An Efficient DNA Sequence Compression using Small Sequence Pattern Matching

Murugan. A[1] and Punitha. K[2],
amurugan1972@gmail.com punithsathish@gmail.com
Dr. Ambedkar Government College, Chennai, India, Agurchand Manmull Jain College, Chennai, India

Abstract
Bioinformatics is formed with a blend of biology and informatics technologies and it employs the statistical methods and approaches for attending the concerning issues in the domains of nutrition, medical research and towards reviewing the living environment. The ceaseless growth of DNA sequencing technologies has resulted in the production of voluminous genomic data especially the DNA sequences thus calling out for increased storage and bandwidth. As of now, the bioinformatics confronts the major hurdle of management, interpretation and accurately preserving of this hefty information. Compression tends to be a beacon of hope towards resolving the aforementioned issues. Keeping the storage efficiently, a methodology has been recommended which for attending the same. In addition, there is introduction of a competent algorithm that aids in exact matching of small pattern. The DNA representation sequence is then implemented subsequently for determining 2 bases to 6 bases matching with the remaining input sequence. This process involves transforming of DNA sequence into an ASCII symbols in the first level and compress by using LZ77 compression method in the second level and after that form the grid variables with size 3 to hold the 100 characters. In the third level of compression, the compressed output is in the grid variables. Hence, the proposed algorithm S_Pattern DNA gives an average better compression ratio of 93% when compared to the existing compression algorithms for the datasets from the UCI repository.
Key words:
DNA sequences, ASCII symbol, Binary Codes, compression, pattern matching, Data compression.

1. Introduction

Various kinds of computation can be performed by manipulating the DNA and related molecules [1]. Data storage and access has become a significant concern with the ceaseless increase in the data volume. This has been clearly witnessed in storing the large DNA sequences which demands ample disk space. Resultant, data compression turns out to be the most mandatory and competent approach for handling such a scenario. With compression, there is reduction in both the disk storage as well as time acquired in subsequent data processing. There has been recommendation of several techniques for handling the concerns associated with the large volume of DNA sequence data sets. Towards this, the compression-based algorithms have emerged significant as they offer efficient data storage and remove redundancy to understand the biologically important molecules [2]. Pattern matching usually refers to identifying specific pattern of characters within a huge file and is considered as the core process of DNA sequencing.

Apparently, the pattern matching algorithms performs an iterative scan over the sequence or a text. The existing research work examines the applicability of a novel Pattern matching algorithm in context with the ASCII Symbols. There has been significant application of Pattern matching algorithms in the realm of computational biology pertaining to feature extraction, searching, disease analysis and structural analysis. The suggested algorithm encourages minimum comparisons in string sequence thereby minimizing the attempts involved in each character comparison. Improvised results are gained by using the proposed algorithm in comparison with the other algorithms. Following is the organization of the paper: section 2 put forth the related work. Section 3 presents the proposed work along with the algorithm. Results acquired from the experiment in Section 4. Section 5 lay down the concluding remarks.

2. Related Work

DNA compression algorithms are classified into three types. The first category comprises of naive-bit encoding wherein the characters are depicted using a particular code word. The second category comprises of a dictionary-based or substitutional compression. Generally, there exist various repeated sequences in a DNA sequence which can be possibly replaced by references to a dictionary that may be constructed offline or maintained at runtime as mentioned in [3]. Among various dictionary-based algorithms, the popular are LZ77 and LZ78 [4] [5]. In addition, there exist some statistical methods through which effective compression rates can be obtained by generating probabilistic models with respect to the genome datasets according to [6]. The third category comprises of referential compression in which any repeated sequence acts as an in an input dataset which can be replaced with a reference to any external DNA sequences as stated in [7]. In any algorithm, data structure is of utmost significance where the aim is to acquire good compression ratios or quick pattern search within sequences. Towards this, algorithms pertaining to bioinformatics employ self-index-based data structures which help in preventing storage of large text files along with the index, as per [8]. Just by using the index alone, any part of the text can be generated again. CSA (compressed suffix array), SSA (Succinct Suffix Array), and FM-index are certain self-indexes as stated in [9].

