International Journal of Engineering & Technology, 7 (4) (2018) 5844-5850

SPC

International Journal of Engineering & Technology

Website: www.sciencepubco.com/index.php/IJET
doi: 10.14419/ijet.v7i4.14873
Research paper

Bounded probability based textual data compression for fiber-optic communication

Bosco Paul Alapatt [1] *, A. Kavitha [2]

[1] Research Scholar, Department of Computer Science, Bharathiar University, Coimbatore, India
[2] Assistant Professor, Department of Computer Science, Gongunadu Arts and Science College, Coimbatore, India
**Corresponding author E-mail: boscoallapatt@gmail.com*

Abstract

Fiber optic communication becomes very popular due to its nature of high data rate. Though fiber optic communication offers faster data transmission, it suffers from the drawback of massive amount of data being generated, stored or transmitted. Data compression techniques are introduced to minimize the size of data which eventually reduces the bandwidth utilization, storage space and data transmission at a faster rate. This paper presents a new dictionary based encoding technique called Bounded probability based textual data compression algorithm called BPT algorithm. The BPT algorithm generates a codeword based on the dictionary, which contains the binary code based on the probability of occurrence of characters in the input data. For decompression, there is a need to transmit the coding table along with the compressed data. The proposed BPT algorithm is tested using a set of benchmark textual dataset from The Calgary Corpus and The Canterbury Corpus. The experimental results verified the superiority of the BPT algorithm over the state of art methods in terms of different measures namely compression ratio (CR), compression factor (CF), bits per character (bpc) and space savings.

Keywords: *Fiber Optic Communication; Dictionary Based Coding; Data Compression; Textual Dataset*.

1. Introduction

For every web-oriented applications and services, fiber-optic transmission system carries Terabits of data per second to achieve faster data transmission and avoids congestion. But, this transmission system is quite vulnerable to numerous security risks from passive optical access networks, to trans-Pacific submarine cable [1-2]. For example, these cables are capable to access readily and can be simply tapped by employing tapping devices, for instance, transitory coupling devices and bending the fiber cable [3]. Eavesdropping attacks are undetected for an extended period, where these cables are suspiciously positioned with the transmission link. Nowadays, recent studies focus over information security [25-28] which is a growing threat in fiber optic network. This issue has to be weeded out soon because the optical transmission system is a general medium of transmitting data among public which it provides increased efficacy and huge accessibility. Many efforts are being undertaken to encrypt and to secure transferring of information. Fiber optic networks are extensively employed to transfer different kinds of information comprising of secret and private, hence provides huge level of security for every layers. Including the new approaches such as encryption and compression, there are six types of optical security solution like quantum imaging, phase retrieval algorithms to perform different attacks, ghost imaging, nano and micro-scale implementation, theory of a rigorous optical information security. Out of the above methods, compression-based encryption is the best solution due it security reasons and compression benefits [4, 5]. By encrypting the actual information into cipher text, the encryption preserves the data transfer [29-35]. The eavesdropper is not capable to retrieve the information, without knowing the key for encryption. In study [6-10], the optical encryption is extensively studied.

To diminish the bit count needed to communicate a picture or audio or video, data compression methods are employed. Data compression is the science or art of presenting data in a compact form. Through recognizing and employing patterns present in information, the compact representation of data leads to compression. The data might be numbers, pixels of picture, waveforms or speech. The cause behind the requirement of data compression is that a massive amount of data that we produce everyday and employing it in a digital manner – comprising of numbers depicted by bytes of information. Additionally, the number of bytes needed to depict the multimedia information is comparatively huge. By employing low bandwidth at a faster rate, data compression algorithm offers broadcasting massive data with the utilization of fewer resources. Moreover, even the information is compromised, in addition to encrypting methods; the attacker requires knowing the characteristic of compression method.

So, this paper presents a new dictionary based encoding technique called Bounded probability based textual data compression algorithm called BPT algorithm. The BPT algorithm generates a codeword based on the dictionary, which contains the binary code based on the probability of occurrence of characters in the input data. For decompression, there is a need to transmit the coding table along with the compressed data. The proposed BPT algorithm is tested using a set of benchmark textual dataset from The Calgary Corpus and The Canterbury Corpus. The experimental results verified the superiority of the BPT algorithm over the state of art methods in terms of different measures namely compression ratio (CR), compression factor (CF), bits per character (bpc) and space savings.

2. Related works

The conventional coding methods that are assumed as the predecessors of data compression with the recently projected text compression methods are reviewed in this section. In (Patel, Angela, Dangarwala, & Choudhary, 2015)(Rani & Singh, 2016), many attempt are done to develop efficient lossless text compression methods.

