Complexity Analysis of Compressing Genomic Sequence Data with Chained Hash Indexing in Multiple Dictionary-based LZW

A.S. Keerthy*, S. Manju Priya

Abstract--- ***Data compression is the most discussed topic among the researchers as well as people working in the data industry. Huge volume of data comes from different sources and in a variety of formats like audio, video, pictures, text data, numeric data, etc. Among the variety of data available for researchers to work on, the most prominent are the genomic data produced by biological research labs. With the advent of high speed sequencing machinery and techniques, the amount of genomic data being produced is surpassing the Moore's Law. To store data proficiently and use it efficiently, compression of data is the best choice that researchers can opt for. Considering the specialty of genomic data, the compression methodology must be lossless. Keeping all these factors in consideration, a multiple dictionary based LZW compression technique was proposed and implemented. This paper computes the complexity analysis of the methodology and compares it with the currently existing ones.***

Keywords--- ***Complexity Analysis, Compression, Genomic Data, Lossless, MDLZW.***

I. INTRODUCTION

Genomic data compression is carried out lossless so that, when decompressed there is no difference between the original data and decompressed data[1]. Out of different methods of lossless compression, the most effective one for implementation is dictionary-based Liv-Zempel-Welch (LZW) compression algorithm[2]. Though the methodology dates back to early 1970s [3], various improvisations and modifications have been applied on it to make it suitable for current data explosion. In the present scenario, the scientific community is facing huge data explosion in scientific and technical fields. To cope with the work developed due to the arrival of data is the major challenge faced by research community and data scientists. With advancements in genome sequencing techniques, biologists have a handful of resources with them. They are facing it as a challenge to keep the data secure as well as analyze them [4].

To develop an efficient compression methodology, catering to the genomic data sequence was the need of the time, especially for the research community. The compression makes use of multiple dictionaries instead of a single one as in the basic model. Multiple dictionaries reduce the search time for a pattern across the dictionary. The dictionaries have chained hash indices which further make it expedient for insertion as well as searching[5]. The paper analyses the computational complexity of the comparison algorithm and presents lemmas based on the theorem.

Manuscript received February 01, 2019

A.S. Keerthy*, Department of Computer Science, Karpagam Academy of Higher Education, Coimbatore, Tamil Nadu, India. (e-mail: keerthysanthosh@gmail.com)

S. Manju Priya, Department of CS, CA & IT, Karpagam Academy of Higher Education, Coimbatore, Tamil Nadu, India. (e-mail: smanjupriya@gmail.com)

II. LITERATURE REVIEW

With advancement in technology, data flow has increased exponentially in all fields. The huge amount of data that are generated need to be analyzed properly as well as stored efficiently for future reference. Text data compression has been an area of study for the past few decades. Successful implementations have been developed for compressing or decompressing text data. Image compression, video compression, etc. are also being done. It is with the new era sequencing technology that the biologists started demanding compression for efficient data storage. The genomic data which were generated have to be analyzed and stored efficiently for future use. Also, the genomic data have to be compressed lossless, since they have to be reproduced without any character altered or missing on decompression [6,2].

Compressing genomic data has been implemented initially with text compression methods. It soon proved inefficient as the character set is different. Genomic data character set comprises A, G, C, T representing Adenine Guanine, Cytosine and Thymine respectively which are the building blocks of genomic data. The compression of genomic data was the need of the biological community[7]. Following the path of text data compression, LZW is considered as the most suitable dictionary-based lossless compression method for genomic data.

Various implementations of variations of basic LZW for text data compression has been proposed by eminent researchers [8].The major drawback in most of the implementations is the huge size of the dictionary generated. The size of the dictionary makes search through it more time consuming [9,10].To overcome the drawback of a single dictionary, Ming-Bo Lin proposed multiple dictionaries with variable-word-width [11]. He further improvised the parallel dictionary implementation by encoding the output code words using Huffman coding [12].

Perapong et al. proposed another implementation of text data compression with LZW using parallel dictionary (PDLZW), where the input data are fed into a shift register using a barrel shifter and the dictionary is updated using windowed second chance updating technique[13].

LZW with multiple indexed dictionaries proposed by Nishad and Manickachezian, brought lossless compression to a new light, where the sorted dictionaries reduce the search complexity by using binary search [14].

The authors further enhanced the method for text data by indexed k-nearest twin neighbor clustering and proposed binary insertion sort [15]. The work was successfully tested for a variety of text data and claimed to achieve94.59% compression ratio [16].

Since efficient compression for genomic data was in demand from the scientists of the biological community, various methods were proposed for compressing genomic sequence data. COMRAD used iterative frequency dictionary creation for compressing genomic sequences [17]. Quip was a reference based lossless compression tool [18]. Genome Resequencing Encoding (GReEN) was a reference based tool for compressing genomic data [19].