Manuscript received August 5, 2021
Manuscript revised August 20, 2021
https://doi.org/10.22937/IJCSNS.2021.21.8.37

In addition, there exist numerous string-matching techniques that helps in resolving the issues pertaining to determining the existence of a pattern in a sequence or of a substring within an existing string [10] [11]. The following section gives an insight on diverse options of string-matching techniques, with many of them based upon algorithms like Brute-force algorithm, Bayer-Moore algorithm, Knuth-Morris-Pratt algorithms that performs exact matching in string as put forth in [12], [13]. While many other techniques rely upon approximate string-matching algorithms and dynamic programming. In IBKPMPM (Indexed based K-Partition Multiple Pattern Matching Algorithm) mentioned in [14], based upon the k value, both the string and pattern is divided into number of substrings with length k and each substring being referred to as a 'partition'. First character of all the partitions is compared. If they match, then same is continued for the second character and so on. The process continues until there is a mismatch or till the entire pattern matches with the sequence. In the technique of IBKPMPM (index based forward backward multiple pattern matching) algorithm specified in [15], matching of the characters is performed one by one in both forward and backward direction. The process continues till the entire pattern matches or there occurs a mismatch. In the technique of MSMPMA (Multiple Skip Multiple Pattern Matching Algorithm) mentioned in [16], the input text is searched for determining all occurrences of the pattern with respect to the skip technique. Index helps in determining the start location of the matching. Also, there is comparison of the Text characters from the well-defined point with the pattern characters. The match numbers help in determining the skip value (ranges 1 to m-1). In IBSPC mentioned in [17], indexes are employed for the DNA sequence. For pattern searching within the string, character index that has least occurrence is employed. Index Based Algorithm specified in [18], the index table is generated based on character index that occurs quiet often. Thereafter, the pattern is aligned with string and the occurrence of patterns is matched numerous times one by one from left to right [19][20].

The present research has put forth a novel approach that is competence enough in determining similarity among multiple patterns. Yet another pattern matching algorithm has been ascertained to elevate the functionality of pattern matching S_Pattern DNA algorithm by the means of ASCII Symbols. That is the ASCII symbols helps in determining the value comparison amidst pattern and substring. Thus, using the proposed approach, multiple pattern occurrences can be identified from a given file. The performance and result obtained from the recommended algorithm surpasses the rest other prevailing algorithms [21]. Moreover, praiseworthy results are obtained when DNA sequence dataset is employed by the above algorithm. Performance of both the proposed and exiting algorithms along with the results are being compared and presented [22]. It's quite apparent from the experimental results that the proposed approach is effective enough when applied on the DNA sequences.

3. Proposed Work

Initially, segment the DNA sequence into two, three, four, five and six data segments such that each segment is encoded like "#" for first match, "*" for second match and so on. By the means of ASCII Symbols, comparison of characters is done. Next there is description of the procedure for searching patterns in the DNA sequences (two, three, four, five and six segment). Also, a match is determined with rest of the input sequence. If there is a match, the pattern gets replaced with any symbol possessing the ASCII Symbols. The process repeats from segment size is 2 to 6 in the first level. In the second level, ASCII symbol compressed file are again compressed by using the LZ77 compression algorithm. In the third level, the grid variables are formed with size 3 and it holds 100 characters each and followed by the implementation of compression formula to generate compression ratio with a compressed sequence. It's concluded that usage of grid variable maximize the compression. The recommended approach yields significant performance and benefit by employing the ASCII symbols and grid variables in comparison to the prevailing algorithm that leads to better compression. The resultant compression ratio is clearly less than the original text. The Figure 1 depicts the overall process of proposed algorithm S_Pattern DNA.

