JCSE Journal of Computing Science and Engineering

Regular Paper

Journal of Computing Science and Engineering,
Vol. 15, No. 2, June 2021, pp. 59-71

Compression Techniques for DNA Sequences: A Thematic Review

Rosario Gilmary* and Akila Venkatesan
Department of Computer Science and Engineering, Pondicherry Engineering College, Pondicherry, India
rosario.gilmary@pec.edu, akila@pec.edu

Govindasamy Vaiyapuri
Department of Information Technology, Pondicherry Engineering College, Pondicherry, India
vgopu@pec.edu

Abstract
Deoxyribonucleic acid (DNA) is the basic entity that carries genetic instructions. This information is used in the evolution, progression, and improvement of all species. It is estimated that 10 CD-ROMs are required to store the genomic data of an individual being. With the increase in DNA sequencing equipment, an extensive heap of genomic data is created. The increase in DNA data in public databases is surpassing the rate of growth in storage space, thereby raising a significant concern related to data storage, transmission, retrieval, and search. To reduce the data storage and storage expense, lossless compression procedures were applied. Conventional compression methods are not proficient while compressing the biological data. Hence, several unique and contemporary lossless compression mechanisms were used to achieve improved compression ratio in biological sequences. Here, we scrutinize the diverse existing compression procedures that are appropriate for the compression of DNA sequences. The efficiency of algorithms is compared in terms of compression ratio, the ratio of the capacity of the compressed folder, and compression/decompression time. Main challenges and future research directions in DNA compression are also presented. Emphasis has been given to special references related to contemporary techniques.

Category: Bioinformatics

Keywords: DNA sequences; Lossless compression; Genomic sequence compression; Horizontal compression; Vertical compression

I. INTRODUCTION

Data compression is a technique of encoding the stored data so that they consume fewer bits. It is defined as the procedure of minimizing the quantity of data required for the transfer and storage of a given portion of information. Data compression can be lossy, hybrid, or lossless.

Lossy compression is also called irreversible compression where there is a partial loss in the data stored. Here, the original data cannot be retrieved completely from the compressed folder. The process is widely used in the compression of images and videos. Usually, it follows prediction, transformation, or quantization mechanisms. Prediction mechanisms like pulse-code modulation (PCM) represent the sampled analog data in the digital format. Transformation mechanisms carry out a series of distinctive transformations. Discrete cosine transform (DCT) and fast Fourier transform (FFT) fall under this category.

Open Access http://dx.doi.org/10.5626/JCSE.2021.15.2.59 http://jcse.kiise.org

Received 27 December 2020; **Accepted** 31 March 2021
*Corresponding Author

 pISSN: 1976-4677 eISSN: 2093-8020

Quantization techniques like wavelet quantization are performed by approximating the range of values to a distinct quantum value.

Lossless compression implies no loss of data or quality. In lossless compression, the original data, without a loss of a single bit, can be retrieved from the compressed folder. Every lossless compression technique functions by recognizing any condition of redundancy in given input and symbolizes the non-redundant data in a significant way. Lossless compression mechanisms are applied in various fields. They are used in audio files, video files, graphics, cryptography, 3D graphics, genetics, and genomes.

Hybrid compression involves the combination of both lossy and lossless compression procedures. For example, the technique is used in electrocardiogram processing and transmitting them by the global system for mobile communication (GSM).

In the field of genetics and genomes, lossless compression procedures are used to compress the genomic data of living organisms such as deoxyribonucleic acid (DNA) sequences. DNA, which embodies the genes and genomic data, is stored in 46 chromosomes. The double helix structure of the DNA is formed by coiled biopolymers. Each iota of DNA is made of a deoxyribose group, a phosphate group, and a base. DNA is a blend of four bases called adenine (A), thymine (T), cytosine (C), and guanine (G). Any specific genome is composed of three billion bases in a precise and explicit order. DNA is present in every cell, except red blood cell (RBC).

The research laboratories store the genomic data in global nucleotide databases such as the DNA Data Bank of Japan (DDBJ), the European Bioinformatics Institute EMBL databases, GenBank at the National Center for Biotechnology Information (NCBI), the Gene Expression Omnibus (GEO), and the Sequence Read Archive (SRA). Proceedings in these databases are integrated in accordance with the norms of International Nucleotide Sequence Database Collaboration (INSDC). The size of DNA sequences of different species varies between terabytes to petabytes. By October 2020 statistics, GenBank had 698,688,094,046 bases and 219,055,207 sequences, while the Whole Genome Sequencing (WGS) had 9,215,815,569,509 bases and 1,432,874,252 sequences.

This article is organized as follows. Section II explains the classification of various DNA compression techniques. In Section III, key challenges in DNA compression are presented. Section IV explains the findings and future scope in DNA compression. Finally, we conclude in Section V.

II. PROPOSED CLASSIFICATION OF DNA COMPRESSION TECHNIQUES

In the following section, various papers about DNA compression are discussed. Fig. 1 explains the classification of various DNA compression techniques. Typical lossless compression techniques designed specifically for genetic data (DNA) can be either vertical mode or horizontal mode.

A. Canonical Approaches

DNA sequences can be compressed by several typical text compression techniques. However, it is a challenge to compress the biological sequences by standard compression procedures because they are particularly made for English text files. Some distinctive compression tools include Huffman codes, PPM (prediction by partial match), CTW (context tree weighting), LZW (Lempel–Ziv–Welch), gzip, RLE (run-length encoding), and bzip2. LZ77 [1] and LZ78 [2] are the standard compression approaches introduced by the researchers, Abraham Lempel and Jacob Ziv. Both are the basic effective approaches to compress the DNA sequence [3-5]. In the LZ77 technique, a segment of the existing sequence is used like a

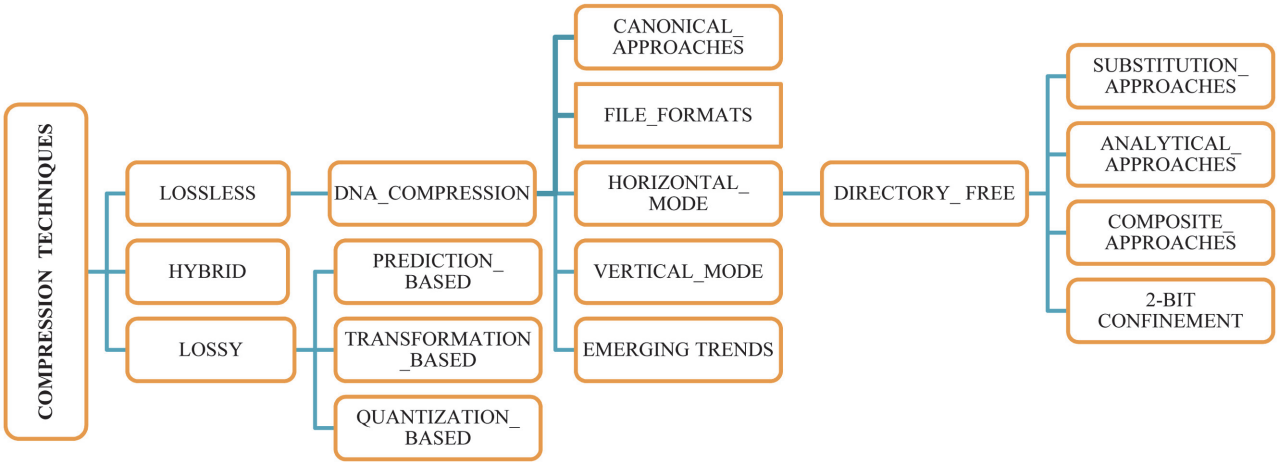

Fig. 1. Classification of DNA compression techniques.