III. METHODOLOGY

LZW was identified as the most suitable method for lossless compression of genomic data. The basic algorithm discussed compression of text data[3]. For genomic data compression, since the character set was smaller than text data, LZW based compression *w*ith multiple dictionaries was designed and developed [5]. The algorithm for compression of genomic data assigned codes for each of the new pattern identifies and adds them to the respective dictionary D_i where 'i' is the length of the pattern identified. The algorithm is detailed in the following steps:

1. Initialize dictionary with character set and assign codes to each
2. Initialize STRING -> First character of I/P file
3. SET M ->2, CODE -> 5
4. While NOT EOF
5. Assign CHAR ->Next(CHAR) in I/P file
6. Assign L-> length(STRING+CHAR)
7. Assign INDEX -> CALCULATE_INDEX(STRING+CHAR,M)
8. SEARCH L^{th} dictionary for INDEX
   a. If INDEX found
      i. SET FLAG -> SEARCH(STRING + CHAR, L)
   b. ELSE
      i. Create new INDEX in L^{th} dictionary
9. IF FLAG IS TRUE
   a. SET STRING -> STRING + CHAR
10. ELSE
   a. Update Output File with CODE of STRING
   b. Add CODE, STRING+ CHAR to L^{th} dictionary
   c. SET CODE ->CODE+1
   d. SET STRING -> CHAR
11. Continue from step 4
12. Function CALCULATE_INDEX(X,M)
    Return $(\sum_{i=0}^{M}(HEX(to\ lower(X_i)MOD\ 5) * 4^{M-i}))$

Fig. 1: MDLZW Compression Algorithm

The computational complexity of the algorithm described in figure1 is discussed by stating a theorem on it and a few lemmas derived from the theorem.

3.1. Theorem

The computational complexity of compressing genomic data using Multiple Dictionary in Lempel-Ziv-Welch method (MDLZW)algorithm with chained hash table implementation while compression is $O\left(\frac{|D|\log|D|+\sum_{j=1}^{n}\left(\sum_{i=1}^{2}\frac{\propto_i}{D_j}\right)+|D|\sum_{j=1}^{|D|}\left(\sum_{i=1}^{|D_i|}\frac{\propto_i}{D_j}\right)}{|D|}\right)$ where |Di| is the size of i^{th} dictionary, $\propto_i$ is the time taken to search i^{th} dictionary.

Proof

Let N be the number of different characters in the character set.

Since each dictionary holds the strings of the same length, D_i denotes dictionary holding strings of length i.

Length of entries in each of the dictionary D_i = i.

Maximum number of entries in the i^{th} dictionary = N^i

For each of the pattern of length i, identified during compression, a single insertion is made into the corresponding dictionary D_i. The index within each dictionary is searched for or created if not found. To build a chained hash dictionary, time taken = $\frac{\sum_{i=1}^{n}\propto_i}{n}$.[20].

The computational complexity in index creation of dictionary

$$= \frac{\sum_{i=1}^{2}\propto_i}{|D_1|} + \frac{\sum_{i=1}^{2}\propto_i}{|D_2|} + \frac{\sum_{i=1}^{2}\propto_i}{|D_3|} + - - - + \frac{\sum_{i=1}^{2}\propto_i}{|D_n|}$$

$$= \sum_{i=1}^{j}\frac{\sum_{i=1}^{2}\propto_i}{|D_j|}$$

The complexity in searching a dictionary

$$= \sum_{i=1}^{|D_1|}\frac{\propto_i}{|D_1|} + \sum_{i=1}^{|D_2|}\frac{\propto_i}{|D_2|} + \sum_{i=1}^{|D_3|}\frac{\propto_i}{|D_3|} + - - - + \sum_{i=1}^{|D_n|}\frac{\propto_i}{|D_n|}$$

$$= \sum_{j=1}^{|D|}\left(\sum_{i=1}^{|D_i|}\frac{\propto_i}{|D_j|}\right)$$

Total computation cost calculated is sum of complexity of index creation and complexity of searching

$$= \sum_{j=1}^{n}\left(\sum_{i=1}^{2}\frac{\propto_i}{|D_j|}\right) + |D|\ \sum_{j=1}^{|D|}\left(\sum_{i=1}^{|D_i|}\frac{\propto_i}{|D_j|}\right)$$

Average computation is

$$= \frac{\sum_{j=1}^{n}\left(\sum_{i=1}^{2}\frac{\propto_i}{|D_j|}\right) + |D|\ \sum_{j=1}^{|D|}\left(\sum_{i=1}^{|D_i|}\frac{\propto_i}{|D_j|}\right)}{|D|}$$

Complexity of compression

$$= O\left(\frac{|D|\ log|D| + \sum_{j=1}^{n}\left(\sum_{i=1}^{2}\frac{\propto_i}{|D_j|}\right) + |D|\sum_{j=1}^{|D|}\left(\sum_{i=1}^{|D_i|}\frac{\propto_i}{|D_j|}\right)}{|D|}\right)$$

Lemma 1

Any dictionary D_i can have at most N^2 indices where N is the size of character set.