2.1. Traditional methods

The conventional lossless text compression methods like Arithmetic coding, Run Length Encoding (RLE), Huffman coding, different versions of Lempel Ziv (LZ) coding, Burrows Wheeler Transform (BWT) coding are focused in this section. RLE [6] is a variable to block encoding method that works as below: When an input character d occurs n successive times in the input sequence, then n occurrences of the input series are replaced through a pair nd. The n successive occurrences of the input characters are denoted as run length of n. When there is low redundancy in the input information, this technique fails to compress the information efficiently. Huffman coding [7] is a variable length coding method, assign shorter codewords for frequent appearance of characters and longer code words for infrequent appearance of characters in. Let $P(i)$ be the probability of the i^{th} message, then $\sum_{i=1}^{N} P(i) = 1$, where N denotes the count of characters in the input series. The average length of the message L_{av} and is described as $L_{av} = \sum_{i=1}^{N} P(i)L(i)$, where $L(i)$ is the number of coding digits assigned to it. Here, the decoding procedure is very difficult and it is to recognize the final bit of the coded character even though it is simple to implement. To decode it, it is required to send the Huffman table with the compressed file. By encoding the information to unique symbols instead of codewords, the arithmetic coding [8] avoids the disadvantage of Huffman coding. Within the interval [0, 1], it encodes a message through real numbers. And it contains some disadvantages, after the encoded information is fully arrived, the decoder can initiate the decoding procedure. The whole file get corrupts if there is one false bit within the encoded information.
When the actual information needs to be compressed includes frequently repeating patterns, the dictionary based coding methods are helpful. When a pattern arrives in the input series, they get coded along with the index to dictionary. While the pattern is not available in dictionary, it is independently coded with some other compression method and append to the dictionary with an index. There are two kinds of dictionaries that are available: one is static and other one is adaptive. When the actual data information is unidentified, adaptive dictionary is employed. In lossless file compression, generally employed dictionary based coding method is Lempel-Ziv algorithm (LZ). It is adaptable to variety of file formats and it is used extensively. This approach is less efficient for small files and more efficient for large files. The length of the dictionary would be large when compared to actual file size in smaller files. The two major versions of LZ were built by Lempel and Zip named as [9] and LZ77 [10]. Mainly these methods differ by means of finding and searching matches. From the current position, the LZ77 algorithm generally employs a sliding window concept and searches for matches in a window in a predetermined distance. LZ78 algorithm appends strings to dictionary and it follows a conservative method. LZW coding [11] build a dictionary of frequently appearing patterns. With a reference to dictionary, the LZW substitute the detected patterns. The number of bits required for references also increases when the size of the dictionary is increased. To start the decoding procedure, the LZ coding requires transmitting the dictionary from encoder to decoder; hence it is main disadvantage of LZ coding. At the end, Burrows-Wheeler Transform (BWT) [12] is called as block sorting compression method that rearrange the character string into runs of identical characters. It employs two methods to compress the information named as RLE and move-to-front transform. It easily compresses the information and it is helpful in cases where the string comprises of the runs of recurring characters. The significant characteristics of BWT that nature of reversibility and does not require any additional bits.

2.2. Recently proposed text compression algorithms

To change the conventional coding methodologies like arithmetic coding and Huffman coding, [13] projected a neural network (NN) based compression method. The projected technique depends on the procedure of "predictive or model based coding". It is used to compress few newspaper articles and achieves enhanced CR when compared to LZ method. However, when compared to the conventional techniques, it shows slow performance. The two issues present in the compression of dynamic databases are addressed by [14]. The issues are managing the document insertions and memory need at the decoding side. Compression with ant dictionaries (DCA) algorithm is built by [15] which uses "negative information" about the text by means of ant dictionaries. The advantages of these methods are faster compression rate for predefined sources, synchronization property results to effective parallel compression and faster decompression rate. To improve the efficacy of the given compression methods, few pre-processing methods are applied before compression. A pre-processing method was projected by [16] for text compression. It needs no external dictionaries, because it is language independent. It works under algorithm that operates in a series manner named as word replacement, capital letter conversion, phrase replacement, end of line (EOL) coding, recording of alphabets. But, the pre-processing cost seems to be high. A new compression method is presented in [17] for small text files. It is modeled specially for very tiny text minimization. Other compression method known as b64 pack for short messages is projected in [18]. It works under two steps and it is a lightweight and efficient method. In the initial step, it transfers the actual text to a compressed format. In the subsequent step, it includes the transformation which reduces the message size through a built-in fraction of the actual size. The pre-processing algorithm projected in [19] for BWT coding, pre-processing algorithm for Huffman coding, versions of LZ coding, arithmetic coding. With a reduced file size, a genetic reversible transformation is commenced to transfer a text file to another format. For static text compression through the relaxation of prefix property of the dictionary, [20] commenced a greedy method for static text compression. To achieve increased speed in distributed systems, a finite state machine (FSM) implementation of greedy based compression with an arbitrary dictionary is projected. To compress the sentiment sentences in aspect based sentiment analysis, a novel compression method known as sent comp is projected in [21]. In the domain of text compression, [22] uses data mining tools. By combining the frequent itemset mining (FIM), Huffman coding is enhanced. For frequently occurring patterns, shorter codewords has been assigned. A graph based technique is used for searching series of characters employed in compression procedure presented in [23]. In one pass of the graph, this scheme develops a graph in one pass of the text and mines each and every pattern that is significant for compression. For textual data, [24] projected a new FIM based Huffman coding technique employing hash table (FPH2). Character based method is employed in the conventional Huffman coding methods whereas optimized (pruned) set patterns are implicated in the coding process.

3. The proposed algorithm

The proposed BPT algorithm is a dictionary based coding technique which encodes the input character using a bounded code words, which are derived based on the probability of occurrences. Once all the characters are read by the proposed algorithm, the algorithm first determines the number of occurrences of every input character in the text file. Then, a codeword with shortest length is assigned to the frequently occurred character, and longer length codeword is allocated to the least frequently appeared char-

acter. Thus, variable length code words are allocated to every character in the input data. Using this process, the number of bits needed to save the most frequently appeared characters can be easily reduced. At the same time, longest codeword for least frequently occurring characters does not increase the storage area. The block diagram of the BPT algorithm is shown in Fig. 1.