dictionary.

The LZ77 technique involves an encoding and decoding phase. The encoder analyzes the input data through the sliding window which has a search buffer and look-ahead buffer. The search buffer encloses a segment of a late encoded sequence, whereas the look-ahead buffer has the consequent segment of the sequence to be encoded. The window is retained alike during decoding. This approach achieves a better compression ratio for many sequences and consumes only a small memory; however, encoding can be time-consuming [1]. LZ78 creates its dictionary with an entire set of bases in the input sequences which are identified earlier. LZW is a lossless algorithm that replaces strands of bases with distinct unique codes. Most of the canonical approaches such as LZW, gzip, and bzip2 are not very effective in compressing the DNA sequence because only 2.1 bpb (bits per base) of compression ratio can be achieved on an average [6]. The major disadvantage in Huffman coding is that the chances of occurrence of nucleotides are not unique [7]. Similarly, the compression ratio is not better in PPM when compared with the 2-bit condition. Though arithmetic coding and CTW are comparatively better, they do not provide better decompression speed [8].

The traditional text compression algorithms fail to compress DNA sequences. In the worst case, they may increase the content of the sequences, heading to negative compression rates. Most of the substitutional compression techniques yield negative compression rates. And statistical compression techniques yield a very small positive compression rate. Arithmetic encoding of order-2 is a comparatively better standard compression technique with a compression percentage of 3.86%. However, its outcomes rely on the likelihood of sequence occurrence and depend on previous sequences.

B. File Formats

FASTA format is widely used in genomics. In FASTA format, the sequences are depicted as codes of distinct letters [9]. The sequences can be either bases or amino acids. Every sequence is represented in two parts: (i) begins with ">" and (ii) is followed by the sequence. Multi-FASTA format is a unique pattern of interpreting diversified FASTA formatted sequences. FASTQ file format is a significant compression technique to compress a sequence of genomic data. It is very simple and is used in interchanging the file formats. The FASTA/Multi-FASTA file compression can be performed by DELIMINATE [10], MFCompress [11], gzip [12], and LZMA procedures [13]. The DELIMINATE technique is performed through delta encoding and cumulative elimination of base characters. The compression is followed by a 7-Zip archiver. The MFCompress functions by determining the following character in the sequence through order-k context. In LZMA procedures, the redundancy in the FASTQ file is removed. Later, the identifiers, the quality scores are compressed, and the reads are encoded. Finally, the processed data is compressed using LZMA. SAM (Sequence Alignment/Map) format is another file format that signifies DNA sequences [14]. It also has two parts: (i) begins with "@" and (ii) is followed by the alignment section. The binary form of SAM format is called as "BAM" format. The DNA sequences are stored in SAM format. Further, they are processed using SAM processing tools. The SAM format is an arrangement pattern against the cited sequences. DNA Zip packages play a vital role in the transfer of data, distribution of data, and replicating the data by electronic mails. The technique involves sequences of procedures to compress the size of the human genome such that they can be transmitted by email. Later, parallel sequencing of high-throughput sequencing (HTS) data was introduced. In the technique, in the process phase, concise sequences are mapped to reference genomic data. CRAM supports decisive storage by using reference-based compression techniques. Here the distinct sequences are lined up with reference genetic data. It is followed by encoding the dissimilarities between the distinct and reference genomic data in the repository [15]. SAMZIP is a specifically designed encoding technique for the data present in the SAM format. It promotes the compression ratio by utilizing the preceding information and specifications of the existing file formats and considerably promotes the transmission period. NGC [16] is a compression tool specifically designed for compressing a set of DNA stored in the SAM format. NGC supports both loss and lossless techniques. First, similar features of the read-values mapped to the positions of the genome are analyzed. It is then utilized to cut down the required codes. Howison [17] introduced a high-throughput compression procedure called SeqDB in 2013. It was presented to store NGS data so that the expense of such data gets reduced.

The SeqDB method is the combination of data-parallel byte packing technique and Blosc compressor by which high-throughput compression is achieved. Jones et al. [18] introduced QUIP, where an assembly-based compressor called de nova assembly is used. It was designed specially for compressing NGS data. Conventional assemblers use de Bruijn graphs which consume large memory. To overcome the memory issue, de Bruijn graphs are replaced by a probabilistic data structure. The outcomes of file format-based compressions are discussed in Table 1. Here, the MFCompress provides a better average performance concerning the compression ratio. However, in terms of compression/decompression time and memory, gzip yields better results.

C. Horizontal Mode of DNA Compression

DNA sequences that are compressed by taking only the substrings of the entire sequence fall under horizontal

Table 1. Evaluation results of file format-based compression techniques

Technique	Dataset	Performance metric				
		Compression ratio	Compression time (min)	Decompression time (min)	Compression memory (MB)	Decompression memory (MB)
DELIMINATE	Human	4.95	15.08	5.90	685.55	73.54
	Chimpanzee	5.36	16.64	6.62	683.31	73.55
	Rice5	5.22	0.59	0.33	368.11	43.41
	CAMERA	5.07	198.24	85.38	684.66	687.44
MFCompress	Human	4.96	34.47	29.48	1,240.45	1,266.49
	Chimpanzee	5.43	33.93	28.55	1,377.85	1,374.18
	Rice5	5.15	2.15	1.72	1,169.64	1,082.49
	CAMERA	5.39	571.74	223.61	2,764.16	2,668.16
Gzip	Human	3.59	8.85	0.50	6.87	6.87
	Chimpanzee	3.85	8.79	0.52	6.87	6.87
	Rice5	3.28	0.53	0.03	6.87	6.87
	CAMERA	3.52	141.33	7.66	6.78	6.87
LZMA	Human	4.39	117.70	1.26	199.52	31.58
	Chimpanzee	4.68	18.31	1.26	199.52	31.58
	Rice5	4.46	6.71	0.07	199.71	25.79
	CAMERA	4.51	1,720.09	16.48	199.71	31.58
QUIP	ERR174310_1	4.76	95.36	140.09	395.16	389.22
	ERR1743310_2	4.93	94.76	123.29	394.69	391.01
	ERR194146_1	4.78	326.87	438.08	397.96	392.16
	ERR194146_2	4.77	296.14	426.60	396.16	392.41

mode-based compression. Here, reference is made only to the substrings. The horizontal mode of DNA compression can be substitution approaches, analytical approaches, composite approaches, or 2-bit confinement.

