International Journal of Applied Engineering Research ISSN 0973-4562 Volume 10, Number 15 (2015)pp35064-35067

An Empirical study of DNA Compression using Dictionary Methods and Pattern Matching in Compressed Sequences

KeerthyA S

Research Scholar, Karpagam University, Coimbatore-641021 and

Dr.Appadurai

Associate Professor, Department of Software Systems and Information Technology, Karpagam University, Coimbatore-641021

Abstract

The need for storage, analysis and transfer of genomic data is vital for the biological research community in the current era. The current methodologies of data transfer are not sufficient as the growth of data is in the range of 50-100 PB every year and is expected to rise higher. The reduction in the cost of Human Genome sequencing is pointing towards an era of personalized medicine. Hence effective management of Genetic Data is vital which makes Data Compression unavoidable. We investigate the research challenges in the field and identify the unsolved issues. Compressing DNA data reduces the cost of maintenance and transfer of data. Also pattern matching in compressed sequences reduces the cost of revealing the hidden characteristics of DNA. This paper proposesto design a novel data compression algorithm and develop pattern matching tool in compressed sequences.

Index Terms—Genomic Data, Data compression, LZW, Pattern matching, Sequence alignment.

I. INTRODUCTION

Now the storage of mass data is not a severe problem, whereas transmitting the data across the internet creates overhead of time and cost. The cost of storage can be reduced by deleting a sequence after the analysis and resequencing it later if needed. Research community considers it inappropriate and not a good technological conduct.Centralization of data effectively reduces data replication cost and makes access to huge repositories easy. The genomic data are stored in public data repositories in variant databases. The better way is to store the sequenced data by compressing it.

Compression is the process of reducing the space requirement to store data using mathematical algorithms which can be lossy or lossless.Lossless compression is mandatory as it allows reconstruction of original sequence on decompression. Lossless compression method using dictionary replaces substrings by using a dictionary built at runtime or offline. The dictionary based algorithms, detect repetitions by book keeping previously occurring sequences [1].

Repetitions in DNA sequences are accounted by simple repeats in long sequences of non-coding regions, repetition of material within a genome and existence of reverse complement. [3]

The key areas of bioinformatics and computational biology where data compression is used includes storage of biological sequences, estimating entropy, whole genome pattern matching, cataloging and indexing of genome related data, segmentation of biological sequences and pattern discovery. [9]

II. COMPRESSION OF GENOMIC DATA

Storing data in compressed format reduces the space requirement for storage and speed up circulation of data. Dictionary methods of data compression dates back to 1977-78 by Zivand Lempel (Figure1). The textual data is processed from left to right and long repetitions of consecutive characters are encoded using references to previously compressed parts of data. It is proved to be better than statistical compression method using Huffman code. But the rapid growth of data from sequential experiments demands better compression ratios [1].

Another aspect of data compression is data indexing. Since individual genome is static data, indexing is applicable and makes pattern searching easy. A notable indexing technique involves the LZ-based indexes which are efficient in removing redundancy to a great extent[2].

The peculiarity of genomic data is the high level of similarity between individuals of same species. This factor of similarity can be used for efficient compression of data by detecting redundancy and constructing dictionary while allowing fetching of individual items in any order. [3]

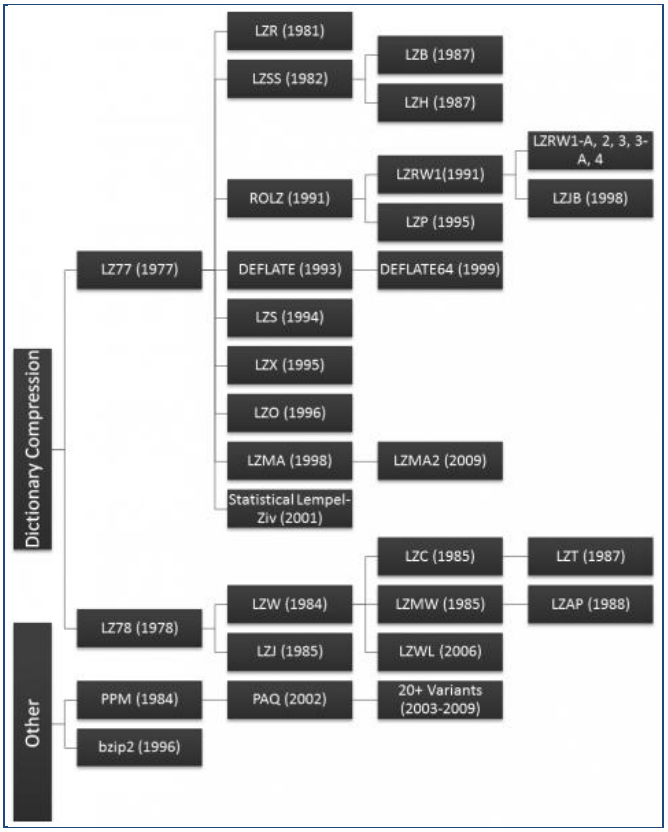

Figure:1 (Source: ETHW)