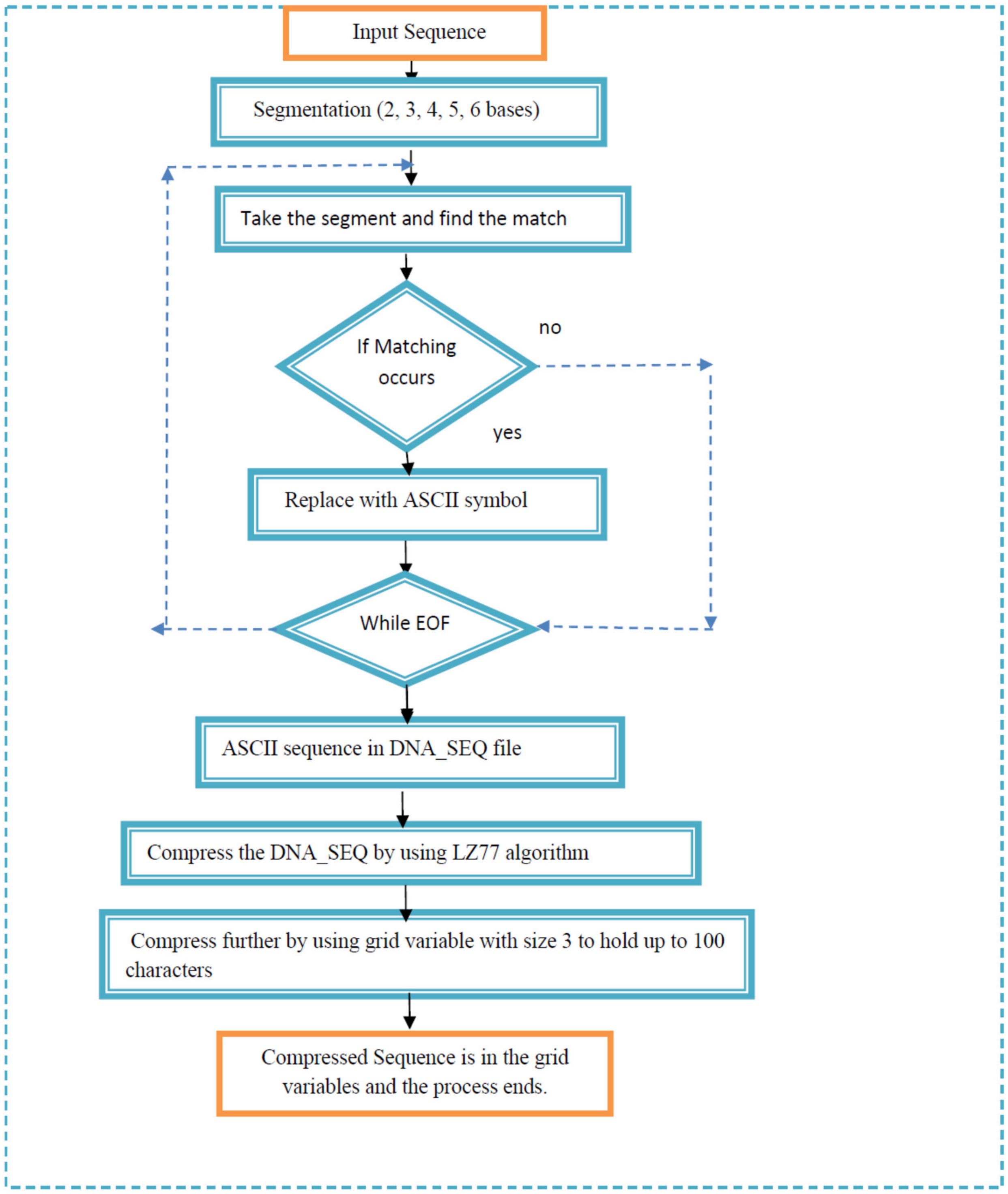
Input Sequence
Segmentation (2, 3, 4, 5, 6 bases)
Take the segment and find the match
If Matching occurs
no
yes
Replace with ASCII symbol
While EOF
ASCII sequence in DNA_SEQ file
Compress the DNA_SEQ by using LZ77 algorithm
Compress further by using grid variable with size 3 to hold up to 100 characters
Compressed Sequence is in the grid variables and the process ends.

Fig. 1 Overall Architecture of Proposed Algorithm

3.1 Proposed Algorithm

The proposed algorithm is given below to obtain the compressed sequence.

Algorithm S_Pattern DNA:
Input: DNA Sequence DNA[n]
Output: Compressed sequence

```
Begin
  Segment the sequence into 2, 3, 4, 5 and 6
  Set segment size MIN=2 and MAX=6 for segment compression in DNA[n]
  while(MAX)
      L1: P1=SEGMENT[I]
      for J = 0 to N
         if (DNA_SEQ contains P1) then
           Replace all matches by RANDOM ASCII CODE in DNA_SEQ
           REF_DECRYPT[J++]= P1
         else
           increment I by 1
           goto L1;
         end if
      end for
   end while
         Compress the ASCII CODED DNA_SEQ using LZ77 compression algorithm
     Assign the output to DNA_SEQ
     Set MAX_LEN_DNA_SEQ:= Length(DNA_SEQ)
     while( MAX_LEN_DNA_SEQ)
       for I=0 to MAX_LEN_COMPRESSED_FILE
         COMPRESS[I]= SubString(J,J+100)
         DNA_SEQ+= FSTR[I]
         J+=100
         increment I by 1
       end for
     end while
   Compressed sequence in DNA_SEQ
End
```

