultaneously while checking for the similar subsequence from the input. This algorithm operates in multiple passes. Each time while scanning the input sequence it discovered a longer sub-sequence, it simply updates the dictionary with it. The compressed sequence string with the dictionary is further encoded. The above mentioned algorithms are unable to do the storage utilization due to the approach of finding the approximate match from the given sequence, Manzini and Rastero thoroughly examines the various sequences and detects three types of repeat can be occurred in any sequence, i.e. non-repeated, approximate repeat and reverse complement repeat. Depending on this category they invented DNA-X [13] algorithm which performs in better storage utilization and also gives higher compression rate. The modernization of Cfact algorithm was done by Lee et al.[14] with four phases. In the first phase exact matched sub-strings are detected to form a suffix tree, these were comprehended to approximate repeat using dynamic programing method in the next phase, it further recognize the non-overlapped portion from it and finally Fibonacci encoding is used for the repeated portions. Dimitris Antoniou et al. [15] incorporate the functioning of splay tree in their compression algorithm. It has been proved that conventional way of compression of the genome by considering the sequence altogether is not always gives the optimal compression rate whereas Kalyan Kumar Kaipa et al. [16] designed a new approach of genome compression by partitioning the whole sequence into uneven chunks of sequences, then encode them independently. To keep track of the repeated parts in various chunks a hash table is maintained. It is necessary to recognize the replicated parts within a sequence to achieve optimal compression. Hamming distance between various sequences can be measured to identify the repeated parts within them; this method is used in DNAPack [17] algorithm.

2.3. Substitution and Statistics Oriented Algorithms

Dictionary based algorithms tries to find out the duplicate segments within a genome but it unable to deal with approximate repeated segments whereas the statistical algorithms only concentrates on detecting the sub-sequences which appears frequently within a sequence. So it is obvious that statistical methods give better performance than dictionary based approach. When we combine these two approaches its performance is extremely noticeable. Following this approach in 2000 Matsumoto et al. introduces CTW+LZ [18]. This algorithm first scan the sequence to detect the palindrome and approximate repeated segments through dynamic programming and hash based functions, after that it uses LZ77 encoding technique to compress the longer replicated partitions and the shorter one is encoded using CTW (context tree weighting) structure. The algorithms based on NML (normalized maximum likelihood)[19] and GeNML [20] are also belongs to this category. The NML firstly divides the whole sequence into equal sized blocks then it tries to locate the approximate repeated segments within each block. The approximate repeated segments of the present block are further encoded with the reference to a segment of previously occurred block with least hamming distance and the exact replicated portions are encoded using simple Markov model. Thus the inability to deal with approximate repeated segments with in a sequence is reduced after invention of NML. GeNML is the improved version of the previous algorithm. This algorithm divides the sequence into variable sized blocks. Here the reference sequence can be modified (insertion, deletion or

substitution) according to the present sequence. In 2010 Mishra et al. presented DNA Sequence Compressor (DNASC 21]. In this algorithm both horizontal and vertical compression took place. For horizontal compression they used extended Lempel-ziv encoding technique. In the next phase the previous encoded sequence is further compressed vertically taking a block size of 6 and window size of 128.

2.4. Statistics Oriented Algorithm

The statistical based algorithms focus on the occurrence frequency of the sub sequences within a genome. The probability distribution of each symbol must be recognized to achieve high compression. The most common statistical encoding technique is Huffman encoding [22]. It produces a binary tree depending upon the occurrence of the substring. The substring with higher frequency is represented using shorter code and substring with lower frequency is represented with longer codes. This approach is very useful in DNA compression. The compressed sequence is stored with an additional Huffman code table for computing the compression ratio when required. This same reference table can be shared among many sequences while compression. Hence it also reduces the storage cost. CDNA [23] is the first statistical based algorithm introduced by Loewenstern et al. in the year 1997. This algorithm scans the whole sequence to keep track of the probability distribution of each symbol. This algorithm is trying to detect the approximate matched subsequence with reference to a previously occurred sequence having smaller Hamming distance. In the next year Allison et al. proposed ARM [24] algorithm which is also based on statistical analysis of occurrence of sub-sequences within a genome. To calculate the probability of a subsequence the algorithm first observe the production of each sub-sequences and then adding the probabilities of all sequences. XM [25] is another statistical algorithm which compresses the sequence depending upon the frequency of appearance of each symbol. The appearance frequency means how many time that symbol is appeared with in the sequence. It is also known as probability distribution of the symbol. This probability distribution of symbols is further passed to an arithmetic encoder for final encoding. To determine the occurrence of each symbol the algorithm took help of a group of systems i.e. (1) order-2 Markov models; (2) order-1 context Markov models and (3) a copy expert. Diogo Pratas and Armando J. Pinho [26] discover a statistical compression algorithm whose compression methodology is made up of integrating six different Markov model's functional logic. Gene-Compressor [27] ,this statistical algorithm works differently than XM ,it follows three steps for compression. In the first step a Huffman encoding scheme is selected for each symbol of the sequence depending upon the probability of occurrence. In the next step the encoded output is divided into blocks and in the final step these blocks are further encoded using run-length encoding. All the above mentioned compression techniques have some similar sub-sequences present in the input sequence but if the blocks of any given input sequence is totally distinguish from each other i.e. the sub-sequences within the blocks have no common parts in them. In the 2008 I. Tabus et al. [28] tries to find the solution in their innovated algorithm. As the blocks are totally distinct from each other the algorithm treats each block individually. All the blocks are treated separately by Markov model to identify the sequence with minimum length after selection it is further compressed using an arithmetic compressor. Kalyan Kumar Kaipa et al.[29] discovers another compression algorithm based on Markov model but the algorithm only do the encoding for the non-replicated portions and dissimilar parts within a replicated sequence.