1) Substitution Approaches

Substitution-based compression algorithms are techniques where the DNA sequences can be compressed by exchanging the lengthy DNA sequences with short pointer substitutes. Grumbach and Tahi [19] introduced a compression algorithm called BioCompress to compress the DNA sequences using the LZ algorithm. Here, distinct repeats and reverse complement repeats are stored using the n-ary tree that is then encoded with the length of repeat and location of prior repeat. For non-repeats parts, 2 bpb is applied. Later, a better form of DNA compression was introduced called BioCompress-2 [20], which is similar to BioCompress. It applies order-2 arithmetic coding to encode non-repeat parts. The outcomes of both the methods reveal that the mean compression ratio achieved is 1.850 bpb and 1.783 bpb for BioCompress and BioCompress-2, respectively.

Cfact [21] is a two-pass method. Here, a suffix tree is constructed followed by LZ encoding in the first and second pass, respectively. The Cfact algorithm does not have enough outcomes to compare. However, similar to BioCompress, non-repeat parts are encoded with 2 bpb. But they consume more time because of the two passes. Chen et al. [22] introduced GenCompress in two forms: GenCompress-1 and GenCompress-2 which generates a finer compression ratio of 1.742 bpb. Here, GenCompress-1 and GenCompress-2 utilize Hamming distance and edition distance, respectively. In GenCompress [22], the authors stated that it consumes many hours to compress the DNA sequence of 29k bases. DNA compress [23] is an advanced version of GenCompress. It follows two passes and uses software called PatterHunter [24]. Here, the LZ compression algorithm is used for encoding. The mean compression ratio achieved through GenCompress is 1.725 bpb. DNA-X algorithm [25] is a single pass technique and functions on LZ77. Though they encode only exact repeats for encoding, they use fall-back techniques like DNA2 or DNA3 [25]. Lee et al. [26] introduced DNAC algorithm which functions in four sections. It is an updated version of the Cfact algorithm. The algorithm constructs a suffix tree followed by

Table 2. Evaluation results of 2-bits based compression techniques

Technique	Dataset	Performance metric		Technique	Dataset	Performance metric	
		Compression rate (%)	Compression ratio			Compression rate (%)	Compression ratio
BioCompress	MIPACGA	0.48	-	GenCompress-2	MTPACGA	6.96	-
	MPOMTCG	1.39	-		MPOMTCG	5.10	-
	C1INTXX	16.26	-		CHNTXX	19.31	-
	MPOCPCG	8.44	-		CHMPXX	16.52	-
	YSCCHRIII	1.62	-		HUMGHCSA	45.14	-
	HUMGIICSA	32.75	-		HUMHBB	9.29	-
	HUMHBB	3.22	-		HUMHDABCD	9.54	-
	HUMHDABCD	3.86	-		HUMDYSTROP	3.84	-
	HUMDYSTROP	0.00	-		HUMHPRTB	8.50	-
	HUMHPRTB	2.27	-		VACCG	11.93	-
	VACCG	7.91	-		HEHCMVCG	7.64	-
	HS5HCMVCG	5.90	-	DNAC	CHMPXX	-	1.6716
BioCompress-2	MTPACGA	6.24	-		CHNTXX	-	1.6127
	MPOMTCG	3.11	-		HEHCM	-	1.8492
	CHNTXX	19.14	-		HUMDY	-	1.9116
	CHMPXX	15.76	-		HUMGH	-	1.0272
	HUMGHCSA	34.63	-		HUMHB	-	1.7897
	HUMHBB	6.16	-		HUMHD	-	1.7951
	HUMHDABCD	6.15	-		HUMHP	-	1.8165
	HUMDYSTROP	3.69	-		MPOM	-	1.8920
	HUMHPRTB	4.67	-		MTPA	-	1.8556
	VACCG	11.93	-		VACCG	-	1.7580
	HEHCMVCG	7.60	-	DNAPack	CHMPXX	-	1.6602
GenCompress-1	MTPACGA	6.96	-		CHNTXX	-	1.6103
	MPOMTCG	5.07	-		HEHCMVCG	-	1.8346
	CHNTXX	19.31	-		HUMDYSTROP	-	1.9088
	CHMPXX	16.53	-		HUMGHCSA	-	1.039
	HUMGHCSA	44.07	-		HUMHBB	-	1.7771
	HUMHBB	9.37	-		HUMHDABCD	-	1.7394
	HUMHDABCD	10.00	-		HUMHPRTB	-	1.7886
	HUMDYSTROP	3.82	-		MPOMTCG	-	1.8932
	HUMHPRTB	8.72	-		PANMTPACGA	-	1.8535
	VACCG	11.94	-		VACCG	-	1.7583
	HEHCMVCG	7.64	-				

implementing dynamic programming on the repeats. Later, non-overlapping duplicates are extracted and the Fibonacci method of encoding is followed. DNAPack by Behzadi and Le Fessant [6] uses Hamming distance as

substitutions for duplicates and complimentary duplicates. The non-repeat regions are encoded using order-2 arithmetic coding, CTW, and naïve bit encoding. Though DNAPack performs well, it is passive in execution due to complex computation. Table 2 explains the performance evaluation of different substitution-based compression techniques. In substitution-based compression techniques, DNA-X is comparatively quicker in implementation, memory efficient, and uses less space. DNAPack yields a better average compression ratio of 1.74 bpb. However, due to complex computation, it is not suitable for large datasets.

2) Analytical Approaches

DNA sequences can be compressed by several systematic and logical compression algorithms. These algorithms come under analytical approaches. CDNA, a statistical compression technique was introduced by Loewenstern and Yianilos [27]. This technique relies on prediction and probability. Here, the occurrence of each character in the sequence is determined by analyzing its presence in the past. It also estimates the possibilities of the occurrence of the character in the future. Allison et al. [28] introduced a genuine statistical approach to compress DNA sequences called the ARM algorithm. Here, all the possibly generated subsequences are considered and their corresponding probabilities are determined. CDNA and ARM algorithms have better performance and compression ratios compared with substitutional approaches. A classic statistical compressor called the XM algorithm was introduced by Cao et al. [29]. In this technique, the expert probabilities of occurrence of successive symbols are considered and combined using Bayes' Law. Later, every symbol is encoded using arithmetic encoding. XM algorithm provided a noteworthy compression ratio of 1.714 bpb. Many of these methods are expensive and time-consuming; however, they yield better compression rates for short sequences. Pinho et al. [30] proposed finite-context models. It works on the principle of two finite-context models at various orders. The performance evaluations reveal that the outcomes of this technique are better than techniques of similar computation complexity. Analytical compression techniques are explored in Table 3.