Proof

As explained in the algorithm for compression using MDLZW, the dictionary entries are sorted according to the index calculated using the characters of the sub sequences to be put into the dictionary.

Computational Complexity of dictionary Index= $\sum_{i=1}^{2}(N\%5)4^{i-2}$ / = O(N +N) = O(2N) = O(N)

Maximum number of indices each of the dictionary can have =|N × N| = | N^2|

Since the number of characters in the character set of the data under study (genomic data with A, T, G and C as characters) is 4,

|Dictionary Index| = 16.

Lemma 2

Any dictionary D_i can have at most 4^i entries.

Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication

Proof by induction

Length of pattern stored in Di = |i|

For a character set of N different characters, number of patterns stored in one dictionary,= $|(\prod_{i=1}^{n} N)|$

By rule of Induction,

Maximum number of entries in D_1 =|N|. N is the maximum single character entry possible for dictionary D_1.

Similarly, maximum number of entries in D_2 =$|N^2|$

And so on…

Hence by Law of Induction, Maximum number of entries in D_n =$|N^n|$

Therefore, Maximum number of entries in D_{n+1}

=$|N^n||N| = |N^{(n+1)}|$

Since the number of characters in the character set of the data used for compression is 4,

The maximum number of entries in dictionary $D_i = 4^i$

Lemma 3

*A*s the number of characters increase, the pattern repetition decreases or as the size of character set decreases the number of patterns increase.

Proof

With the 'N' characters in the character set, the possible combination of sequence of length 'r' with character repetitions permitted are = N^r

Case 1:N>r

Possibility of different characters appearing in subsequence of length r = C(n,r).

With more number of characters than the length of pattern, the number of patterns that could be generated was more and hence there would be less number of patterns. Hence, the probability of repetition of patterns was also less.

Case 2: N≤ r,

Possibility of different characters appearing in the subsequence of length 'r' = n^r With the number of characters less than the length of pattern, the patterns generated could be repeated more frequently than in the previous case. Hence pattern repetition could be more frequent when N≤r.

Lemma 4

The last dictionary generated would have index number equal to the length of the longest pattern that was generated from the data.

Proof

Each dictionary is generated when a pattern of corresponding length is encountered.

Dictionary is generated when a pattern of length 2 is detected first = D_2

For each of the new patterns generated the dictionary of corresponding length = |Pattern|

When a pattern of length 'i' is to be inserted, dictionary Di is generated if it does not exist.

Hence if the longest pattern generated is of length 'n', the last dictionary generated =D_n

IV. RESULTS AND DISCUSSION

The computational complexity of MDLZW with chained hash index table has been derived. Complexities of few other LZW based algorithms are considered for comparative analysis.LZW with RLE (Run length Encoding) claims to have complexity of O(nlog|D| + n), where D is the size of single dictionary generated with run length of encodings [21].Similarly comparison can be done with WDLZW (Word based LZW), where the space between words in the text to be compressed is eliminated and the words are encoded & entered into the single dictionary generated. WDLZW has a complexity of O(nlog(|D|)), where n is the number of words and D is the size of single dictionary generated [22]. Since MDLZW creates and uses multiple dictionaries in the place of a single dictionary used by the algorithms mentioned above, a direct comparison of complexities is not possible. Another comparable compression using multiple dictionaries for text data is by Nishad PM et al.[20]. This algorithm has a computational complexity as follows:

$$O\left(\frac{|D|\log|D|+(|D|*(\frac{1}{K}\sum_{i=1}^{K}|C_i|*\left(\log\prod_{i=1}^{|C_i|}i\right)^{\frac{1}{|C_i|}}))+|D|+\left(\frac{1}{K}\sum_{i=1}^{K}(|C_i|\log|C_i|)\right)}{|D|}\right)$$

Since this algorithm is used for compressing text data, the computational complexity is more than MDLZW designed for genomic sequence data because of the size of character set used in both cases. The text data uses a character set of 256 characters, whereas the genomic data is restricted to 4 characters.

The proposed algorithm is implemented on Intel® Core™ i7 (2.40GHz 8GB RAM) running on Windows 10 with Python 3. Testing results and comparison of performance with other algorithms will be published in the near future.