2.5. Reference Sequence Oriented Algorithm

When researchers are working on compressing genomes of same species it is found that they are highly similar and to compress this kind of genomes the referential compression approach was introduced. In this method the target genome sequence (which is to be compressed) is compressed with respect to or a set of previously known sequences (reference sequences). This approach gives best results for similar type of genomic sequences. It has been found that the compression rate of reference based algorithms is highest compared to all other types of algorithms. To get the best result good reference strings must be chosen and for that K-mer hashing technique applied where the value of k should be greater than 15 in order to avoid random matches. The main challenge with this approach is to find long suitable matching sequence and this goal can be achieved with the help of suffix tree or hash based structure. Marty C. Brandon et al.[30] proposed one algorithm where they concentrates only to store the dissimilarity between the reference sequence and the to be compressed sequence. The dissimilarity could be found in any of the following three form i.e. insertion, deletion or replacement. After pointing out the differences between the two sequences various encoding schemes like Golomb [31], Elias [32], Huffman [33] are used to do the further encoding.). Christley S et al[34] in 2009 present DNAzip algorithm , this algorithm also tries to found out the mismatches occurred between the reference sequence and the target sequence but in the form of SNPs (single-nucleotide polymorphism (SNP) or an INDEL (an insertion or a deletion of multiple bases) . Congmao Wang and Dabing Zhang [35] introduces GRS technique which finds longest common sequence between two given sequences and if the length of the matched sequence is more than a predefined threshold value then the differences between the sequences is compressed using Huffman encoding technique otherwise the reference sequence and the input sequence is divided into smaller sequences and the procedure repeats. The algorithms based on self-indexing gives better compression rates. The algorithm encodes the input sequence using LZ77 method with respect to the suffix array of the available reference

sequence. Shanika Kuruppu et al. present a reference oriented algorithm based on self-indexing structure, known as RLZ [36] algorithm is further revised and RLZopt [37] formulated where they give emphasize on local look-ahead optimization technique. This algorithm calculates longest increasing subsequence which allows efficiently encode positions. Szymon Grabowski and Sebastian Deorowicz discover another new approach [38] established on RLZopt with compression technique using LZ77 algorithm. The uniqueness of this algorithm is that it compresses the input sequence depending upon a set of reference sequences. This algorithm gives special recognition for encoding approximate matches within the sequence. Additionally Lempel-Ziv algorithm considers length of matches and distance between matched portions. Further the compression is done by shared Huffman encoding on the input blocks. GReEN [39] introduced by Armando J. Pinho et al. is another reference based model which builds upon copy-expert and it tries to find out the k-mer matching between the input sequence and the reference sequence. The algorithm works differently when it founds the length of the reference sequence and the target sequence are same. In that case it only encodes the SNPs expecting the sequence to be already aligned. Hyoung Do Kim and Ju-Han Kim produce another algorithm [40] which is a web based approach depending on LZ77 compression style with random access. Heba Afy, Muhammad Islam, and Manal Abdel Wahed [41][42] implemented a sequence alignment tool as a referenced based compression technique which is used to calculate the modified distance between pair of sequences. The sequences with least dissimilarity will be selected as reference sequence. The concept of RLZopt was further revised by Deorowicz S et al. this algorithm known as GDC [43]. The important features of this algorithm are 1) it does not select the reference sequence randomly ,2) sub-sequences of the target sequence are not necessarily be the part of reference sequence, 3) substrings are identified through approximate matching and 4) the algorithm partitioned the whole sequence into roughly equal size of blocks for encoding. It used Huffman encoding technique to ensure random access of the compress data.