3.2 Implementation of the Algorithm

The steps of the algorithm for compressing an input sequence which contains character of every two, three, four, five and six segment sequence is initialized until end of file. The analysis of algorithm is as follows.
Step 1: Read the input sequence
Step 2: Take two bases (AA) from the sequence and fine the matching with remaining input sequence. If matching occurs, then replace it with any symbol which having the ASCII Symbols.

```
AAGTACGTATTTCCCTTTAGGGACGTTGGAACATTGAC
CTAGGTTAGCATTACTTTCTAAAGTACGTATGCCCCAA
AATGGGCAAGTCGTTGCCCTT
                ⇩
#GTACGTATTTCCCTTTAGGGACGTTGGAACATTGACCT
AGGTTAGCATTACTTTCT#AGTACGTATGCCCC##TGGG
C#GTCGTTGCCCTT
                ⇩
#GTACGTATTTCCCTTTAGGGACGTTGGAACATTGACCT
AGGTTAGCATTACTTTCT#AGTACGTATGCCCC##TGGG
C#GTCGTTGCCCTT
                ⇩
#*ACGTATTTCCCTTTAGGGAC*TGGAACATTGACCTAG
*TAGCATTACTTTCT#A*ACGTATGCCCC##TGGGC#*CGT
TGCCCTT
                ⇩
#*@GTATTTCCCTTTAGGG@*TGGA@ATTG@CTAG*TA
GCATT@TTTCT#A*@GTACGCCCC##TGGGC#*CGTTGCC
CTT
                ⇩
#*@$ATTTCCCTTTAGGG@*TGGA@ATTG@CTAG*TAG
CATT@TTTCT#A*@$ACGCCCC##TGGGC#*C$TGCCCTT
                ⇩
#*@$%TTCCCTTTAGGG@*TGGA@%TG@CTAG*TAGC%
T@TTTCT#A*@$ACGCCCC##TGGGC#*C$TGCCCTT
                ⇩
#*@$%^CCCT^AGGG@*TGGA@%TG@CTAG*TAGC%T@
^TCT#A*@$ACGCCCC##TGGGC#*C$TGCCC^
                ⇩
#*@$%^&CT^AGGG@*TGGA@%TG@CTAG*TAGC%T@^
TCT#A*@$ACG&##TGGGC#*C$TG&C^
                ⇩
#*@$%^&CT^AG(@*T(A@%TG@CTAG*TAGC%T@^TCT#
A*@$ACG&##T(GC#*C$TG&C^
```

Like this, the process repeats until the end of file with remaining bases from three to six and gives better compression.

3.3. Compression Ratio Calculation

Each ASCII Symbols in a DNA sequence requires at least a few bits to encode. The goal of DNA compression is to reduce the number of bits required to represent each base. The final parsed sequence are included in the proposed method's output, and the compression ratio specifies the bits per symbol (bps) and can be calculated using the following formula [24] (see Eq. 1).

$$1-\frac{(\text{No.of bits in the output})}{(\text{No.of bits in the input sequence})}X100 \qquad (1)$$

It is observed that it achieves good compression ratio as compared to existing approaches. However, the Pattern matching algorithm based on ASCII Symbols results whereas decreases
in pattern size with a good compression ratio. After applying the equation, the compression ratio of DNA sequences achieves with a reduction in storage space, and matching compression is applicable.

4. Results and Discussion

The present section put forth a number of experiments that allows comparison of proposed and existing algorithms. Thereafter, evaluation is performed with the number and size of patterns in context with the performance. Experiments are carried over different pattern sizes thereby listing the comparison output [24]. Also, the experiments are performed on the text file that is basically a collection of datasets of DNA sequences fetched from Genbank through UCI repository. Datasets employed comprises of six types of DNA sequences such as HUMAN GENES (HUMDYSTROP, HUMHDABCD, HUMHBB, HUMGHCSA, and HUMHPRTB) and being the Virus (VACCG).

For exhibiting the performance, the proposed algorithm is compared with other existing algorithms. There is no need of any pre-processing operation prior to comparison since the proposed algorithm doesn't consume any pre-processing time. And this makes the proposed algorithm efficient enough.

For verifying the existence of pattern in the given sequence, an efficient Pattern-matching algorithm has been utilized that relies upon ASCII Symbols having minimum compression ratio, time interval and complexity. The recommended approach helps in examining diverse patterns and plotting the graph through the results obtained and thereafter analyse it. It's very apparent from the experimental that the proposed small Pattern matching algorithm S_Pattern exhibits surpassed performance in contrast to other prevailing approaches and the same is being depicted in Table 1.