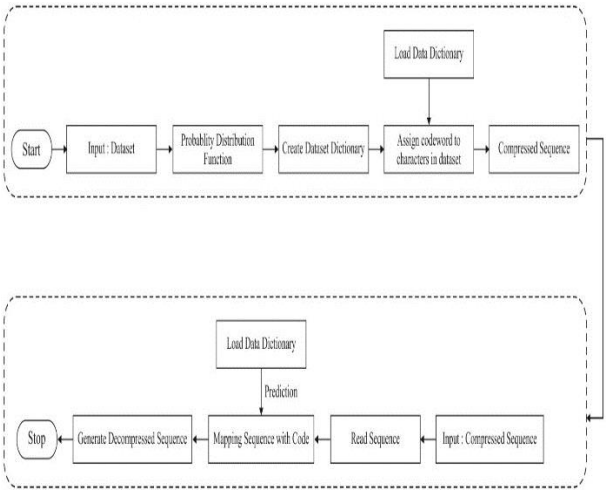

Fig.1: Block Diagram of Proposed Method.

Table 1: Coding Table of the Proposed Method

S. No	Pattern	No. of bits needed by BPT algorithm	No. of characters stored
1	0	1	2
2	1	1	
3	00	2	
4	01	2	4
5	10	2	
6	11	2	
7	000	3	
8	001	3	
9	010	3	
10	011	3	8
11	100	3	
12	101	3	
13	110	3	
14	111	3	
15	0000	4	
16	0001	4	
17	0010	4	
18	0011	4	
19	0100	4	
20	0101	4	16
21	0110	4	
22	0111	4	
23	1000	4	
24	1001	4	
25	1010	4	
…	…		…
…	…		…
254	1111111	7	128

The usual occurrences of any input character in the text file can be of 26 alphabets (a-z), numerals (0-9), and some special characters (. ? /:.,). For instance, to store a combination of 63 characters in text file, it needs minimum of 1 bit and a maximum of 5 bits. In the worst case, the BPT achieves the bit rate of 5 whereas the traditional methods require 8 bits for the same. The proposed algorithm, to store two input characters, the projected algorithm taken only one bit; the codeword will be 0 and 1 for the any two frequently appeared characters which marks that there is need of one bit alone to store it. Likewise, the code words 00, 01, 10 and 11 will be generated for any 4 input characters which imply that it requires only 2 bits to store 4 characters. Similarly, the proposed algorithm generates the code of 1111111 for the 254 token codes. The BPT follows symmetrical compression where the encoding and decoding process is the exact opposite of each other. Since the coding table is generated based on the probability of occurrences in the text, there is a need to transmit the coding table along with the compressed data.

Once the compressed data and coding tables are received at the decoder, the BPT algorithm performs the reverse process of compression algorithm. First, the codeword will be read from the compressed data and the corresponding data will be retrieved by performing a lookup in the dictionary. The process will be repeated until all the compressed data are reconstructed to produce the original data without any loss of information.

4. Performance evaluation

Various experiments are done using the Windows 10 operating system running on a general-purpose PC with 8GB of RAM and an Intel i7 core running at 2.70GHz to measure the different aspect of text compression with BPT algorithm. The dataset applied, performance metrics and gained outcomes are given in the subsequent subsections.

4.1. Dataset description

For implementation, the textual dataset which is available in public are used.

Table 2: Dataset Description

File	Size (in bytes)	Description	Type
bib	111 261	A bibliography in UNIX refer format	Text
paper1	53 161	A technical paper in troffformat	Text
paper2	82 199	A technical paper in troffformat	Text
progc	39 611	A source program in C	C code
trans	93 695	Document teaching how to use a terminal	Text
alice29	152 089	English text (Alice in Wonderland)	Text
fields	11 150	C source code	C code
lcet10	426 764	Technical document	Text
asyoulik	125 179	English play (As You Like It)	Text

The two well-known standard dataset gathered in the year 1987 and 1997 are The Calgary Corpus and The Canterbury Corpus ("Dataset: The Calgary Corpus and The Canterbury Corpus," 1987). Initially, 'The Calgary Corpus' dataset comprises of 18 files with various kinds of information including images, object files and texts are employed. We employed C source code file for simulation and text files. The dataset size ranges from 39,611 bytes to 152,089 bytes. To measure the efficiency of the projected BPT method, we uses a sum of 9 files (7 text files, 2 C source code files). The standard dataset's descriptions are provided in the Table 2.

4.2. Results and discussion

There are 9 datasets taken for evaluation of compression performance with the projected method BPT and the other existing methods as described in table 2. The used dataset is bib, paper 1, paper 2, progc, trans, alice29, fields, lcet10 and asyoulik. Among all the datasets, bib, alice29 and asyoulik are larger dataset, progc and trans are programming source code, paper 1 and paper 2 are usual text datasets. Hence, among all datasets, it is difficult to compress larger size dataset.