V. CONCLUSION

Complexity analysis of multiple dictionary-based compression algorithm for compressing genomic sequence data using chained hash indexing has been performed. The lemmas derived from the theorem gives more insight into the size and complexity of the dictionaries created as well as the search that happens within each of the dictionaries. The use of multiple dictionaries promises to reduce the search time and hence the compression time.

REFERENCES

1. Sayood, Khalid, Introduction to data compression, (2012), 4th ed. Morgan Kaufmann, Elsevier.
2. Wandelt, Sebastian, Marc Bux, and Ulf Leser., Trends in genome compression, Curr Bioinform 9.3, (2014), 315-326.
3. Ziv, J., and Lempel A., "A universal algorithm for sequential data compression, IEEE Trans. Inf. Theory (1977), 23.3, 337-343.
4. Keerthy A S and Dr Manju Priya S., Lempel- Ziv-Welch Compression Using Multiple Dictionaries, KJCS,(2017) Vol 11, Issue 2, 11-15.
5. Keerthy, A. S., and S. Manju Priya,(2017)Lempel-Ziv-Welch Compression of DNA Sequence Datawith Indexed Multiple Dictionaries.IJAER,12.16, 5610-5615.
6. Canovas R., and Alistair M.,2013, Practical compression for multi-alignment genomic files, Proceedings of theThirty-Sixth Austalasian Computer Science

Conference-Volume 135. Australian Computer Society, 51-60.
7. Grumbach S., and FarizaT., 1993, Compression of DNA sequences, Proceedings of Data Compression Conference, DCC'93, IEEE, 340-350.
8. Kotze H.C., and G.J. Kuhn., 1989, An evaluation of the Lempel-Ziv-Welch data compression algorithm, in Proc Communications and Signal Processing, COMSIG. Southern African Conference on IEEE, p.65.
9. Keerthy A S, Manju Priya S, 2016, Comparative analysis of Data and Compression and Pattern Matching Techniques for Biological Big Data. Available athttp://ijarcet.org, Accessed April 30, 2018
10. Keerthy A S, Appadurai, 2015, An empirical study of DNA compression using dictionary methods and pattern matching in compressed sequence, IJAER, vol 10, pg 35064-35067.
11. Ming-Bo Ling, 1997, A parallel VLSI Architecture for the LZW Data Compression Algorithm, International Symposium on VLSI Technology, Systems and Applications, 98-101.
12. MLin, Ming-Bo, Jang-Feng Lee, and Gene Eu Jan, 2006, A lossless data compression and decompression algorithm and its hardware architecture, IEEE Trans. Very Large Scale Integr. (VLSI) Syst 14, no. 9, 925 -936
13. Vichitkraivin, Perapong, and Orachat Chitsobhuk., 2009, An Improvement of PDLZW implementation with a Modified WSC Updating Technique on FPGA. World Acad Sci Eng Technol, 36,611-615.
14. Nishad, P. M., and Manicka Chezian R., 2012, Enhanced lzw (lempel-ziv-welch) algorithm by binary search with multiple dictionary to reduce time complexity for dictionary creation in encoding and decoding, IJARCSSE 2.3,192 -198.
15. Nishad, P. M. and R. Manicka Chezhian, 2012, Optimization of LZW(Lempel-Ziv-Welch) Algorithm to Reduce Time Complexity for Dictionary Creation in Encoding and Decoding, AJCSIT, 114-118
16. Nishad, P. M., and R. Manicka Chezian., 2012, Avital approach to compress the size of DNA sequence using LZW (Lempel-Ziv-Welch) with fixed length binary code and tree structure, IJCA, 43.1, 7-9.
17. Kuruppu, Shanika, et al., 2012, Iterative dictionary construction for compression of large DNA data sets. IEEE/ACM TCBB 9.1, 137-149.
18. Jones, Daniel C., et al., 2012, Compression of next-generation sequencing reads aided by highly efficient de novo assembly, Nucleic Acids Res 40.22, 171-171.
19. Pinho, Armando J., Diogo Pratas, and Sara P. Garcia., 2011, GReEn: a tool for efficient compression of genome resequencing data., Nucleic acids Res 40.4, 27-27.
20. Nishad P M, 2014, A novel approach to reduce computational complexity of multiple dictionary Lempel ziv welch mdlzw using indexed k nearest neighbor iknn clustering and binary insertion sort algorithm, [Ph.D Dissertation], Bharathiyar University, TN,India.
21. Huang W., Weimin W., and Hui Xu, 2006, A Lossless Data Compression Algorithm for Real-time Database, Intelligent Control and Automation, 2006. WCICA 2006. The Sixth World Congress on. Vol. 2. IEEE, 6645 – 6648.
22. Jiang, J., and Jones S., 1992, Word-based dynamic algorithms for data compression, IEEE Proceedings I (Communications, Speech and Vision) 139.6, 582-586.