2.6. Reference-Free Oriented Algorithms

Reference based methods provides best result when to be compressed sequence and the reference sequence are having similarity between them. But sometime target genomes having lack of similarities with known ones and such as in case of de novo sequencing and difference between the target sequence and the reference sequence is not properly obtainable then we need reference free compression techniques. . In 2012 Bose T, Mohammed MH, Dutta A, et al. [44] proposed BIND algorithm which encodes each target sequence using two binary strings for compression. In the first string base A or T are set with 0 and base G or C are set with 1, but in the second string base T or C are set with 0 and base A or G is set with 1. After assigning the bases with their respective bit value the length of 1 and 0 are recorded for further encoding. DELIMINATE[45] algorithm was discovered in the same year by Mohammed MH, Dutta A, Bose T where it first scan the whole sequence to find out the two most dominating bases within the sequence then it is delta encoded[46] ,after that it is deleted from the sequence and the remaining sequence is further designated using binary code. It had been noticed that DELIMINATE produces better result than general purpose algorithms such as gzip, bzip2 etc. DNAEnc3 [47] developed by Pinho AJ et al. also found to be a reference free compression technique. This algorithm is based on the performance of various Markov models. The models of different orders worked into different portions of the sequence in order to detect contextual and palindrome information from the sequence. The performance of the models is further analyzed to choose the best model for encoding the various parts of the sequence.

Thus when an appropriate reference sequence is available it is feasible to use reference based methods for better compression gain. On the other hand reference free methods are totally independent of any other reference sequence and they give not only better compression rates but also efficient in time.

III. READ COMPRESSION

While working with DNA sequencing sometimes we need to reconstruct the DNA sequence in order to get the original sequence so aligning and merging of DNA fragments are done from a longer sequence. This process is required as sequencing technology cannot read the whole genomes in one go but rather reads small portion of fragments between 20 to 30000 bases. Typically these small fragments are known as reads. In this section we will discuss some read compression technologies. This reads are aligned directly to the reference sequence and it is further used for SNP detection. The difference between genome sequencing and read sequencing is that genome sequencing can be erroneous. Each reads have a quality score which denotes the probability of the base to be in correct position , so from this quality score it can be decided whether the base is its actual position or not. This score is very essential in SNP detection methods.

3.1. Algorithms based on dictionary formation

Oscar Herrera and Angel Kuri-Morales proposes one dictionary based approach [48] which detects the rate of appearing of each meta symbols in multiple sequence. The meta symbols are nothing but the alphabetical symbols with an additional gap in between the sequence which can be further replaced by any symbol. The algorithm maintains a separate dictionary structure to store each meta symbols to

achieve better compression rate. Next in 2008 Giulia Menconi, Vieri Benci, and Marcello Buiatti discovers another approach named CASToRe [49] which is a higher version of Lempel-Ziv compression. This algorithm also maintains an additional dictionary to do the comparison between the sequences with the dictionary entries. Whenever it detects a new entry to the dictionary it stores the new entry in the form of two already stored sequence of the same dictionary. Zexuan Zhu, Jiarui Zhou, Zhen Ji, et al proposed an approach POMA [50] which is an optimization based approach. This algorithm categorizes the repeated part into four types as: direct, mirror, pairing and inverted. The fragments which reoccur at most in the sequence are added to the dictionary.

3.2. Algorithms based on statistics

Vishal Bhola, Ajit Bopardikar, Rangavittal Narayanan, et al developed a non-referential lossless compression algorithm in FASTQ format in 2011 named DSRC [51]. The file has four parts sequence identifier, raw sequence, description and quality scores. Different encoding schemes and compression techniques are used for each part of the sequence. The Sequence identifiers and description parts are further scanned to check for the needless information. Markov experts are used here to compress the raw sequences and runlength encoding scheme has been used to compress the quality score part. In 2011 Kiyoshi Asai et al. prepared a read compression technique [52] focusing on the quality score portion of any sequence. It has been observed that information within the quality score degrades after compression for FASTQ files. So they emphasizes on lossy and lossless compression of quality score. In this paper it has been shown that lossy compression for quality score gives impressive result by reducing storage cost. Waibhav Tembe et al. produce another scheme which also gives importance to quality score of any read sequence. In this new algorithm G-SQZ [53] the base value and quality score forms a pair i.e. (base, score) and each pair of this base and score is encoded using Huffman coding, further this coded contents are written into a binary file. Sebastian Deorowicz and Szymon Grabowski invented block based compression approach DSRC [54]. In this algorithm the FASTQ file is divided into three parts as identifiers, raw bases and quality score and different compression techniques are applied to those parts. The quality score pattern found to be of two type quasi-random and repetitive and for each of them two different compression technology is being used here. Huffman coding is used for quasi-random type and run-length encoding technique is used for repetitive quality score steams. In 2011 Wei-Hsin Chen et al. [55] uncovers an approach where a complete compression sequence system with a data management component and a graphical user interface is produced. In this approach the sequence length is encoded using Fibonacci code and the conflicting nucleotides are represented using 2-bit encoding.