The compression performance is measured to estimate the performance of the projected method by means of six metrics such as Compression Ratio (CR), Compression Factor (CF), Compression Time (CT), and bits per character (bpc), Space savings and Packet size. The CR is defined as,

$$\mathrm{CR} = \frac{\text{No.of bits needed to represent uncompressed file}}{\text{No.of bits needed to } represent \text{ compressed file}} \quad (1)$$

Table 3: Comparison of Compression Performance with Various Methods

Dataset	Compression Ratio (CR)				Compression Factor (CF)				bits per character (bpc)				Compression Time (CT)			
	Proposed	b64pack	Huffman	Arithmetic	Proposed	b64pack	Huffman	Arithmetic	Proposed	b64pack	Huffman	Arithmetic	Proposed	b64pack	Huffman	Arithmetic
bib	0.31	0.51	0.65	0.65	3.18	1.90	1.52	1.53	2.51	4.84	5.23	5.22	122	326	340	124
paper1	0.33	0.46	0.62	0.62	3.04	2.14	1.59	1.59	2.63	4.54	5.01	5.01	35	109	110	47
paper2	0.3	0.45	0.58	0.57	3.29	2.34	1.73	1.72	2.43	4.61	4.63	4.62	64	165	95	76
progc	0.34	0.53	0.65	0.65	2.92	1.71	1.53	1.52	2.74	5.04	5.23	5.24	31	91	65	34
trans	0.29	0.41	0.68	0.67	3.42	2.46	1.47	1.47	2.34	4.43	5.44	5.43	75	123	140	80
alice29	0.28	0.47	0.58	0.57	3.62	2.12	1.73	1.74	2.21	4.58	4.61	4.58	123	298	170	135
fields	0.33	0.40	0.63	0.64	3.02	2.50	1.59	1.55	2.65	3.60	5.04	5.13	10	26	29	12
lcet10	0.27	0.47	0.71	0.58	3.67	2.12	1.40	1.71	2.18	4.67	5.69	4.67	246	584	600	362.02
asyoulik	0.31	0.46	0.60	0.60	3.2	2.60	1.65	1.65	2.5	4.24	4.84	4.82	98	266	240	105

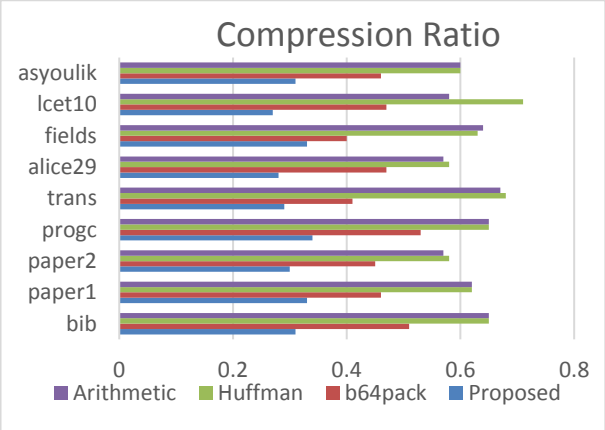

Fig. 2: Performance Measure in Terms of Compression Ratio.

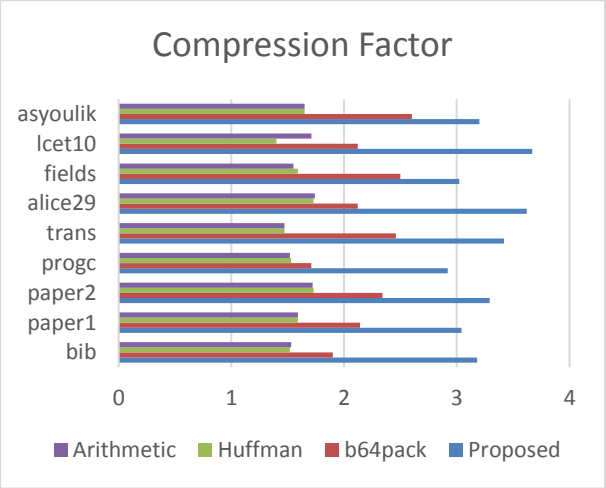

Fig. 3: Performance Measure in Terms of Compression Factor.

Table 3 shows the comparative results of the proposed and compared methods interms of CR, CF, bpc and CT. Fig. 2 shows the compression performance interms of CR. The rate of CR greater than 1 concludes to negative rate of compression in which the volume of the compressed data is larger than the actual volume of the given dataset. Arithmetic coding and Huffman produce the CR value of 0.65 for the massive textual dataset bib which denotes that the data still remains uncompressed of 65% of its actual volume. For all the datasets, it may be usual dataset like paper 1 or paper 2, Huffman coding produces the CR value of 0.58 and 0.71 which denotes the compression performance is not efficient because it lefts 30 to 50 percentage of the data size equal to the size before compression. When comparing with the recent existing method b64 pack, the compression performance ranges from 0.41 to 0.53 which exhibits it compresses 40 to 50 percent of the actual dataset.

It exhibits better outcomes when compared to the Arithmetic coding and Huffman techniques but it fails to outperform the projected BPT algorithm where BPT produces the CR of 0.31 for the massive dataset bib. The BPT shows better CR than other methods on all the applied dataset.

The inverse of CR is termed as CF, where the rate of CF larger than 1 denotes better compression and the lesser CR denotes data expansion.

$$\mathrm{CF} = \frac{\text{No.of bits needed to represent compressed file}}{\text{No.of bits needed to } \textit{represent} \text{ uncompressed file}} \qquad (2)$$

The CF rate somewhat enhances by the b64 pack algorithm which ranges from 1.71 to 2.60 for all the dataset. Specifically, for bib dataset, it shows the CR of 1.90 and it manages provide comparable results to the above methods. The projected BPT exhibits superior compression performance where the higher CF value varies from 2.92 to 3.67 and it attains maximum CF rate of 3.67 for the massive textual dataset named lcet10. Therefore, the performance outcomes shows that the projected BPT algorithm shows comparatively enhanced results over the existing methods.