Table 3. Evaluation results of analytical compression techniques

Technique	Dataset	Performance metric
		Compression ratio
CDNA	HUMDYSTROP	1.93
	HUMGHCSA	0.95
	HUMHBB	1.77
	HUMHDABCD	1.67
	HUMHPRTB	1.72
	MPOMTCG	1.87
	VACCG	1.81
ARM	Dromaster	1.853
	HUMHBB	1.728
	CHNTXX	1.614
	YeastChr3	1.913
XM	HUMDYSTROP	1.9031
	HUMGHCSA	0.9828
	HUMHBB	1.7513
	HUMHDABCD	1.6671
	HUMHPRTB	1.7361
	MPOMTCG	1.8768
	VACCG	1.6749
Finite-context model	Yeast	1.915
	Mouse	1.780
	Arabidopsis	1.886
	Human	1.732

3) Composite Approaches

More effective compression ratios can be achieved by merging analytical and substitutional compression techniques, called composite methods. Apostolico and Lonardi [31, 32] introduced an off-line algorithm that handles the duplicated segments of the sequence. Here, accurate duplicates are considered in every redundancy check. It chooses a sub-segment that provides the best compression suffix tree. The off-line compressor provides a compression ratio of 1.938 bpb and is not designed particularly to compress DNA sequences. It is a regular compressor that functions better for DNA sequences; however, it is time-consuming. Matsumoto et al. [7] introduced a distinct algorithm, CTW+LZ. This technique uses the context tree weighting principle. It uses dynamic programming and LZ encoding. Here, the probabilities of occurrence of successive characters are analyzed based on the weighting principle. The technique recognizes the proximate duplicates and encodes them. It consumes high execution time for lengthy sequences. The mean compression ratio achieved by CTW+LZ is 1.738 bpb which is preferred over GenCompress and BioCompress-2. DNA sequences can also be compressed using the normalized maximum likelihood (NML) model for discrete regression [33]. This method partitions the sequences into blocks of fixed size. A regressor is a sub-segment with minimal Hamming distance used to compress the block, which has occurred previously. Later, encoding is done by a bitmask of order-0. GeNML [34] is an improved version of NML. Here, the sub-segments that are encoded previously are referenced and the blocks of fixed size are encoded using order-1 finite-context arithmetic encoder. GeNML achieves the compression ratio of 1.69 bpb—standard benchmark data

Table 4. Evaluation results of composite compression techniques

Technique	Dataset	Performance metric
		Compression ratio
Off-line algorithm	CHMPXX	1.9022
	CHNTXX	1.9985
	HEHCM	2.0157
	HUMDY	2.0682
	HUMGH	1.5993
	HUMHB	1.9697
	HUMHD	1.9740
	HUMHP	1.9836
	MPOM	1.9867
	MTPA	1.9155
	VACCG	1.9075
CTW+LZ	HUMDYSTROP	1.9175
	HUMGHCSA	1.0972
	HUMHBB	1.8082
	HUMHDABCD	1.8218
	HUMHPRTB	1.8433
	MPOMTCG	1.9000
	VACCG	1.7616
NML	CHMPXX	1.67
	CHNTXX	1.62
	HEHCMVCG	1.86
	HUMDYSTROP	1.91
	HUMGHCSA	1.06
	HUMHDABCD	1.74
	HUMHPRTB	1.78
	MPOMTCG	1.90
	MTPACG	1.85
	VACCG	1.78
GeNML	CHMPXX	1.66
	CHNTXX	1.61
	HEHCMV-CG	1.84
	HUMDYS-TROP	1.91
	HUMGHC-SA	1.01
	HUMHDA-BCD	1.71
	HUMHPR-TB	1.76
	MPOMTC-G	1.88
	MTPACG	1.84
	VACCG	1.76
DNASC	CHMPXX	1.50
	CHNTXX	1.51
	HEHCMV-CG	1.80
	HUMDYS-TROP	1.89
	HUMGHC-SA	0.91
	HUMHDA-BCD	1.61
	HUMHPR-TB	1.71
	MPOMTC-G	1.88
	MTPACG	1.80
	VACCG	1.70

except for HUMHBB. GeNML is excelling DNAPack, CTW+LZ, GenCompress, and BioCompress-2 both in terms of compression ratio and implementation time. DNASC is a compressor that considers five-characters A, T, G, C, and N. The technique compresses the sequence horizontally by extended LZ method followed by horizontal compression considering blocks of size 6 and window of size 128. This technique provides a better compression ratio than GenCompress, BioCompress-2, CTW+LZ, GeNML, and DNA compress. This technique provides the mean compression ratio of 1.54 bpb that is a standard benchmark data except for HUMHBB. Perfor- mance evaluations of composite compression techniques are discussed in Table 4.

4) 2-BIT Confinement

In the 2-bit codation method, each of the bases is assigned unique two bits; A=00, T=11, C=10, and G=01. The non-repetitive portions, as well as repetitive portions, are compressed. And the input sequence is partitioned into segments of four, 8-bit characters. Rajarajeswari et al. [35] introduced a Java-based tool for compression of DNA sequences called a Genbit compress tool (GBC). Initially, two bits for each character are assigned. Thus, 256 combinations are obtained. The entire input sequence is converted into the number of 8-bit numbers. A single fragment of four bases is represented by an 8-bit number. If successive portions are similar, then a distinct bit 1 is appended as the 9th bit. If successive portions are dissimilar, then a distinct bit 0 is appended as the 9th bit. The rest of the bases are assigned 4 exclusive two bits. For a non-biological sequence, the best, average, and worst cases are 1.125 bpb, 1.727 bpb, and 2.238 bpb, respectively. GBC provides a mean compression ratio of 2.23 bpb definitive benchmarks except for HUMHDABCD, HEHCMVCG, HUMGHCSA, and MPOMTCG. DNABIT Compress introduced by Rajarajeswari and Apparao [36] is a 2-bit-based compression that accredits exclusive bit code for exact repeats and reverses repeats. DNABIT yields 1.58 bpb—standard benchmark (besides HUMGHCSA and MIPACGA). GenCodex [37] has two modules that use multi-cores and graphical processing units. Here, the focal point is throughput and compression ratio. It yields a better compression ratio for biological sequences with maximum repeats than the one with fewer repeats. For both serial and parallel versions, the compression ratio remains identical. If the DNA sequence has a fragment that is not duplicated in it, then it is represented by a single byte, whereas if the fragment is repeated in the sequence, then it is represented using two bytes. The first byte represents the fragment and the second byte represents the repetitions. HUFFBIT is also a compression procedure comprising two major steps and was introduced specifically for DNA sequences by Rajeswari et al. [38]. Initially, an extended binary tree is designed followed by extracting variable length codes (Huffman codes)