3.3. Referenced based read compression

Identifying the mismatch occurrence of read with in a sequence with respect to a reference sequence is essential for compression. The algorithms based on reference sequence mainly contain two phases: mapping the reads and encoding them. While compression quality score became prioritized as it plays a vital role on compression rate. In GenCompress[56] read compression algorithm reads are aligned to a reference sequence, here Bowtie is used for aligning the read and then Golomb , Elias Gamma , MOV or Huffman coding is used to encode the mapping results. It stores the starting position, the match length and an optional difference list describing the mismatches with the reference string. Sequencing errors may occur at the end of reads to avoid this base mismatches are indexed form the end of the reads. GenCompress can only compress the four bases, it cannot handle any additional information or information of quality score. Christos Kozanitis, Chris Saunders, Semyon Kruglyak, et al. originate another similar approach named SlimGene [57] which done the alignment using CASAVA software toolkit. This may be a lossy or losseless compression scheme. This algorithm only emphasize on encoding technique to save the storage space. This algorithm uses Huffman encoding and arithmetic encoding. The mapping of reads and the encoding process is done by two binary vectors in this algorithm. In 2011 Fritz MH-Y et al introduces CRAM [58] algorithm , which also follow the similar approach as the previous algorithm in addition with that this process use an additional data structure de Bruijn graph to keep track of unmapped reads for a reference sequence. De Bruijn graph was also used in Quip [59], which was invented by Jones DC et al., in this algorithm the reference sequence is created form the target sequence itself. NGC [60] algorithm goes through the read alignment columns in order to find out the similarity among the multiple reads which are mapped to a particular genome sequence position. These reads are further encoded using run-length encoding technique. . In 2013 Bonfield JK et al. produces Samcomp [61] which is based on SAM format. A totally distinct approach of read compression was introduced by Markus H. Fritz et al. [62] which is based on image compression technology. The matched position of the bases is stored using Huffman coding then this base positions are further delta encoded using Golomb encoding according to their appearing order in the reference sequence.

3.4. Reference-free read compression

Sometimes for some compression implementation reference sequences selection is not possible as in de novo sequencing in that case reference-free compression methods are useful. . In 2011 Yanovsky V. ReCoil introduces a read compression algorithm ReCoil [63] which is based on the logic of constructing maximum spanning tree(MST). At first an undirected graph is built in which vertices designate the reads and the edges designates the common kmers number

between the end vertices. Depending on this graph an MST is formed and traversed from a randomly selected root node. The reads for each root nodes are stored directly and other surviving reads are encoded. BEETL [64] was presented by Cox AJ, Bauer MJ, Jakobi T, et al. in 2012 which follows the method of Burrows Wheeler Transformation (BWT) to point out the repeated reads within the sequence. Next it is further compressed using gzip, bzip2 etc. Assembling the reads which share common elements is sometime very useful technique in read compression. SCALCE[65] algorithm form a group of reads which have common "core" substrings next compression is done by gzip or LZ77 .

IV. RECENT WORK

Here we will discuss some recent invented algorithms on genome compression which is based on recent advanced technology. In august 2011 Heba Afify, Muhammad Islam and Manal Abdel Wahed present a DNA lossless differential compression algorithm [66]. This algorithm is of differential type which generates difference sequence depending upon a op-code table. Here the sequences are stored in the form of reference sequence, sequence differences and differences within the location of the sequences. Bacem Saada, Member, IAENG, Jing Zhang proposed a Modified DNABIT [67] algorithm in 2016. This is a two phase compression technique. In the first phase, modified version of DNABIT compression algorithm is used to compress and convert the DNA sequence into binary representation. In the next phase, compression is done to the resulting DNA using the Extended-ASCII encoding through which one character can represent four nucleotides or more. It gives better compression ratio than other existing algorithms. In May 2016 Rajesh Mukherjee, Subhrajyoti Mandal , Bijoy Mandal discovered an approach based on reversed sequencing [68]. This algorithm scans the whole sequence to find out the reverse substring. This algorithm maintains a library file to store the reverse substring and its corresponding original reverse substring. The reverse original substring of the algorithm setup a Dynamic Look up Table to store the ASCII character which is placed on the source file to get better compression. Recently in September 2017 Rexline S J and Trujilla Lobo F produced a DNA compression algorithm [69] using pattern recognition technique. This algorithm consists of two phases. In the first phase it finds the repeats, palindromes, complements and reverses complements and generates the Pattern Code Table. The source file is encoded using the Pattern Code during the second pass.

