Design and Analysis of an Improved Nucleotide Sequences Compression Algorithm Using Look up Table (LUT)

Govind Prasad Arya1, R.K. Bharti2, Devendra Prasad3

1Uttarakhand Technical University, Dehradun, Uttarakhand, India & Assistant Professor- Sikkim Manipal University, Gangtok, Sikkim, India
2BTKIT Dwarahat, Dist-Almora, Uttarakhand, India
3Uttarakhand Technical University, Dehradun, Uttarakhand, India

ABSTRACT

DNA (deoxyribonucleic acid), is the hereditary material in humans and almost all other organisms. Nearly every cell in a person’s body has the same DNA. The information in DNA is stored as a code made up of four chemical bases: adenine (A), guanine (G), cytosine (C), and thymine (T). With continuous technology development and growth of sequencing data, large amount of biological data is generated. This large amount of generated data causes difficulty to store, analyse and process DNA sequences. So there is a wide need of reducing the size, for this reason, DNA Compression is employed to reduce the size of DNA sequence. Therefore there is a huge need of compressing the DNA sequence. In this paper, we have proposed an efficient and fast DNA sequence compression algorithm based on differential direct coding and variable look up table (LUT).

1. INTRODUCTION

For decade, DNA sequence compression has become an area of research for researchers. In labs, researchers are continuously analysing these DNA sequences for various purposes. So for analysing these DNA sequences, DNA sequences need to be stored somewhere and transmitted from one place to another. But because of having very large size of DNA sequences, it results in very high transmission cost. From 1982 to present, the numbers of bases in GenBank are getting doubled approximately in every 18 months. So we require a very efficient algorithm to compress these DNA sequences. There is a direct coding algorithm which uses 2 bits for representing each of the 4 nucleotides. As DNA sequence consists of 4 nucleotide bases A, C, G & T called exons(i.e. coding regions or protein synthesis) or introns(i.e. non-coding regions or no protein synthesis), 2 bits are sufficient to represent each of the 4 bases.

1.1. Limitations of Existing DNA Compression Algorithms

a) Few existing compression algorithms such as GZIP, COMPRESS, BZIP2, WinRAR or WinZip uses more than 2 bits per byte for coding the DNA sequence. These algorithms do not utilize the common properties found in DNA sequence and this causes lower compression rate.

b) There are few algorithms like GenCompress, BioCompress which uses properties found in DNA sequences for compression. Their approximate compression rate is 1.74 bits per base i.e. 78% in compression rate. But these algorithms have very high running time.
1.2. Shortcomings of “A Compression Algorithm for Nucleotide Data Based on Differential Direct Coding and Variable Length Lookup Table (LUT)”

The existing algorithm uses ASCII code ranging from -65 to -1 for triplet stored in field length LUT, -127 to -65 for multiples of triplet (6,9,12, so on) stored in variable length LUT, ASCII code -128 was used to handle unknown character N & other ASCII codes (from 0 to 127) were used to store auxiliary characters.

Sometimes there may be an ambiguity in identifying the nucleotide base; it may either be an A or C, an A or G and so on. The extended DNA alphabet can be used to identify all these possible combinations where all the 15 possible combinations of the standard 4 nucleotides are given unique symbols. We identify the extended DNA alphabet as \( \Sigma = \{ A, B, C, D, G, H, K, M, N, R, S, T, V, W, Y \} \). The above algorithm has not used extended DNA alphabet. The data model of existing algorithm is shown in Table 1.

**Table 1. Data Model Used by Existing Algorithm**

| Type of Data      | Description                  | Range     | Look-Up Table     |
|-------------------|------------------------------|-----------|-------------------|
| Auxiliary Symbol  | ASCII                        | 0 to 127  | Fix Length LUT    |
| Triplets          | Set of three characters (basae) | -1 to -64 |                   |
| Multiple of Triplet| Set of 6, 9, 12… characters (base) | -65 to -127 | Variable Length LUT |
| Unknown           | ?                            | -128      |                   |

1.3. Proposed Algorithm

In this paper, we are proposing an algorithm which is the modification of “A Compression Algorithm for Nucleotide Data Based on Differential Direct Coding and Variable Length Lookup Table (LUT)” in following ways.

In the existing algorithm, all the ASCII codes from 0 to 127 was reserved to represent auxiliary symbols \( \{ A, G, T, C \} \) which does not utilize all ASCII codes. We have 4 DNA alphabets and 16 doubles; in total we need \( 4 + 16 = 20 \) ASCII codes. Hence we have \( 256 - 20 = 236 \) ASCII codes which can be used to represent multiples of doubles to store in variable length LUT. By using these codes we can achieve better compression as compare to existing algorithm.

Model: We consider the ASCII characters between the ranges -128 to 127. The range between (112 to 127) = 16 minus ASCII codes of respective 4 DNA alphabets \( \Sigma = \{ A, C, G, T \} \) is used to represent doubles stored in fixed size LUT and remaining ASCII codes (256-16=240) except 1(used for EOF) is used to represent multiples of doubles stored in variable size LUT except four ASCII codes used for representing 4 DNA symbols which are A, T, G, C. The data model of proposed algorithm is shown in Table 2.

Coding: Here we would encode DNA sequence using method described in the above model. We have a database which contains two tables; one Fixed Length LUT to store fixed 16 doubles and second Variable Length LUT to store maximum 235 combinations of multiples of doubles (4, 6, 8, 10 so on). We scan the DNA sequences character by character until end of file (EOF) character is encountered. Every time we read two characters (doubles) from uncompressed DNA sequence which definitely found in Fixed Length LUT, then we read next doubles, now we have a group of four characters. Initially this group of four characters is not available in Variable Length LUT. Hence we will store that group in this table and write respective ASCII code of last matched word in the output file. Whenever we find a word in Variable Length LUT, we will search another multiple of doubles (4, 6, 8, 10 and so on) in the table and if not found then store that combination of doubles in the table and write ASCII code of last matched table entry in the output file. When we find repetition of a nucleotide base like R (i.e RRRRRR), that will be written in output file as it is. The word having length less than 2 characters will also be written as it is.