III. TEXT COMPRESSION USING LZW

Lempel and Zivintroduced Substitution Coding, making use of pointers to previous words or parts of words for compressing textual data. LZW (Lempel-Ziv-Welch) modified it by constructing a dictionary of words or parts of words in the message and then use pointers to the words in the dictionary. LZW coder uses the dictionary as a tool to generate a compressed output. The coder also expand the dictionary with new patterns and provide the compressed data output. When a string with entry in dictionary is encountered, the corresponding index is output. The limitation of original LZW is the maximum dictionary size of 4K. Suggested improvisations are to flush out the dictionary as it exceeds the size and overwrite least recently used entries in the dictionary [19].

IV. PATTERN MATCHING IN COMPRESSED SEQUENCES

Pattern matching in biological sequences arise from the desire to identify different characteristics about a DNA sequence. It assists in aligning two biological sequences and identifies the amount of similarity between them.Also pattern matching identifies common subsequences like promoters, functional motifs within a given sequence and how well a given sequence fits into a pattern [18].

Pattern matching in compressed sequences is the processof performing string matching in a compressed text without decompressing it [20].

V. LITERATURE REVIEW

A. Biological sequence Compression

A number of studies by the eminent researchers are done in literature towards different techniques of data compression and pattern matching. This paperanalyzed more relevant and recent available methodologies for data compression.

S. Kuruppu et alproposes COMRAD, a dictionary based compression technique. COMRAD uses an iterative procedure for compressing a set of DNA sequences. In COMRAD, for all iterations a frequency dictionary is created and substitution is made. It is observed that even though the first iteration of frequency dictionary creation is of O(n), subsequent iterations depend on the number of substitutions made in the previous iterations. Since it is very difficult topredict the number of substitutions made in iterations, the authors asymptotically predict a compression cost of $O(n^i_{t-1} \log n^i_{t-1})$. The space consumption in each step is directly proportional to the number of distinct pattern substrings and hence it is memory intensive [3].

Jones et al developed Quip a lossless reference based compression tool to compress NGS data. They claim three times the speed of gzip, a common tool for genome compression. The authors also claim that for single genome samples, Quip gives highest compression consistently as compared to assembly based compression [4].

Genome Resequencing Encoding (GReEn) by Pinho et al,is a reference genome based tool for compressing genome resequencing data.The tool performs compression efficiently when the target sequence is similar to reference sequence. The compression time depends on the size of sequence as well as similarity with reference sequence. Hence it is not possible to predict time of encoding for all cases [5].

HebaAfify et al propose a differential compression algorithm which uses an opcode table to identify the difference between reference sequence and target sequences. The algorithm aimed at compressing a database of sequences.After compression, the method stores a reference sequence, set of differences and locations of differences. The authors claim a 195 fold compression if the reference sequence is ideal but does not have any practical implementation [6].

Heath et al proposed a frame work to compress and manipulate a group of genomic sequences using a selected reference genome. An indexing system is used to store the differences identified [7].

DNAEncodeWG proposed by Kim et al identifies matching regions in whole genome sequence with the input query sequence. It records the characteristics of the region and differences between two sequences. The method can be applied only when the whole genome sequence of an organism is accessible through web. The encoding time totally depends on server and network status [8].

Giancarlo et al in their review paper points out various areas of bioinformatics and computational biology where compression is used. They suggested that versatility, parameter free data, association mining and speed are the main advantage of using data compression in biological investigation [9].

Toshiko et al discussed the properties of DNA sequences that enable effective compression. They proposed an algorithm that combines CTW and LZ which searches reverse

complements and approximate repeats in the sequence using hash table and dynamic programming. On successful identification,the algorithm represents the subsequence by storing its length and distance [10].

Table 1: List of dictionary based Lossless compression tools/ algorithms for whole genome sequences.

Tool/ Algorithm	Technique Used	Compression Ratio	Compression Time(MB/s)*
COMRAD	RAY	5.5	0.3
Quip	SM, AC	0.35	22
GReEn	AC	172	8.33
Differential CompressionAlgorithm	Opcode	0.005	-
Genome Compression	Huffman Coding	98.8	-
DNAEncodeWG	Diff Analysis	0.19	-
CTW+LZ	PPM, CTW	1.74	-

**Values are taken from original papers. '-' represent unknown values.*

A comparative analysis of the tools and algorithms discussed in the literature review is summarized in Table1. GReEn shows highest compression ratio with moderate speed among the tools analyzed. The authors pointed out that the success depends highly on the similarity between target sequence and reference sequence [5]. Hence it is not a reliable method for generalizing DNA compression.