V. FILE FORMATS

Genome sequencing produces bulk of data even after compression. The normal files are not fitted to store these voluminous data. There are many file formats which are used to store the sequence data each having some advantages over other. Here we are going to discuss the various data formats. This file formats are mainly supported by SRA (sequence read archive). It is a data archive which stores various biological sequences so that researchers can use access this information to reproduce new sequences by comparing the various data sets.

5.1. SAM

This file format is used to store read alignment within a sequence. Heng Li invented this Sequence Alignment/ Map format which is a TAB delimited format. This format consist of two parts the header section and the alignment section. The header section should start with a "@" symbol. The alignment segment should content 11 compulsory ordered alignment related data. The header section of each data contains a 'TAG:VALUE' which is used to describe the format and content of the value. The value of the 11 compulsory alignment related data are either '0'or '*'. Those 11 fields are as follows:

- QNAME: it is string type
- FLAG: it is integer type
- RNAME: it is string type
- POS: it is integer type
- MAPQ: it is integer type
- CIGAR: it is string type
- RNEXT: it is string type
- PNEXT: it is integer type
- TLEN: it is integer type
- SEQ: it is string type
- QUAL: it is string type

5.2. BAM

This file format is the binary representation of the SAM format. BAM files are further compressed into BGZF format. BAM files also contain a header section and an alignment section. After decompression of BAM file it can be understood by human depending upon some SAM/BAM utility tools.

5.3. CRAM

This file format was discovered by EBI .This file format can be used to store compress lossless as well as lossy form of data. The advantage of this file over BAM format is that it gives better compression result than BAM. Also transition between CRAM to BAM can easily be done. In this file format data can be stored either in CRAM format or using some compressor like gzip, bzip2.

5.4. SFF

SFF or Standard Flowgram Format is a file format which can store at most 454 reads. These reads are different from normal DNA reads as it does not come up with base measurement information in place of that it provides the length details of the coming homo polymer string in the

sequence. This format possess three parts a common header segment, a read header segment and a read data segment. The data of each segment are mixture of numeric and character data.

5.5. HDF5

HDF5 is a file format which maintains a hierarchical data format. HDF5 is not only a file format but also a data model and library. It consists of two basic structures, the first one is group and the second one is dataset. The group structure is formed with any number of group or dataset with their corresponding metadata. Whereas the dataset structure is formed using array of data elements of multiple dimensions with their corresponding metadata. The group structure is composed of two parts group header and group symbol table. Group header includes name of the group header and the list of group attributes. Group symbol table on the other hand includes the list of objects which are the member of that particular group. The dataset structure has two parts the header and the data array. The header portion contains the information to access the data array and metadata.

5.6. FASTA

FASTA file format was first introduced by Bill Pearson. So it is also named as Pearson format. In this format the starting of new sequence is indicated by '>' symbol. It stores the information in plain text format so it very easily understandable. Each sequence should be of 80 characters or less than that in each line. This file format sometimes comes up along with a QUAL file which is used to store the quality score information of nucleotide.

FASTQ: FASTQ [70] is one of the most commonly used file format in genome sequencing. It is an extended version of FASTA format which are able store the quality score information of each nucleotides with in a sequence. This format was introduced by Jim Mullikin. This format is easy to understand and represent that is why it became one of the most popular used file formats in the world. This format stores a numeric value with each nucleotide of any sequence defining quality score of the nucleotides. . Initially FASTQ format was used for Sanger capillary sequencing. This particular type of FASTQ format was known as Sanger FASTQ format. The other FASTQ formats are SOLEXA, ILLUMINA + etc. In this format each information represented using four lines. The first line is indicated by '@' symbol with the sequence name. Next the sequence lines which have no bound on the number of characters appeared in a sequence. The third line is initiated by '+' symbol, in this line the information about the sequence may be repeated. Sometime it consists of a single character which helps to maintain the reduced file size. The fourth line dedicatedly used to store the quality score information.

VI. DISCUSSION AND CONCLUSION

In this survey paper we discussed four different types of genome compression algorithm and give an overview of various genome compressions and read compression algorithms. Next we provide the different file formats which are used to store the genome sequences. All the algorithms discussed above have the common goal to store the genome sequence in a compressed form to reduce the storage cost. The dictionary based algorithms emphasizes to find out the similar sequence or reverse sequence or palindromes to achieve the reduced form while the statistical based algorithms emphasizes on the probability distribution of symbols within a sequence for compression. Among all of the four types referenced based algorithm performs better with highest compression rate. This compression rate could be enhanced much more if we can incorporate cloud computing in this as it introduces parallel compression technique but in that case synchronization among various nodes would be a subject to notice for better achievement.