Table 1: Comparison of Compression ratio of Various Algorithms (in percentage)

S. No	DNA Sequences	Original File Size (Bytes)	Compression Ratio (%)		
			Optimal seed based algorithm	Improved Compression	S_Pattern DNA
1	HUMDYSTROP	38,770	82	90	99
2	HUMGHCA	66,496	90	89	95
3	HUMHBB	73,308	82	89	90
4	HUMHDABCD	58,563	84	88	92
5	HUMHPRTB	56,832	84	89	93
6	VACCG	1,91,735	84	89	90

Figure 2 depicts the proposed approach as well as existing compression ratio approaches. When compared to other existing [23][24] , the compression ratio of the proposed S_Pattern DNA algorithm based on ASCII Symbols approach achieves good compression. However, proposed small Pattern matching algorithm based on ASCII Symbols results pattern compression size is been reduced compared to other techniques with the effect on results.

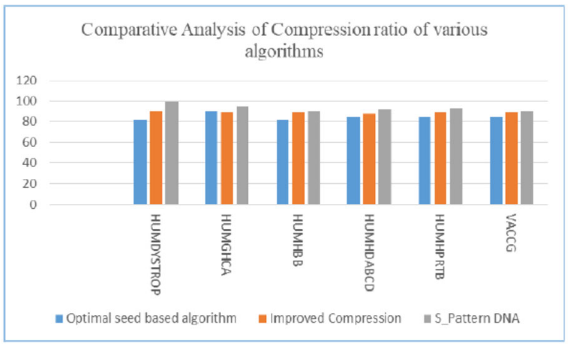

Fig. 2 Comparative Analysis of Compression ratios of various Algorithms

Table 2 depicts the comparison of compression time of proposed compression algorithm S_Pattern DNA with other existing compression algorithms [23] [24]. The S_Pattern DNA compression algorithm gives better compression ratio in fewer seconds when compared with other existing algorithms.

Table 2: Comparison of compression time of various compression algorithms (in secs)

Sequence	Optimal seed based algorithm	Improved Compression	S_Pattern DNA
HUMHDYSTROP	1.5	37	4
HUMGHCSA	2.5	65	5
HUMHBB	2.8	89	5
HUMHDABCD	2.2	63	4
HUMHPRTB	2	65	8
VACCG	4	80	21

Figure 3 represents the high speed of the proposed algorithm S_Pattern DNA when compared to the other existing compression algorithms.

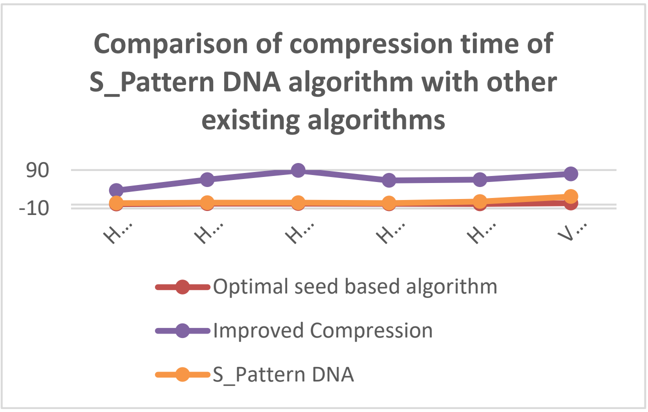

Fig. 3 Comparison of Compression time

5. Conclusion

The pattern matching algorithm S_Pattern DNA that employs ASCII Symbols for pattern matching in DNA sequences. Input sequences having any length can be handled with such an algorithm. Moreover, it helps in reducing the compression ratio as well as the total number of comparisons. The aforementioned algorithm delivers at par performance in comparison to the existing approaches as well as deal effectively with DNA sequence datasets. Unlike other commonly used algorithms, the recommended algorithm aids in minimizing the total number of comparisons along with the compression ratio. Also, repetitive patterns could be determined at a higher level of abstraction (like voluminous datasets) using the proposed approach. In future, the proposed algorithm can be extend to more segment (from 6 to more) to gain the efficient compression.

References

[1] Murugan A., Lavanya B. and Shyamala K., "A Novel Programming Approach for DNA Computing", International Journal of Computational Intelligence Research, vol. 7(2), pp. 199-209, 2011.

[2] Pothuraju Rajarajeswari and Allam Apparao, "DNABIT Compress – Genome Compression Algorithm, Bioinformation, vol. 5(8), pp. 350-360, 2011.