The genome compression method, proposed by Heath et al, also shows good compression ratio. But it is still in research phase and no tool is developed so far based on the proposed algorithm [7].

The tools were compared based on the results provided by the authors and no common data was used to compare the tools. The lack of efficient tool to compress whole genome sequences points towards the need of a novel technology for DNA data compression and benchmarking tool.

B. Textual data compression using LZW

Rahul Gupta et al proposeda method for compressing dynamic textual data using LZW compression algorithm. They used anindexed lexicon table for storing already encountered substrings. The input text is compressed and stored as blocks of data. The index table also maintains information about the blocks. When new data is to be appended; only the corresponding block has to be decompressed and the appended data is added to the block and compressed. In standard LZW algorithm the whole compressed file has to be decompressed to append any additional data [11].

Kodituwakku and Amarasingheconducted an experimental comparison of lossless data compression algorithms for textual data. The algorithms were compared based on compression ratio, compression factor, saving percentage, compression time entropy and code efficiency. For non statistical based algorithms like RLE and LZW, entropy and code efficiency could not be calculated. It is also observed that LZW does not work well for large files as the dictionary size for compression and decompression is huge [12].

Parvinder Singh et al proposed an enhancement for improving LZW algorithm by eliminating frequent flushing of dictionary to reduce processing time. The authors point out the short comings of LZW algorithm for text data compression and suggest improvisations. Firstly whenever a dictionary gets filled a replacement strategy is initialized that replaces shorter strings by longer string for efficient compression ratio. Another suggestion is a two level dictionary modification scheme where two dictionaries are used. Primary dictionary stores frequently used entries and have smaller code size. Secondary dictionary stores codes with larger size. As the primary dictionary gets filled up the replacement strategy removes nodes from primary dictionary to secondary dictionary. This can be further enhancedwith the usage of Bi-mode Encoder. Individual bytes are sent in uncompressed mode (as it is) and sequence of bytes are sent in compressed mode (compressed using LZW)[13].

C. Pattern Matching in Compressed text files

Tao Tao and Amar Mukherjee implemented Amir's approach for pattern matching and suggest a novel algorithm for compressed pattern matching using Aho-Corasickalgorithm. The authors report a time complexity of O(n+mt+r) and space complexity of O(mt) [14].

Dictionary quasi filling by Kim et al used a modified dictionary adaptation method to remove the index coding redundancy of LZW. Additional string matching is needed by their method and is less complex than multiplication in arithmetic coding [15].

Compressed String Matching (CSM) and Fully Compressed String Matching (FCSM) differs in compressing text alone in CSM and both text and pattern in FCSM. Gasieniec and Rytter proposed sequential and parallel approaches for FCSM, for compressed text using LZW algorithm. Both procedures perform preprocessing before actual search.They claim a time complexity of O((n+m)log(n+m)) for LZW compressed sequences in the case of FCSM [16].

Collage system proposed by Kida et al is a framework based on existing dictionary based algorithm for compressed pattern matching. The system finds all occurrences of a pattern in a text without decompression. To identify multiple patterns the system needs to be modified [17].

VI. PROPOSAL

Based on the survey conducted we analyzed the lack of a unique system for compressing DNA sequences and pattern matching. So we propose a novel algorithm based on LZW for compressing DNA sequences and a system for pattern matching in compressed sequences.

With the advancements in sequencing techniques there is a big leap in the genomic data availability. As a matter of fact storage and transportation of this Big Data can be achieved effectively only when data can be compressed efficiently. The

literature review clearly points out that dictionary methods provide efficient data compression and in particular LZW is effective in compressing textual data [13]. With this situation we propose to designa compression algorithm based on LZW for genomic data which focus on reduced dictionary size.

Pattern matching techniques for biological sequences developed so far concentrates only on uncompressed data. Pattern matching in compressed data can reduce the memory and time requirements to identify the match. The literature survey brings out the importance of pattern matching in biological data and points out the success of pattern matching using LZW algorithm for textual data [16]. Hence we propose to develop a system based on LZW compression for pattern matching in compressed genomic data without decompressing it.

Identifying similarities between genomic sequences of organisms of same species can aid healthcare industry as it contributes to personalized medicine and drug discovery. Hence aligning multiple sequences is an interesting area of research. We propose to develop a system that would perform multiple sequence alignment in compressed data.