REFERENCES

[1] P.Raja Rajeswari (1) Allam Apparo (2), V.K. Kumar, Genbit Compress Tool(GBC): A Java-Based Tool to Compress DNA Sequences and Compute Compression Ratio(bits/base) of Genomes , Acharya Nagarjuna University, India, (2) Jawaharlal Nehru Technological University, India and (3) S.V.H. College Of Engineering, India7 Jun 2010

[2] Jacob Ziv and Abraham Lempel. A universal algorithm for sequential data compression. IEEE Transactions on Information Theory, 23(3):337-343, 1977

[3] Ateet Mehta and Bankim Patel. Dna compression using hash based data structure. International Journal of Information Technology & Knowledge Management, 3:383-386, 2010.

[4] Piyuan Lin, Shaopeng Liu, Lixia Zhang, et al. Compressed pattern matching in dna sequences using multithreaded technology. In 3rd International Conference on Bioinformatics and Biomedical Engineering, ICBBE'09, 2009.

[5] Pothuraju Rajeswari and Allam Apparao. Dnabit compress -genome compression algorithm. Bioinformation, 5(8):350-60, 2011.

[6] Grumbach, S. and Tahi, F. (1993). Compression of DNA sequences. In DCC'93:Proceedings of the Conference on Data Compression, pages 340-350.

[7] Grumbach, S. and Tahi, F. (1994). A new challenge for compression algorithms:Genetic sequences. Information Processing and Management, 30(6), 875-886

[8] Rivals, E., Delahaye, J., Dauchet, M., and Delgrange, O. (1996). A guaranteed compression scheme for repetitive DNA sequences. In DCC '96: Proceedings of the Conference on Data Compression, page 453.

[9] Chen, X., Kwong, S., and Li, M. (2000). A compression algorithm for DNA sequences and its applications in genome comparison. In RECOMB'00: Proceedings of the 4th Annual International Conference on Computational Molecular Biology, pages 107-117.

[10] Chen, X., Li, M., Ma, B., and Tromp, J. (2002). DNACompress: fast and effective DNA sequence compression. Bioinformatics, 18(12), 1696-1698.

[11] T. Matsumoto, K. Sadakane, and H. Imai. Biological sequence compression algorithms. Genome l.Informatics, 11:43–52, 2000.