Table 5. Evaluation results of 2-bits based techniques

Dataset	Technique	Performance metric
		Average compression ratio
MPOMTCG	GENBIT	1.727
HEHCMVCG	HUFFBIT	1.611
HUMGHCSA	DNABIT	1.580
MIPACGA	GenCodex	1.420
HUMGHCSA	DNACRAMP	1.143
HUMHDABCD	PGBC	1.330

from the binary tree. It is used to determine the bit count from the encoded sequence length. The procedure was subjected to unreal DNA sequences. HUFFBIT yields a finer compression ratio of 1.006 bpb, 1.611 bpb, and 2.109 bpb for best, average, and worst case, respectively. DNACRAMP was proposed by Prasad and Kumar [15] for both repetitive and non-repetitive sequences. In this procedure, the compression ratio of the DNA sequences gets modified linearly concerning the sequence such that the accomplishment made is O (n). It yields a compression ratio of 1.143 bpb—standard benchmark. The principal aim of partitioned group binary compression (PGBC) is to support the unfavorable conditions of previous algorithms. PGBC considers the length of the input DNA sequence and divides them into four sections. The procedure is followed by encoding where every six sections are clubbed to form a single partition. Every partition is provided with a respective unique bit code before sub partitioning them. Later, every sub partition is grouped into a single partition. By applying PGBC, the compression ratio achieved is 1.33 bpb. Table 5 explains the compression ratio of different 2-bit based techniques.

Among the 2-bit based techniques, the performance of GenCodex is better than other algorithms in terms of compression ratio and throughput. However, most of these techniques use dynamic programming. Thus, their implementation is difficult and time-consuming. Conversely, PGBC algorithms are less complex and quick in implementation.

D. Vertical Mode of DNA Compression

In the vertical mode of DNA compression, information corresponding to two sequences is considered, where one of the sequences is considered as the reference sequence. These are directory-based compression mechanisms where the input data is read and checked for the same in the dictionary. If present, the index value is updated. Regular Huffman-based coding is an encoding procedure that attains 65% to 85% of compression. The coding method used here involves interpreting absolute representations that compress the data without modifying the relative form. Afify et al. [39] introduced the differential compression algorithm to compress the genomic data. The compression technique involves considering the similarity in the sequence directory. Re-Pair [30] helps in choosing the reference sequence. Re-Pair uses an offline directory-based compression technique which is highly expensive. Relative Lempel–Ziv (RLZ) [40] uses Lempel–Ziv compression to store and retrieve massive genomic data. The proposed technique stores index values of the base sequence and is followed by compressing the additional sequences by LZ77 encoding. Improved RLZ [41] uses improved Lempel–Ziv compression over the genomic data. Here, an elementary non-greedy parsing technique is used that is more efficient, cheaper, supports rapid decompression, and supports increased compression by 50% compared with the initial RLZ technique. DSRC [42] was developed to compress sequences in FASTQ format. It is free of charge and provides a better compression ratio as well as increased performance. Here, quality scores play a major role. The quality scores represent the uncertainties in sequence identification processes. Genome re-sequencing (GRS) [43] is an encoding compression tool that was developed concerning an increase in reference genome sequences. This procedure does not use single nucleotide polymorphisms commonly called SNP. Here similarity degree between the reference sequence and target sequence is determined using Huffman encoding. A decompression tool called the de novo compression tool was introduced to decompress the sequences compressed using GRS tool. GReEn [44] was introduced to overcome the disadvantages of GRS. It uses the reference genetic sequence to compress the re-sequencing data of the genome. It is also accessible without charge. DNA-COMPACT [45] is a lossless two-pass compression procedure introduced to compress the biological sequences. The performance is accomplished by considering the complementary contextual models without considering the reference sequences. The proposed algorithm yields a compression ratio of 1.8 bpb and 1.7 bpb for the experiments conducted on yeast and bacteria which is comparatively higher than other algorithms. Efficient referential genome compressor (ERGC) [46] is an example of the algorithm that runs in stages where each stage is independent of its prior stage. This procedure follows a greedy alignment algorithm based on hashing. The performance is determined regarding compression ratio and running time. ERGC provides better decompression compared with other algorithms. FRESCO (FRamework for REferential Sequence Compression) [47] is an open-source framework for the compression of massive genetic data. It produces better compression speed, i.e., one to two orders in magnitude quicker than the existing proposals. In FRESCO, the compression ratio is raised by choosing an analogously superior reference sequence and

Table 6. Evaluation results of vertical mode-based compression techniques

Technique	Dataset	Performance metric
		Compression rate
GRS	TIGR6. Ref: TIGR5	82.0
		Encoded size
Huffman-based coding (G_SQZ)	SRR013951_2	1.12
	SRR027520_1	1.40
	SRR027520_2	1.44
	SRR007215_1	0.13
	SRR010637	0.49
	SRR014961_2	9.35
		Average compression ratio
FRESCO	10 random sequences	2.3
RLZ-standard	10 random sequences	10.7
		0-order entropy (bpb)
RLZ-improved	S. cerevisiae	0.15
	S. paradoxus	0.25
	H. sapiens	0.48
		(bpb)
DNA-COMPACT	NC_013929	1.7153
	NC_01438	1.7025
	NC_013595	1.7189
	NC_013131	1.7366
	NC_010162	1.7230
		Compression ratio
Differential modal compression	SH-S5	0.1527
	SM-S13	0.0417
	SM-S17	0.0417
	SM-S20	0.0417
DSRC	ERR174310_1	4.64
	ERR174310_2	4.82
	ERR194146_1	4.72
	ERR194146_2	4.70
GReEn	HS8, Ref: HSCHM8	102.98
	HS11, Ref: HSCHM11	60.74
	HS11, Ref: PI11	11.99
	HS11, Ref: PA11	4.21
	HSK16, Ref: HS16	45.88
	RICE16, Ref: RICE7	169.16
ERGC	HS8, Ref: HSCHM8	4.47
	HS11, Ref: HSCHM11	4.30
	HS11, Ref: PI11	4.30
	HS11, Ref: PA11	4.30
	HSK16, Ref: HS16	9.16
	RICE16, Ref: RICE7	4.44
GDC	10 random sequences	2.00
GDC2	HS8, Ref: HSCHM8	239.72
	HS11, Ref: HSCHM11	110.97
	HS11, Ref: PI11	27.49
	HS11, Ref: PA11	6.21
	HSK16, Ref: HS16	193.34
	RICE16, Ref: RICE7	169.16
iDoComp	HS8, Ref: HSCHM8	241.77
	HS11, Ref: HSCHM11	104.08
	HS11, Ref: PI11	29.19
	HS11, Ref: PA11	6.44
	HSK16, Ref: HS16	173.93
	RICE16, Ref: RICE7	204.38
CoGI	YH, Ref: KOREF_20090224	244.42