Comparing algorithms help to bring out the efficiencies and drawbacks of algorithms. Genomic data compression algorithms lack a benchmark systemfor comparison [12]. In this scenario, we propose to suggest performance measures that are applicable to all compression algorithms for effective comparison.

VII. CONCLUSIONS

In the survey we analyzed a series of methods used by previous researchers for genomic data compression and pattern matching.In this era of increasing volume of genome sequencing and resequencing, considering the cost of storing and transmitting this data, efficient compression tools are always in demand. These tools could assist in analysis of human genome variation between individuals and hence could be a key for progress in personal medicine effects.LZW has been successfully used in textual data compression, but its weakness is the increased dictionary sizehence increased computational cost. A lossless dictionary based tool using LZW with reduced dictionary size would definitely help in achieving high compression ratio and reduced computational cost and achieve pattern matching in compressed sequences.

REFERENCES

[1] Sebastian Wandelt, Marc Bux and Ulf Leser, “Trends in Genome Compression”, Current Bioinformatics, 2014, pg315-326.

[2] Sebastian Deorowicz,Szymon Grabowski,“DNA compression for sequencing Data”, Algorithms for Molecular Biology,2013

[3] ShanikaKuruppu, Bryan Beresford-Smith, Thomas Conway and Justin Zobel, “Iterative Dictionary Construction for Compression of Large DNA Data sets”, Published by IEEE/ACM Transactions on Computational Biology and Bioinformatics, 2012, pg 137-149

[4] Daniel C Jones, Walter L Ruzzo, XinxiaPeng and Michael G Katze, ”Compression of next generation sequencing reads aided by highly efficient de novo assembly”, Nucleic Acid Research, 2012.

[5] Armando J Pinho, DiogoPratas and Sara P Garcia, “GReEn: a tool for efficient compression of genome resequencing data”, Nucleic Acid Research, 2012.

[6] HebaAfify, Muhammad Islam and Manal Abdel Wahed, “DNA lossless diffrential compression algorithm based on similarity of genomic sequence database”, IJCSIT, 2011, pg 145-154.

[7] Lenwood S. Heath, Ao-ping Hou, Huadong Xia and Liqing Zhang, “A Genome Compression Algorithm Supporting Manipulation”,LSS ComputSystBioinform Conf. Vol. 9. 2010, pg 38-49.

[8] Hyoung Do Kim, Ju-Han Kim, “DNA Data Compression Based on Whole Genome Sequence”, JCIT,2009, pg 82-85.

[9] Raffaele Giancarlo, DavideScaturro and FilippoUtro, “Textual Data compression in computational biology:a synopsis”, Bioinformatics,2009, pg 1575-1586.

[10] Toshiko Matsumoto, KunihikoSadakane, Hiroshi Imai, “Biological Sequence Compression Algorithms”, Genome Informatics, 2000, pg 43-52.

[11] Rahul Gupta, Ashutosh Gupta, SunnetaAgarwal, ”A Novel Data Compression Algorithm for Dynamic Data”, IEEE, 2008, pg 266-271.

[12] S. R. Kodituwakku, U.S. Amarasinghe, “Comparison of Lossless Data Compression Algorithms for text data”, IJCSE, 2010,pg 416-425

[13] Parvinder Singh, ManojDuhan, Priyanka,”Enhancing LZW Algorithm to increase Overall Performance”, IEEE,2006, pg1-4.

[14] Tao Tao, Amar Mukherjee, “Pattern Matching in LZW Compressed Files”, IEEECS, 2005, pg 929-937

[15] Tae Young Kim, Taejong Kim, “Improving Index coding efficiency of Lempel-Ziv-Welch algorithm by dictionary quasi-filling”, Electronic Letters, 1998, 1067-1068.

[16] LeszekGasieniec and WojciechRytter, “Almost optimal fully LZW-compressed pattern matching”, Proceedings Data Compression Conference, 1999, pg 316-325.

[17] Takuya Kida, Yusuke Shibata, Masayuki Takeda, Ayumi Shinohara, SetsuoArikawa, “A unifying framework for Compressed Pattern Matching”, IEEE, 1999, pg 89-96.

[18] Eric C Rouchka, “Pattern Matching Techniques and their applications to Computational Molecular Biology”, 1999

[19] Mohammed Al-Iaham, Ibraheim M. M. ElEMary, “Comparitive study between various Algorithms of Data Compression Techniques”, IJCSNS, 2007, pg 281-291.

[20] Martin Farach, MikkelThorup, “String Matching in Lempel-Ziv Compressed Strings”, ACM, 1995, pg 703-712.