[12] Kuruppu S, Beresford-Smith B, Conway T, et al. Iterative dictionary construction for compression of large DNA datasets. IEEE-ACM Trans Computat Biol Bioinformatics 2012;9:137-49
[13] Manzini, G., and Rastero, M., 2004, A Simple and Fast DNA Compressor, Software: Practice and Experience, 34(14), 1397–1411.
[14] 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.
[15] Dimitris Antoniou, Evangelos Theodoridis, and Athanasios Tsakalidis. Compressing biological sequences using selfadjusting data structures. In Information Technology and Applications in Biomedicine, 2010
[16] Kalyan Kumar Kaipa, Ajit S Bopardikar, Srikantha Abhilash, et al. Algorithm for dnasequence compression based on prediction of mismatch bases and repeat location. In Bioinformatics and Biomedicine Workshops, BIBMW, 2010.
[17] Behzadi, B. and Le Fessant, F. (2005). DNA compression challenge revisited: A dynamic programming approach. In CPM'05: Proceedings of the 16th Annual Symposium on Combinatorial Pattern Matching, volume 3537 of LNCS, pages 85-96.
[18] Matsumoto, T., Sadakane, K., and Imai, H. (2000). Biological sequence compression algorithms. Genome Informatics, 11, 43-52.
[19] 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.
[20] Korodi, G. and Tabus, I. (2005). An effcient normalized maximum likelihood algorithm for DNA sequence compression. ACM Transactions on Information Systems, 23(1), 3-34.
[21] 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.
[22] David A Huffman. A method for the construction of minimum-redundancy codes. Proceedings of the Institute of Radio Engineers, 40(9):1098-1101, 1952.
[23] 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.
[24] Allison, L., Edgoose, T., and Dix, T. I., 1998, Compression of strings with approximate repeats, Proc. ISMB, 8–16.
[25] 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.
[26] Diogo Pratas and Armando J. Pinho. Compressing the human genome using exclusively markov models. In Miguel P. Rocha, Juan M. Corchado Rodrguez, Florentino Fdez-Riverola, and Alfonso Valencia, editors, PACBB, volume 93 of Advances in Intelligent and Soft Computing, pages 213-220. Springer, 2011.
[27] K. R. Venugopal, K. G. Srinivasa, and Lalit Patnaik. Probabilistic Approach for DNA Compression, chapter 14, pages 279-289. Springer, 2009.
[28] I. Tabus and G. Korodi. Genome compression using normalized maximum likelihood models for constrained markov sources. In Information Theory Workshop, 2008.
[29] Kalyan Kumar Kaipa, Kyusang Lee, Taejin Ahn, et al. System for random access dna sequence compression. In International Conference on Bioinformatics and Biomedicine Workshops, 2010.
[30] Marty C. Brandon, Douglas C. Wallace, and Pierre Baldi. Data structures and compression algorithms for genomic sequence data. Bioinformatics, 25(14):1731-1738, 2009.
[31] Golomb S. Run-length encodings. IEEETrans InformTheory 1965;12:399–401.
[32] Elias P. Universal codeword sets and representations of the integers. IEEETrans InformTheory 1975;21:194–203.
[33] Huffman DA. A method for the construction of minimum redundancy codes. Proc IRE 1952;40:1098–101.
[34] Scott Christley, Yiming Lu, Chen Li, et al. Human genomes as email attachments. Bioinfor-matics, 25(2):274-275, 2009.
[35] Congmao Wang and Dabing Zhang. A novel compression tool for efficient storage of genome resequencing data. Nucleic Acids Research, 39(7):e45, 2011.
[36] Shanika Kuruppu, Simon J. Puglisi, and Justin Zobel. Relative lempel-ziv compression of genomes for large-scale storage and retrieval. In Proceedings of the 17th International Conference on String Processing and Information Retrieval, SPIRE'10, pages 201-206, 2010.
[37] Shanika Kuruppu, Simon Puglisi, and Justin Zobel. Optimized relative lempel-ziv compression of genomes. In Australasian Computer Science Conference, 2011.
[38] Szymon Grabowski and Sebastian Deorowicz. Engineering relative compression of genomes. CoRR, abs/1103.2351, 2011.
[39] Armando J. Pinho, Diogo Pratas, and Sara P. Garcia. Green: a tool for efficient compression of genome resequencing data. Nucleic Acids Research, 2011.
[40] Sebastian Kreft and Gonzalo Navarro. Lz77-like compression with fast random access. In Proceedings of the 2010 Conference on Data Compression, DCC'10, pages 239-248, 2010.
[41] Heba Afify, Muhammad Islam, and Manal Abdel Wahed. Dna lossless differential compression algorithm based on similarity of genomic sequence database. CoRR, abs/1109.0094, 2011.
[42] Heba Afify, Muhammad Islam, and Manal Abdel Wahed. Genomic sequences differential compressionmodel. International Journal of Computer Science and Information Technology,3:145-154, 2011.
[43] Deorowicz S, Grabowski S. Robust relative compression of genomes with random access. Bioinformatics 2011;27:2979–86.
[44] Mohammed MH, Dutta A, Bose T, et al. DELIMINATE-afast and efficient method for loss-less compression of genomic sequences. Bioinformatics 2012;28:2527–9.
[45] Pinho AJ, Ferreira PJSG, Neves AJR, et al. On the representability of complete genomes by multiple competing finite-context (Markov) models. PLoS One 2011;6:e21588.
[46] Hunt JJ, Vo K-P, Tichy WF. Delta algorithms: an empirical analysis. ACMTrans Software EngMethodol (TOSEM) 1998;7:192–214.
[47] Pinho AJ, Ferreira PJSG, Neves AJR, et al. On the representability of complete genomes by multiple competing finite-context (Markov) models. PLoS One 2011;6:e21588.
[48] Oscar Herrera and Angel Kuri-Morales. Lossless compression of biological sequences with evolutionary metadictionaries. In Workshop on Machine Learning and Data Mining, 2009.
[49] Giulia Menconi, Vieri Benci, and Marcello Buiatti. Data compression and genomes: a two dimensional life domain map. Journal of Theoretical Biology, 253(2):281-288, 2008.
[50] Zexuan Zhu, Jiarui Zhou, Zhen Ji, et al. Dna sequence compression using adaptive particle swarm optimization-based memetic algorithm. IEEE Transactions on Evolutionary Computation, 15(5):643-658, 2011.
[51] Vishal Bhola, Ajit Bopardikar, Rangavittal Narayanan, et al. No-reference compression of genomic data stored in fastq format. In Proceedings of the 2011 IEEE International Conference on Bioinformatics and Biomedicine, BIBM'11, pages 147-150, 2011.
[52] Raymond Wan, Vo N. Anh, and Kiyoshi Asai. Transformations for the compression of fastq quality scores of next generation sequencing data. Bioinformatics, 2011
[53] Waibhav Tembe, James Lowey, and Edward Suh. G-sqz: compact encoding of genomic sequence and quality data. Bioinformatics, 26(17):2192-2194, 2010.