rewriting them. It is followed by a second-order referential compression. The outcomes conclude that a better compression ratio is achievable in modern hardware. GDC 2 [48] is a compression algorithm for compressing a large collection of genetic data. The proposed algorithm can compress 1,092 diploid genomes about 9,500 times. It is comparatively better in terms of compression speed and ratio and takes a processing speed of 200 MB per second only thereby consuming less cost for storage. The outcome is better by four times. CoGi method [49] is a technique for the compression of the genetic data as an image which differs from other techniques that compress the sequences in a single dimension. Here, the genome is compressed to a two-dimensional bitmap. Then, the bit map is compressed using rectangular partition coding. CoGi mainly focuses on simple implementation rather than optimization procedures. The experimental outcomes prove that CoGi can provide a compression ratio higher than GReEn by eight times. iDoComp [50] is a compression scheme employed for the assembled genomes. It performs both compression and decompression of genetic sequences by considering the reference DNA strings. The technique infers that it is better than the outcomes of previous compression procedures by 60%. Performance evaluations of vertical mode-based compression techniques are discussed in Table 6.

E. Emerging Trends

DNA sequences can be compressed by several new-fangled compression techniques. Here, the most recent compression algorithms are considered. Law [51] discussed the analogy of assorted DNA compression techniques. The study provides an overview of reference-based and references free techniques. Here, two sections of DNA compression were introduced. The first section considers the DNA sequence which possesses identical sub sequences. The second section considers similarities within the sequence. The main importance was given to the choice of the reference sequence. The reference sequence is picked as one in population sequences or considered via statistics of population sequences. The better compression ratio is achieved by complaint choice of reference sequence which can be designed and modified. Cheng et al. [52] designed a clustering-based DNA compression algorithm. Here, the presence of substructures within the DNA sequences is considered. The sequences are segregated by applying clustering mechanisms. Initially, a reference sequence is designed for every cluster. Further, a unique reference sequence is designed considering all the reference sequences of every cluster. With the help of a reference sequence, each cluster is compressed. The method yields better compression by 91% compared with the existing techniques. However, the compression time is compromised. Neha and Salim [53] stated that exceptional compression ratios can be accomplished by considering the explicit encoding of the differences in the sequence. Here, rather than scrutinizing only the similarities present in the sequences, importance is given to the distinctness. DNA sequences can be compressed through a single block encoding scheme [54]. It compresses the DNA sequences in three phases. Initially, the largest base is determined. Its locations in the sequences are denoted by 1. The rest of the bases with lower frequencies are denoted by 0. In phase two, encoding is performed through a single-block encoding scheme. Later, each block is assigned a shorter length code. GeCo3 [55] is a neural network-based DNA compressor. It uses the concept of expert mixing of multiple contexts and substitution-tolerant context models. Here, only the probabilities of symbols in the sequence are considered. They yield better compression at the cost of computational time. IonCRAM [56] is a compression technique for compressing ion torrent sequence files. Here, BAM files are compressed using parallel processing. The flow signals are processed parallelly and the scramble compresses the BAM files. These BAM files are decomposed into sub-files. Each sub-file corresponds to the genomic region. The process achieves a space-saving of 43%. The memRGC [57] identifies the maximum matches between the DNA sequences. It considers certain mutations and the surrounding maximum matches. Thus, the codes of the target sequence are reduced. The compression ratio achieved is better than the existing techniques. Decompression time and memory usage are also reduced.

III. CHALLENGES IN DNA COMPRESSION

Comparison of different compression techniques for DNA sequences is a challenging task. It has two major concerns. First, the source file (implementation code) is not available in most cases. Thus, the compression algorithm cannot be tested on all the datasets. On the other hand, most of the compression techniques need more time and storage requirements. Thus, large DNA sequences cannot be easily tested.

IV. FINDINGS AND FUTURE SCOPE

Each technique discussed in the study has some shortcomings. Certain techniques are sensitive to large datasets. Most of the algorithms that yield significant compression ratio consume heavy computational time. Some techniques are not good for statistical analysis. Among the substitution algorithms, certain types of repeats are searched. Later, they are encoded using a significant algorithm. The LZ technique is the reliable default choice. Most of the analytical approaches provide effective compression; however, they are computationally intensive in practice. Compression techniques with the combination of substitution and analytical frameworks provide better performance. Bit-based compression procedures can be applied to both repetitive and non-repetitive genomic data. They are less complex and quick in implementation since they do not use dynamic programming. In vertical mode-based compression techniques, the important observation is the drastic variance in the performance measure from dataset to dataset. The efficiency of the techniques depends on both the target sequence and the reference sequence. “Proximity” of the target sequence to the reference sequence appears to be a crucial factor. Combining the methods of vertical mode compression and analytical compression can increase the compression ratio by determining other criteria to select a reference sequence. In the future, modern graphical representation approaches should be introduced for an effective and easy interpretation of this complex organized sequence data wherein, the semantics of the data must not be lost. Current methods often fail to take into account the specific structure of the sequence data. This might help in preserving the sequence precisely. Also, existing techniques can be subordinated to other techniques centered on statistical procedures for bandaging DNA classifications at a pace that is more advanced than the proportion of conventional techniques.

V. CONCLUSION

This article investigates the different techniques employed to compress genomic data. Here, a thematic study is conducted on different DNA compression methods. In this study, the techniques used to compress DNA sequences, algorithms, datasets, and evaluation metrics are considered. The primary challenges faced in the compression of DNA sequences are briefly discussed. The findings inferred from the study and future research direction in this domain are also addressed. Despite the adoption of the best compression strategies, the outcomes are still inefficient concerning the amount of genomic data produced. Thus, it continues to be an open challenge for the research community.

ACKNOWLEDGMENTS

This work has been supported by Research Grant No. SPG/2020/000594 under the SERB POWER grant scheme, Science and Engineering Research Board, Government of India, to Akila Venkatesan, Pondicherry Engineering College, India.