The significant metric, bits per character (bpc) is employed to measure the text compression algorithm performance. To compress one character, reside in the input series, the bpc is employed to estimate the total number of bits required, on average and is expressed as,

$$\mathrm{BPC} = \frac{\text{Total no.of bits required to represent a compressed file}}{\text{Total number of characters}} \qquad (3)$$

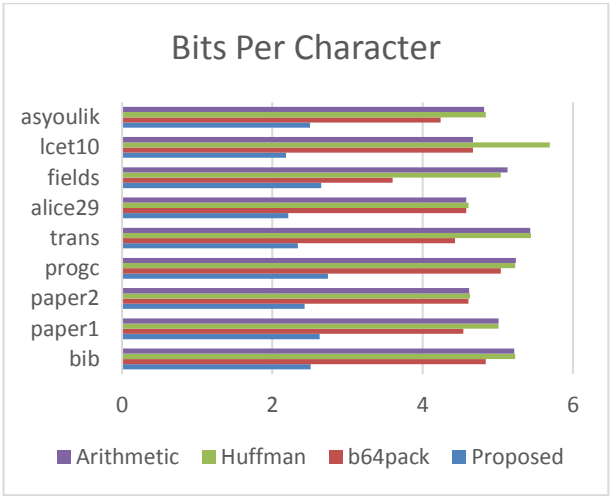

Fig. 4: Performance Measure in Terms of Bpc.

Fig. 4 shows the compression performance in terms of CR. Arithmetic coding and Huffman produces the bpc value of 5.23 and 5.22 respectively for the massive textual dataset bib which denotes that the bpc is slightly reduced compared with Arithmetic coding. For all the datasets, like paper 1 or paper 2, the Huffman coding produces the bpc value from 4.58 to 5.69 which denotes the compression performance for bpc is not efficient because it generates more number of bits per character which results in elongated time for transmission. When comparing with the recent existing method b64 pack, bpc ranges from 3.60 to 5.04 which exhibits that it produces large number of bits, but it manages to outperform the above existing methods. It exhibits better outcomes when compared to the Arithmetic coding and Huffman techniques, but it fails to outperform the projected BPT algorithm where it produces 2.51 bit per character even for the massive dataset bib and comparatively the projected method shows superior performance in terms of bpc.

To create a compressed data from the actual data, the amount of time it takes to compress is called as Compression time (CT). For a better algorithm, the compression time should be less when compared to the other algorithms.

Dataset	Space savings (SS)				Packet size (PS)			
	Proposed	b64pack	Huffman	Arithmetic	Proposed	b64pack	Huffman	Arithmetic
bib	68.62	49.00	35.00	35.00	1203.551724	1956.659	2493.781	2493.781
paper1	67.08	54.00	38.00	38.00	603.5172414	843.2434	1136.546	1136.546
paper2	69.61	55.00	42.00	43.00	861.3103448	1275.502	1643.98	1615.636
progc	65.72	47.00	35.00	35.00	468.2413793	723.9252	887.8328	887.8328
trans	70.77	59.00	32.00	33.00	944.4827586	1324.653	2196.986	2164.678
alice29	72.38	53.00	42.00	43.00	1448.413793	2464.891	3041.78	2989.336
fields	66.88	60.00	37.00	36.00	127.3448276	153.7931	242.2241	246.069
lcet10	72.77	53.00	29.00	42.00	4007.551724	6916.52	10448.36	8535.28
asyoulik	68.77	54.00	40.00	40.00	1348.068966	1985.598	2589.91	2589.91

Fig. 5 shows the comparison of different algorithms in terms of CT. Arithmetic coding and Huffman produces the CT value of 124s and 340s respectively for the massive textual dataset bib which denotes that it consumes more time for compression.

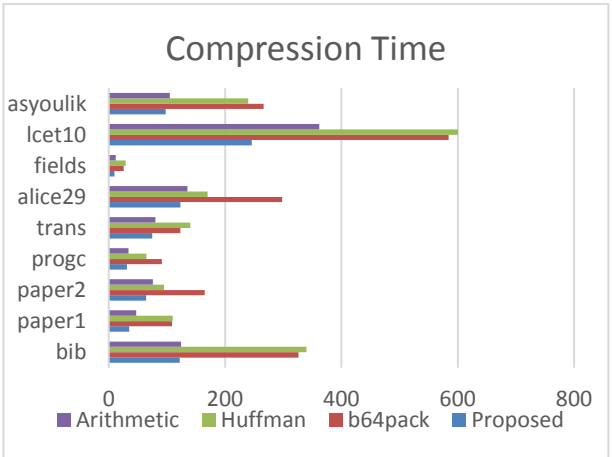

Fig. 5: Performance Measure in Terms of Compression Time (CT).