**Table 2. Data Model for Proposed Algorithm**

| Type of Data             | Description                  | Range     | Look-Up Table     |
|--------------------------|------------------------------|-----------|-------------------|
| Auxiliary Symbol (4 Characters) | ASCII alphabets              | \( \Sigma = \{ A, T, G, C \} \) | NA                |
| Double (16 Words)       | Set of two base characters   | (112 to 127) - \( \Sigma \) | Fix Length LUT    |
| Multiple of Doubles (Max. 235 Words) | Set of 4,6,8… base characters | -128 to 111 | Variable Length LUT |
| Total 256 Words          |                              | -128 to 127 |                   |
2. RESEARCH METHOD

Our proposed algorithm is given below-

\[
\text{WHILE} \left( \text{EOF} \right) \{
\text{Try to read two characters (doubles) in string variable doub from uncompressed sequence}
\]

\[
\text{IF} \left( \text{length of doub} = 2 \right) \text{ and doub is a valid doubles then}
\]

\[
\text{mdoub} = \text{mdoub} + \text{doub}
\]

\[
\text{IF} \left( \text{mdoub found in Variable Length LUT} \right) \text{ then}
\]

\[
\text{Continue to while loop to read next doubles}
\]

\[
\text{ELSE}
\]

\[
\text{Write the integer code of last matched sequence from Fixed Length LUT or Variable Length LUT into output file and also store that sequence in Variable Length LUT along with new integer code}
\]

\[
\text{ELSE IF} \left( \text{length of doub} < 2 \right) \text{ then}
\]

\[
\text{Write sequence stored in doub directly to output file}
\]

\[
\text{ELSE IF} \left( \text{There is n times repetition of a character (i.e R) other than \{A, G, T, C\}} \right) \text{ then}
\]

\[
\text{Write that character as Rn in output file and also write sequence stored in doub with length < 2 directly to output file}
\]

The execution method of proposed algorithm is shown in Table 3.

| Step | Input Sequence | Doubles(t) | Multiple of Doubles(st) | Look-Up Table Status of st | Entry | Encoded Sequence(s) |
|------|----------------|------------|-------------------------|---------------------------|-------|---------------------|
| 1    | ACTGTGACTG     |            |                         |                           |       |                     |
| 2    | AC             | AC         | AC                      | Found                     | AC#   |                     |
|      |                |            |                         |                           | TG++  |                     |
| 3    | TG             | TG         | ACTG                    | Not Found                 | Add with | #                  |
| 4    | TG             | TG         | TGTG                    | Not Found                 | Add with | #+                 |
|      |                |            |                         |                           | TGAC^  | #++                |
| 5    | AC             | AC         | TGAC                   | Not Found                 | TGAC=^ | #++$               |
| 6    | TG             | TG         | ACTG                   | Found                     | ACTG= | $                   |

3. RESULTS AND ANALYSIS

Our proposed algorithm has been applied on ten types of DNA sequences shown in Table 4.

a. The results shown in Table 4 proves that our proposed algorithm would provide better compression ratio in comparison to existing methodologies to compress DNA sequences. This algorithm uses less amount of memory as compared to the other algorithms and it takes less amount of time than other algorithms and it is easy to implement as well.

b. Our proposed algorithm compresses both DNA and RNA sequences. Most of the other compression algorithms use the other properties of sequences such as repeated and non-repeated patterns in DNA sequences. If the sequence is compressed using our proposed algorithm then it would be so easier to make sequence analysis among compressed sequences. It would also be easier to make multi-sequence alignment as well. The compression results of our proposed algorithm are shown in Table 4.
4. CONCLUSION

The previous algorithm which was based on triplets was able to compress the DNA sequence up to 68% of its original size but our proposed algorithm that is based on doubles can compress the DNA sequence up to 70%. In this research paper, we have come up with the idea which is actually a modification in differential direct coding with variable length LUT. The results which we achieved using proposed algorithm, are shown below in table, are much better than the existing results. Our proposed algorithm would lead to much better compression ratio as the multiples of doubles are found frequently in DNA sequences.

REFERENCES

[1] Gregory Veyetal., Differential direct coding: a compression algorithm for nucleotide sequence data, Database (Oxford), Published online 2009 Sep 14. doi: 10.1093/database/bap013.
[2] J. Ziv and A. Lempel, “A universal algorithm for sequential data compression,” IEEE Transactions on Information Theory, vol. IT-23 NO. 3, pp. 337-343, MAY 1977.
[3] X. Chen and M. Lip, “DNA compress: fast and effective dna sequence compression,” Bioinformatics, vol. 18, Pages 1696–1698, DEC 2002.
[4] Bao, S. et al., "A DNA sequence compression algorithm based on LUT and LZ77," DOI: 10.1109/ISSPIT.2005.1577064 · Source: IEEE Xplore, pp. 1-14, January 2006.
[5] Ateet Mehta, et al, "DNA Compression using Hash Based Data Structure," IJIT & KM, vol. 2, pp. 383-386, 2010.
[6] Govind Prasad Arya and R.K. Bharti, "A Compression Algorithm for Nucleotide Data Based on Differential Direct Coding and Variable Length Lookup Table (LUT)," IJCSIT, vol. 3, pp. 4411-4416, 2012.
[7] Li Tan, et al, "K-means clustering based compression algorithm for the high-throughput DNA sequence IEEE Xplore, pp. 952-955, 2014.