REFERENCES

1. L. Felician and A. Gentili, "A nearly optimal Huffman technique in the microcomputer environment," *Information Systems*, vol. 12, no. 4, pp. 371-373, 1987.
2. M. Nelson and J. L. Gailly, *The Data Compression Book*, 2nd ed. New York, NY: M&T Books, 1995.
3. A. Apostolico and A. Fraenkel, "Robust transmission of unbounded strings using Fibonacci representations," *IEEE Transactions on Information Theory*, vol. 33, no. 2, pp. 238-245, 1987.
4. J. B. Connell, "A Huffman-Shannon-Fano code," *Proceedings of the IEEE*, vol. 61, no. 7, pp. 1046-1047, 1973.
5. J. L. Bentley, D. D. Sleator, R. E. Tarjan, and V. K. Wei, "A locally adaptive data compression scheme," *Communications of the ACM*, vol. 29, no. 4, pp. 320-330, 1986.
6. B. Behzadi and F. Le Fessant, "DNA compression challenge revisited: a dynamic programming approach," in *Annual Symposium on Combinatorial Pattern Matching*. Heidelberg, Germany: Springer, 2005, pp. 190-200.
7. T. Matsumoto, K. Sadakane, and H. Imai, "Biological sequence compression algorithms," *Genome Informatics*, vol. 11, pp. 43-52, 2020.
8. H. Sato, T. Yoshioka, A. Konagaya, and T. Toyoda, "DNA data compression in the post genome era," *Genome Informatics*, vol. 12, pp. 512-514, 2001.
9. D. J. Lipman and W. R. Pearson, "Rapid and sensitive protein similarity searches," *Science*, vol. 227, no. 4693, pp. 1435-1441, 1985.
10. M. H. Mohammed, A. Dutta, T. Bose, S. Chadaram, and S. S. Mande, "DELIMINATE: a fast and efficient method for loss-less compression of genomic sequences: sequence analysis," *Bioinformatics*, vol. 28, no. 19, pp. 2527-2529, 2012.
11. A. J. Pinho and D. Pratas, "MFCompress: a compression tool for FASTA and multi-FASTA data," *Bioinformatics*, vol. 30, no. 1, pp. 117-118, 2014.
12. gzip: a data compression utility, http://www.gzip.org/.
13. LZMA SDK (software development kit), https://www.7-zip.org/sdk.html.
14. H. Li, B. Handsaker, A. Wysoker, T. Fennell, J. Ruan, N. Homer, G. Marth, G. Abecasis, and R. Durbin, "The sequence alignment/map format and SAMtools," *Bioinformatics*, vol. 25, no. 16, pp. 2078-2079, 2009.
15. V. H. Prasad and P. V. Kumar, "a new revised DNACRAMP tool based approach of chopping DNA repetitive & non repetitive genome sequences," *International Journal of Computer Science Issues (IJCSI)*, vol. 9, no. 6, pp. 448-454, 2012.
16. L. Sun, H. Luo, D. Bu, G. Zhao, K. Yu, C. Zhang, Y. Liu, R. Chen, and Y. Zhao, "Utilizing sequence intrinsic composition to classify protein-coding and long non-coding transcripts," *Nucleic Acids Research*, vol. 41, no. 17, pp. e166-e166, 2013.
17. M. Howison, "High-throughput compression of FASTQ data with SeqDB," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 10, no. 1, pp. 213-218, 2013.
18. D. C. Jones, W. L. Ruzzo, X. Peng, and M. G. Katze, "Compression of next-generation sequencing reads aided by highly efficient de novo assembly," *Nucleic Acids Research*, vol. 40, no. 22, pp. e171-e171, 2012.
19. S. Grumbach and F. Tahi, "Compression of DNA sequences," in *Proceedings of the Data Compression Conference (DCC)*, Snowbird, UT, 1993, pp. 340-350.
20. S. Grumbach and F. Tahi, "A new challenge for compression algorithms: genetic sequences," *Information Processing & Management*, vol. 30, no. 6, pp. 875-886, 1994.
21. E. Rivals, J. P. Delahaye, M. Dauchet, and O. Delgrange, "Fast discerning repeats in DNA sequences with a compression algorithm," *Genome Informatics*, vol. 8, pp. 215-226, 1997.
22. X. Chen, S. Kwong, and M. Li, "A compression algorithm for DNA sequences and its applications in genome comparison," *Genome Informatics*, vol. 10, pp. 51-61, 1999.
23. X. Chen, M. Li, B. Ma, and J. Tromp, "DNACompress: fast and effective DNA sequence compression," *Bioinformatics*, vol. 18, no. 12, pp. 1696-1698, 2002.
24. B. Ma, J. Tromp, and M. Li, "PatternHunter: faster and more sensitive homology search," *Bioinformatics*, vol. 18, no. 3, pp. 440-445, 2002.
25. G. Manzini and M. Rastero, "A simple and fast DNA compressor," *Software: Practice and Experience*, vol. 34, no. 14, pp. 1397-1411, 2004.
26. A. J. T. Lee, C. Chang, and C. Chen, "DNAC: an efficient compression algorithm for DNA sequences," National Taiwan University, Taipei, Taiwan, 2004.
27. D. Loewenstern and P. N. Yianilos, "Significantly lower entropy estimates for natural DNA sequences," in *Proceedings of the 7th Data Compression Conference (DCC)*, Snowbird, UT, 1999, pp. 151-160.
28. L. Allison, T. Edgoose, and T. I. Dix, "Compression of strings with approximate repeats," in *Proceedings of the 6th International Conference on Intelligent Systems for Molecular*