For all the datasets, if it might be a usual dataset like paper 1 or paper 2, the Arithmetic and Huffman coding produces the CT value ranges between 12s to 600s which denotes the compression performance for CT is not efficient because it need more time even for small datasets. When comparing with the recent existing method b4 pack, it achieves less CT time and manages to outperform the above existing methods except the BPT algorithm. The BPT algorithm executes faster than the compared algorithms. Table 4 shows the comparison of compression performance of different methods in terms of space savings (SS) and Packet size (PS) for the given dataset. Another measure to validate a compression technique is Space savings, which indicates the reduction in file size relative to the uncompressed size and is equated in Eq. (4).

$$\text{Space savings} = 100 \times \left(1 - \left(\frac{\text{No.of bits in compressed data}}{\text{No.of bits in uncompressed data}}\right)\right) \quad (4)$$

The space savings refers to the space that is saved after compression of the file before compression. If the text file size is small, it is easy to transfer the data through fiber optic network.

The Space savings (SS) of Huffman and arithmetic coding is almost same and it attains the SS of 29.0% to 43.00% for all the datasets which exhibits that these methods shows poor compression performance. The amount of SS is somewhat enhanced with the use b64 pack algorithm which ranges from 46.00% to 60.00% for all the dataset. Specifically, for bib dataset it shows 49.00% of SS and it manages to be efficient when compared to the above methods.

The projected BPT exhibits better results with the SS rate ranges from 65.72% to 72.77% and it attains maximum SS of 72.77% for the massive textual dataset named lcet10. Therefore, the performance outcomes shows that the projected BPT algorithm shows comparatively enhanced results over the existing methods.

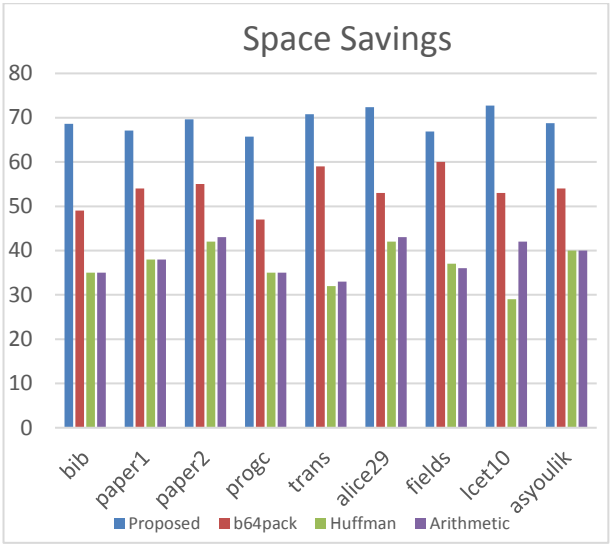

Fig. 6: Comparison of Different Methods in Terms of Space Saving.

Packet size is the other metric to evaluate the compression performance. Packet size should be generated in a reduced amount and is equated in Eq. (5).

$$\text{Packet size} = \frac{\text{No.ofbits required to transmit compressed data}}{\text{payload (232 bits)}} \quad (5)$$

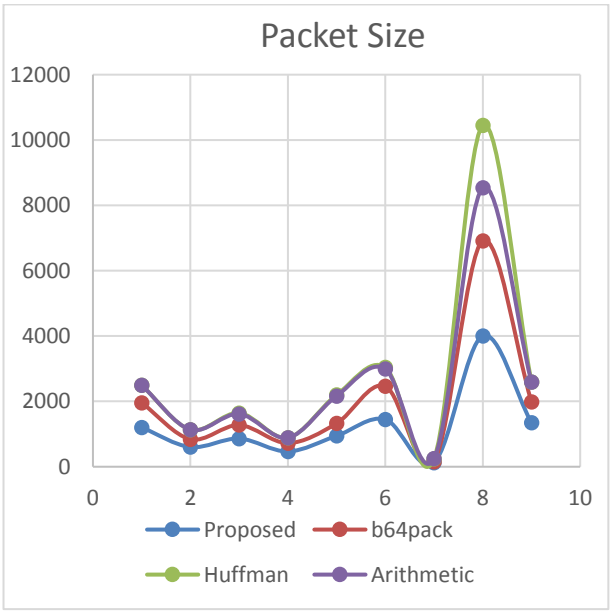

Fig. 7: Comparison of Different Methods in Terms of Packet Size.

Fig. 7 shows the comparative compression performance of different methods interms of PS. During compression, Arithmetic coding and Huffman produces the average packets of 46.069 and 10448.36 respectively. For the massive textual dataset bib, these algorithms require more number of packets to store compressed data. The arithmetic and Huffman coding is ineffective since it needs more number of packets even for small datasets. When comparing with the recent existing method b64, the number of packets ranges between 153.7931 to 2464.891 which produces less number of packets but it manages to outperform the above existing methods. It exhibits better outcomes when compared to the Arithmetic coding and Huffman techniques but it fails to outperform the projected BPT algorithm where it requires the packets of 1956.659 even for the massive dataset bib and comparatively the projected method shows superior performance over PS

We compared the performance of the proposed BTC algorithm with some well-known algorithms for compression algorithms such as Huffman coding, Arithmetic coding and the recently proposed existing method b64pack. By employing the similar set of 9 dataset, the comparison has been made directly with the outcomes of the traditional methods. It reveals few motivating facts that the method works tremendously different depending on the nature of the given dataset during compression over 9 varieties of datasets. The summarization of performance of proposed BTC algorithm with the conventional and recent existing method by means of CR, CF, bpc and CT, Space savings (SS), Packet size (PS). Fig. 1-6 depicted that the compression performance with the proposed and conventional algorithms. Although b64pack algorithm strives to give good outcomes for some dataset, it fails to attain good compression than BTC algorithm. In terms of CR, the b64 is greater than Huffman and arithmetic coding for almost for all the given dataset. However, for the massively given textual dataset 'bib' it fails to achieve good compression. Similarly, Table 2 demonstrates the compression performance with the proposed and conventional algorithms in terms of Space savings (SS) and Packet size (PS). The Arithmetic coding and Huffman shows more or less the same compression performance with respect to SS. The BTC algorithm exhibit superior compression performance even for the massive dataset and programming codes when compared to the recently existing algorithms b64pack. The compression performance in terms of packet size (PS) is depicted in the same table, where the Arithmetic coding and Huffman produces a large number of packet size comparatively b64pack algorithm generates reduced number of packets. Above all, BTC algorithm shows decreased PS rate, therefore it is the superior out of all the traditional methods.