[3] Khairy R., Safar M., and El-Kharashi M.W., "Bloom filter acceleration: A high level synthesis approach". In: Proceedings of IEEE 30th Canadian Conference on Electrical and Computer Engineering (CCECE), pp. 1-6, Windsor, ON, Canada, 2017.

[4] Heng Li., "BGT: Efficient and flexible genotype query across many samples", Bioinformatics, vol. 32(4), pp. 590-592, 2016.

[5] Zheng X., "SeqArray—a storage-efficient high-performance data format for WGS variant calls", Bioinformatics, vol. 33(15), pp. 2251-2257, 2017.

[6] Deorowicz and Sebastian, "FQSqueezer: k-mer-based compression of sequencing data", Scientific reports, vol. 10(1), pp. 1-9, 2020.

[7] Liu Y., Yu Z. and Li J., "Index suffix-prefix overlaps by (w; k)-minimizer to generate long contigs for reads compression", Bioinformatics, vol. 35(12), pp. 2066–2074, 2018.

[8] Lau A.K., Dorrer S. and Leimeister C.A., "Read-SpaM: Assembly-free and alignment-free comparison of bacterial genomes with low sequencing coverage", BMC Bioinformatics, vol. 20(20), pp. 1-15, 2019.

[9] Milton Silva, Diogo Pratas and Armando J Pinho, "Efficient DNA sequence compression with neural networks, *GigaScience*, Vol. 9(11), pp. 1-15, 2020.

[10] Greenfield, "GeneCodeq: quality score compression and improved genotyping using a Bayesian framework", Bioinformatics, vol. 32(20), pp. 3124-3132, 2016

[11] Bonfield J. K. and McCarthy, "Crumble: reference free lossy compression of sequence quality values", Bioinformatics, vol. 35(2), pp. 337–339, 2019.

[12] Du S., Li J. and Bian N., "A compression method for DNA", PLOS ONE Journal, vol. 15(11), Article ID: e0238220, 2020.

[13] Gopinath A. and Ravisankar M, "Comparison of lossless data compression techniques", In: IEEE International Conference on Inventive Computation Technologies (ICICT), pp. 628–633, Coimbatore, India, 2020.

[14] Kavitha P., "A Survey on Lossless and Lossy Data Compression Methods," International Journal of Computer Science Engineering Technology (IJCSET), vol. 7(3), pp. 110–114, 2016.

[15] Nirmala Devi S., Rajagopalan P. and Anuradha V., "Index based multiple pattern matching algorithm using frequent character count in patterns", International Journal of Advanced Research in Computer Science and Software Engineering, vol. 3(5), 2013.

[16] Karel Brinda, "Novel computational techniques for mapping and classifying Next-Generation Sequencing data", PhD thesis, Université Paris-Est, November 2016.

[17] Bhukya, R., and Somayajulu, D. V. L. N., "Exact multiple pattern matching algorithm using DNA sequence and pattern pair. International Journal of Computer Applications, vol. 17(8), pp. 32-38, 2011.

[18] Prashant Pandey and Rob Patro, "Squeakr: an exact and approximate k-mer counting system", Bioinformatics, vol. 34(4), pp. 568-575, 2017.

[19] Numanagić, I., Bonfield, J. K., Hach, F., Voges, J., Ostermann, J., Alberti, C., and Sahinalp, S. C., "Comparison of high-throughput sequencing data compression tools", nature methods, vol. 13(12), pp. 1005-1008, 2016.

[20] Wu L., Yavas G., "Direct comparison of performance of single nucleotide variant calling in human genome with alignment-based and assembly-based approaches", Scientific reports, vol. 7(1), pp. 1-9, 2017.

[21] Danek, A., and Deorowicz, S., "GTC: a novel attempt to maintenance of huge genome collections compressed", BioRxiv, Article ID. 131649, 2017.

[22] Chikhi R., Limasset A., and Medvedev P., "Compacting de Bruijn graphs from sequencing data quickly and in low memory", Bioinformatics, vol. 32(12), pp. 201-208, 2016.

[23] Eric, Pamela Vinitha, Gopakumar Gopalakrishnan and Muralikrishnan Karunakaran, "An optimal seed based compression algorithm for DNA sequences", Advances in Bioinformatics, vol. 2016, Article ID 3528406, pp. 1-7, 2016.

[24] Punitha K. and Murugan A., "Pattern Matching Compression Algorithm for DNA Sequences", In: Proceedings of the International Conference on Sustainable Expert System, vol.176, pp. 387-402, Tribhuvan University, Nepal, 2021.