Biology (ISMB), Montreal, Canada, 1998, pp. 8-16.
29. M. D. Cao, T. I. Dix, L. Allison, and C. Mears, "A simple statistical algorithm for biological sequence compression," in *Proceedings of 2007 Data Compression Conference (DCC)*, Snowbird, UT, 2007, pp. 43-52.
30. A. J. Pinho, A. J. Neves, D. A. Martins, C. A. Bastos, and P. J. S. G. Ferreira, P. J. S. G. "Finite-context models for DNA coding," in *Signal Processing*. Rijeka, Croatia: InTech, 2010, pp. 117-130.
31. A. Apostolico and S. Lonardi, "Off-line compression by greedy textual substitution," *Proceedings of the IEEE*, vol. 88, no. 11, pp. 1733-1744, 2000.
32. A. Apostolico and S. Lonardi, "Compression of biological sequences by greedy off-line textual substitution," in *Proceedings of the Data Compression Conference (DCC)*, Snowbird, UT, 2000, pp. 143-152.
33. I. Tabus, G. Korodi, and J. Rissanen, "DNA sequence compression using the normalized maximum likelihood model for discrete regression," in *Proceedings of the Data Compression Conference (DCC)*, Snowbird, UT, 2003, pp. 253-262.
34. G. Korodi and I. Tabus, "An efficient normalized maximum likelihood algorithm for DNA sequence compression," *ACM Transactions on Information Systems (TOIS)*, vol. 23, no. 1, pp. 3-34, 2005.
35. P. Rajarajeswari, A. Apparo, and V. K. Kumar, "GENBIT COMPRESS TOOL (GBC): a Java-based tool to compress DNA sequences and compute compression ratio (bits/base) of genomes," *International Journal of Computer Science and Information Technology*, vol. 2, no. 3, pp. 181-191, 2010.
36. P. Rajarajeswari and A. Apparao, "DNABIT Compress - Genome compression algorithm," *Bioinformation*, vol. 5, no. 8, pp. 350-360, 2011.
37. D. Satyanvesh, K. Balleda, A. Padyana, and P. K. Baruah, "GenCodex: a novel algorithm for compressing DNA sequences on multi-cores and GPUs," in *Proceedings of IEEE 19th International Conference on High Performance Computing (HiPC)*, Pune, India, 2012.
38. P. R. Rajeswari, A. Apparao, and R. K. Kumar, "Huffbit compress: algorithm to compress DNA sequences using extended binary trees," *Journal of Theoretical and Applied Information*, vol. 13, no. 2, pp. 101-106, 2010.
39. H. Afify, M. Islam, M. A. Wahed, and Y. M. Kadah, "Genomic sequences differential compression model," *International Journal of Computer Science and Information Technology*, vol. 3, pp. 145-154, 2011.
40. S. Kuruppu, S. J. Puglisi, and J. Zobel, "Relative Lempel-Ziv compression of genomes for large-scale storage and retrieval," in *String Processing and Information Retrieval*. Heidelberg, Germany: Springer, 2010, pp. 201-206.
41. S. Kuruppu, S. J. Puglisi, and J. Zobel, "Optimized relative Lempel-Ziv compression of genomes," in *Proceedings of the 34th Australasian Computer Science Conference*, Perth, Australia, 2011, pp. 91-98.
42. S. Deorowicz and S. Grabowski, "Compression of DNA sequence reads in FASTQ format," *Bioinformatics*, vol. 27, no. 6, pp. 860-862, 2011.
43. C. Wang and D. Zhang, "A novel compression tool for efficient storage of genome resequencing data," *Nucleic Acids Research*, vol. 39, no. 7, pp. e45-e45, 2011.
44. A. J. Pinho, D. Pratas, and S. P. Garcia, "GReEn: a tool for efficient compression of genome resequencing data," *Nucleic Acids Research*, vol. 40, no. 4, pp. e27-e27, 2012.
45. P. Li, S. Wang, J. Kim, H. Xiong, L. Ohno-Machado, and X. Jiang, "DNA-COMPACT: DNA compression based on a pattern-aware contextual modeling technique," *PloS One*, vol. 8, no. 11, article no. e80377, 2013. https://doi.org/10.1371/journal.pone.0080377
46. S. Saha and S. Rajasekaran, "ERGC: an efficient referential genome compression algorithm," *Bioinformatics*, vol. 31, no. 21, pp. 3468-3475, 2015.
47. S. Wandelt and U. Leser, "FRESCO: referential compression of highly similar sequences," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 10, no. 5, pp. 1275-1288, 2013.
48. S. Deorowicz, A. Danek, and M. Niemiec, "GDC 2: compression of large collections of genomes," *Scientific Reports*, vol. 5, article no. 11565, 2015. https://doi.org/10.1038/srep11565
49. X. Xie, S. Zhou, and J. Guan, "CoGI: towards compressing genomes as an image," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 12, no. 6, pp. 1275-1285, 2015.
50. I. Ochoa, M. Hernaez, and T. Weissman, "iDoComp: a compression scheme for assembled genomes," *Bioinformatics*, vol. 31, no. 5, pp. 626-633, 2015.
51. B. N. F. Law, "Application of signal processing for DNA sequence compression," *IET Signal Processing*, vol. 13, no. 6, pp. 569-580, 2019.
52. K. O. Cheng, N. F. Law, and W. C. Siu, "Clustering-based compression for population DNA sequences," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 16, no. 1, pp. 208-221, 2019.
53. A. S. Neha and A. Salim, "Towards context-aware DNA sequence compression algorithms," in *Proceedings of 2019 IEEE International Conference on Electrical, Computer and Communication Technologies (ICECCT)*, Coimbatore, India, 2019, pp. 1-4.
54. D. Mansouri, X. Yuan, and A. Saidani, "A new lossless DNA compression algorithm based on a single-block encoding scheme," *Algorithms*, vol. 13, no. 4, article no. 99, 2020. https://doi.org/10.3390/a13040099
55. M. Silva, D. Pratas, and A. J. Pinho, "Efficient DNA sequence compression with neural networks," *GigaScience*, vol. 9, no. 11, article no. giaa119, 2020. https://doi.org/10.1093/gigascience/giaa119
56. M. Shokrof and M. Abouelhoda, "IonCRAM: a reference-based compression tool for ion torrent sequence files," *BMC Bioinformatics*, vol. 21, article no. 397, 2020. https://doi.org/10.1186/s12859-020-03726-9
57. Y. Liu, L. Wong, and J. Li, "Allowing mutations in maximal matches boosts genome compression performance," *Bioinformatics*, vol. 36, no. 18, pp. 4675-4681, 2020.

Rosario Gilmary https://orcid.org/0000-0003-3754-9809

Rosario Gilmary is currently pursuing her Ph.D. in the Department of Computer Science and Engineering at Pondicherry Engineering College affiliated to Pondicherry University, Pondicherry, India She received her postgraduate degree in Computer Science and Engineering from St. Joseph's College of Engineering affiliated to Anna University, Chennai, India, in 2018. Her areas of interest are social networks and data analytics.

Akila Venkatesan https://orcid.org/0000-0003-2639-1452

Akila Venkatesan is working as an Assistant Professor in the Department of Computer Science and Engineering at Pondicherry Engineering College, Pondicherry, India. Her specializations include social network analysis and mining software repositories. She has more than 17 years of experience in teaching and has about 20 papers to her credit. She is a recipient of grants for projects under UGC minor, AICTE MODROB, and SERB Power Grant.

Govindasamy Vaiyapuri https://orcid.org/0000-0002-9371-8868

Govindasamy Vaiyapuri is working as an Associate Professor in the Department of Information Technology at Pondicherry Engineering College, Pondicherry, India. His research interest includes online social networks and data analytics. He is also specialized in Business Intelligence and Complex Event Processing. He has more than 17 years of experience in teaching and has about 25 papers to his credit. He is also the Principal Investigator of the AICTE RPS Project.