Therefore, some large textual dataset comprises of repeated characters and patterns, the b64pack algorithms fails to perform better in these datasets. However, even for dataset with less repeating characters and patterns, it generates less compression rate. It is noted that the arithmetic coding and Huffman exhibits inefficient compression outcome with larger text file datasets. At the end, the BTC algorithm achieves better compression performance for every dataset. Hence, the proposed BPT algorithm is a dictionary based coding technique which encodes the input character using a bounded code words derived based on the probability of occurrences.

5. Conclusion

This paper has presented a new dictionary based encoding technique called BPT algorithm for textual compression algorithm in fiber optic communication. The BPT algorithm generates a codeword based on the dictionary, which contains the binary code based on the probability of occurrence of characters in the input data. For decompression, there is a need to transmit the coding table along with the compressed data. For implementation, the textual dataset The Calgary Corpus and The Canterbury Corpus are used. The experimental results verified the superiority of the BPT algorithm over the state of art methods in terms of different measures namely CT, CR, CF, bpc, SS and PS.

References

[1] Medard, Muriel, Douglas Marquis, A. Richard Barry, and G. Steven Finn. "Security issues in all-optical networks." *IEEE network 11, no. 3(1997)*: 42-48. https://doi.org/10.1109/65.587049.

[2] Skorin-Kapov, Nina, MarijaFurdek, Szilard Zsigmond, and Lena Wosinska. "Physical-layer security in evolving optical networks." *IEEE Communications Magazine*, Vol. 54, No. 8, (2016) :110-117. https://doi.org/10.1109/MCOM.2016.7537185.

[3] Shaneman, Keith, and Stuart Gray. "Optical network security: technical analysis of fiber tapping mechanisms and methods for detection & prevention." In *Military Communications Conference*, MILCOM 2004. 2004 IEEE, Vol. 2, pp. 711-716, 2004.

[4] Alfalou, Ayman, C. Brosseau, Nadine Abdallah, and Maher Jridi, "Simultaneous fusion, compression, and encryption of multiple images." *Optics express*, Vol. 19, No. 24, pp. 24023-24029, 2011. https://doi.org/10.1364/OE.19.024023.

[5] Alfalou, Ayman, Christian Brosseau, N. Abdallah, and M. Jridi. "Assessing the performance of a method of simultaneous compres-

sion and encryption of multiple images and its resistance against various attacks." *Optics express*, *Vol. 21, No. 7, (2013)* pp. 8025-8043. https://doi.org/10.1364/OE.21.008025.

[6] J. Capon, "A probabilistic model for run-length coding of pictures". *IRE Transactions on Information Theory*, *Vol. 100*, 157–163. https://doi.org/10.1109/TIT.1959.1057512.

[7] D. A. Huffman, *A Method for the Construction of Minimum-Redundancu Codes*, *(1952):* 1098–1102. https://doi.org/10.1109/JRPROC.1952.273898.

[8] G. G. Langdon, "An Introduction to Arithmetic Coding. IBM Journal of Research and Development", Vol. 2, No.2, pp. 135–149, 1984. https://doi.org/10.1147/rd.282.0135.

[9] J.Ziv, and A.Lempel, 1978. lz78.pdf. IEEE.

[10] J.Ziv and A.Lempel. "A Universal Algorithm for Data Compression" *IEEE Transactions on Information Theory*, Vol. 23, No.3, (1977): 337–343. https://doi.org/10.1109/TIT.1977.1055714.

[11] T. A. Welch, "A technique for high-Performance Data Compression", *IEEE*, (1984): 8–19. https://doi.org/10.1109/MC.1984.1659158.

[12] M. Burrows and D. Wheeler, "A block-sorting lossless data compression algorithm". *Algorithm, Data Compression*, No. 124, (1994)18. https://doi.org/10.1.1.37.6774

[13] J. Schmidhuber and S. Heil, "Sequentail Neural Text Compression. IEEE Transactions on Neural Networks", *Vol.7, No.1, (1996):*142–146. https://doi.org/10.1109/72.478398.

[14] A. Moffat, J. Zobel and N.Sharman, "Text compression for dynamic document databases". *IEEE Transactions on Knowledge and Data Engineering*, Vol. 9, No.2, (1997) 302–313. https://doi.org/10.1109/69.591454.

[15] M. Crochemore, F. Mignosi, A.Restivo and S. Salemi, "Data compression using antidictionaries." In *Proceedings of the IEEE*, pp. 1756–1768, 2000. https://doi.org/10.1109/5.892711.

[16] J. Abel and W. Teahan, "Universal text preprocessing for data compression". *IEEE Transactions on Computers*, *Vol. 54, No.5, (2005)*:497–507. https://doi.org/10.1109/TC.2005.85.