[54] Sebastian Deorowicz and Szymon Grabowski. Compression of dna sequence reads in fastq format. Bioinformatics, 27(6):860-862, 2011.
[55] Wei-Hsin Chen, Yu-Wen Lu, Feipei Lai, et al. Integrating human genome database into electronic health record with sequence alignment and compression mechanism. Journal of Medical Systems, 36(3):2587-2597, 2011.
[56] Kenny Daily, Paul Rigor, Scott Christley, et al. Data structures and compression algorithms for high-throughput sequencing technologies. BMC Bioinformatics, 11(1):514+, 2010.
[57] Christos Kozanitis, Chris Saunders, Semyon Kruglyak, et al. Compressing genomic sequence fragments using slimgene. In Proceedings of the 14th Annual International Conference on Research in Computational Molecular Biology, RECOMB'10, pages 310-324, 2010.
[58] Fritz MH-Y, Leinonen R, Cochrane G, et al. Efficient storage of high throughput DNA sequencing data using reference based compression. GenomeRes 2011; 21:734–40.
[59] Jones DC, Ruzzo WL, Peng X, et al. Compression of nextgeneration sequencing reads aided by highly efficient denovo assembly. Nucleic Acids Res 2012;40:e171.
[60] Popitsch N, von Haeseler A. NGC: lossless and lossy compression of aligned high-throughput sequencing data. Nucleic Acids Res 2013;41:e27.
[61] Bonfield JK, Mahoney MV. Compression of FASTQ and SAM format sequencing data. PLoS One 2013;8:e59190.
[62] Markus H. Fritz, Rasko Leinonen, Guy Cochrane, et al. Effcient storage of high throughput dna sequencing data using reference-based compression. Genome Research, 21(5):734-740,2011.
[63] Yanovsky V. ReCoil - an algorithm for compression of extremely large datasets of DNA data. Algorithms Mol Biol 2011;6:23.
[64] Cox AJ, Bauer MJ, Jakobi T, et al. Large-scale compression of genomic sequence databases with the Burrows-Wheeler transform. Bioinformatics 2012;28:1415–9.
[65] Hach F, Numanagic I, Alkan C, et al. SCALCE: boosting sequence compression algorithms using locally consistent encoding. Bioinformatics 2012;28:3051–7.
[66] Heba Afify, Muhammad Islam and Manal Abdel Wahed. DNA LOSSLESS DIFFERENTIAL COMPRESSION ALGORITHM BASED ON SIMILARITY OF GENOMIC SEQUENCE DATABASE. International Journal of Computer Science & Information Technology (IJCSIT) Vol 3, No 4, August 2011 .
[67] Bacem Saada, Member, IAENG, Jing Zhang. DNA Sequences Compression Techniques Based on Modified DNABIT Algorithm. Proceedings of the World Congress on Engineering 2016 Vol I WCE 2016, June 29 - July 1, 2016, London, U.K.
[68] Rajesh Mukherjee , Subhrajyoti Mandal , Bijoy Mandal. Reverse Sequencing based Genome Sequence using Lossless Compression Algorithm. International Research Journal of Engineering and Technology (IRJET) Volume: 03 Issue: 05 ,May-2016 .
[69] Rexline S J, Trujilla Lobo F. DNA Compression Algorithm Using Pattern Hunter. International Journal on Computer Science and Engineering (IJCSE).
[70] Peter J. A. Cock,Christopher J. Fields, Naohisa Goto, Michael L. Heuer and Peter M. Rice. The Sanger FASTQ file format for sequences with quality scores, and the Solexa/Illumina FASTQ variants. Plant Pathology, SCRI, Invergowrie, Dundee DD2 5DA, UK.

Authors Profile

Mrs. Rituparna Mitra has completed Master of Science from University of Calcutta in year 2011 and Master of Technology from University of Calcutta in year 2013. She is currently working as an Assistant Teacher in the Department of Computer Science of Kirtipur Nabin Chandra High School, W.B. since January 2014. Her research area is mainly Genome data Compression. She has 4 years of teaching experience and 2 years of Research Experience.

Mr. S. Roy pursed Bachelor of Technology from University of Calcutta, India in 2010 and Master of Technology from University of Calcutta in year 2012. He is currently pursuing Ph.D. and working as Assistant Professor in Department of Computer Science and Engg. Academy of Technology, AOT, MAKAUT,India since 2013. He is a member of IEEE since 2014. He has published more than 6 research papers in reputed international journals. His main research work focuses on Genome data Compression, Big Data Analytics and IoT. He has 6 years of teaching experience and 3 years of Research Experience.