[17] J. NoPlatoš, V. Snášel and E. El-Qawasmeh, "Compression of small text files". *Advanced Engineering Informatics*, *Vol. 22, No. 3, (2008)*, pp. 410–417. https://doi.org/10.1016/j.aei.2008.05.001.

[18] K. Kalajdzic, S. H. Ali and A. Patel, "Rapid lossless compression of short text messages". *Computer Standards & Interfaces*, *Vol. 37, (2015)*:53–59. https://doi.org/10.1016/j.csi.2014.05.005.

[19] L. Robert and R. Nadarajan, "Simple lossless preprocessing algorithms for text compression". *IET Software*, *Vol. 3, No.1, 2009*, pp. 37–45. https://doi.org/10.1049/iet-sen:20070106.

[20] S. De Agostino, "The greedy approach to dictionary-based static text compression on a distributed system", *Journal of Discrete Algorithms*, *Vol. 34, (2015):* 54–61. https://doi.org/10.1016/j.jda.2015.05.001.

[21] W. Che, Y. Zhao, H. Guo, Z. Su and T. Liu, "Sentence compression for aspect-based sentiment analysis". *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, *Vol. 23, No.12, (2015)*, 2111–2124. https://doi.org/10.1109/TASLP.2015.2443982.

[22] C. Oswald, A. I. Ghosh and B. Sivaselvan, "Knowledge engineering perspective of text compression". In *India Conference (INDICON)*, pp. 1–6, 2015. https://doi.org/10.1109/INDICON.2015.7443683.

[23] C. Oswald, I. A. K., J. Avinash and B. Sivaselvan, "A Graph-Based Frequent Sequence Mining Approach to Text Compression". In *International Conference on Mining Intelligence and Knowledge Exploration*, pp. 371–380. https://doi.org/10.1007/978-3-319-71928-3_35.

[24] C. Oswald and B. Sivaselvan, "An optimal text compression algorithm based on frequent pattern mining". *J Ambient Intell Human Comput*, 1–20.

[25] Mohamed Elhoseny, K. Shankar, S. K. Lakshmanaprabu, Andino Maseleno, N. Arunkumar. Hybrid optimization with cryptography encryption for medical image security in Internet of Things. *Neural Computing and Applications*. 2018. https://doi.org/10.1007/s00521-018-3801-x.

[26] K. Shankar, Mohamed Elhoseny, E. Dhiravida chelvi, SK. Lakshmanaprabu, Wanqing Wu. An Efficient Optimal Key Based Chaos Function for Medical Image Security. *IEEE Access*. 2018. https://doi.org/10.1109/ACCESS.2018.2874026.

[27] T. Avudaiappan, R. Balasubramanian, S. Sundara Pandiyan, M. Saravanan, S. K. Lakshmanaprabu, K. Shankar, "Medical Image Security Using Dual Encryption with Oppositional Based Optimization Algorithm", *Journal of Medical Systems*, 42.11 (2018) 1-11. https://doi.org/10.1007/s10916-018-1053-z.

[28] K.Shankar and P.Eswaran. "RGB Based Multiple Share Creation in Visual Cryptography with Aid of Elliptic Curve Cryptography", *China Communications*, 14. 2 (2017): 118-130. https://doi.org/10.1109/CC.2017.7868160.

[29] K.Shankar and P.Eswaran. "RGB Based Secure Share Creation in Visual Cryptography Using Optimal Elliptic Curve Cryptography Technique", *Journal of Circuits, Systems, and Computers*, 25.11 (2016) : 1650138-1 to 23. https://doi.org/10.1142/S0218126616501383.

[30] Nur Aminudin, Andino Maseleno, K. Shankar, S. Hemalatha, K. Sathesh kumar, Fauzi, Rita Irviani, Muhamad Muslihudin, "Nur Algorithm on Data Encryption and Decryption", *International Journal of Engineering & Technology*, 7. 2.26 (2018): 109-118. https://doi.org/10.14419/ijet.v7i2.27.11574.

[31] K. Shankar, Lakshmanaprabu S. K, "Optimal key based homomorphic encryption for color image security aid of ant lion optimization algorithm", *International Journal of Engineering & Technology*, 7. 9 (2018) : 22-27. https://doi.org/10.14419/ijet.v7i1.9.9729.

[32] K. Sathesh Kumar, K. Shankar, M. Ilayaraja, M. Rajesh, "Sensitive Data Security in Cloud Computing Aid of Different Encryption Techniques", *Journal of Advanced Research in Dynamical and Control Systems*, 9. 18 (2017): 2888-2899.

[33] K. Shankar and P.Eswaran. "An Efficient Image Encryption Technique Based on Optimized Key Generation in ECC Using Genetic Algorithm", *Advances in Intelligent Systems and Computing*, Springer, 394 (2016): 705-714. https://doi.org/10.1007/978-81-322-2656-7_64.

[34] K. Shankar and P.Eswaran. "A Secure Visual Secret Share (VSS) Creation Scheme in Visual Cryptography using Elliptic Curve Cryptography with Optimization Technique". *Australian Journal of Basic and Applied Sciences*. 9. 36 (2015): 150-163.

[35] K. Shankar and P.Eswaran. "ECC Based Image Encryption Scheme with aid of Optimization Technique using Differential Evolution Algorithm", *International Journal of Applied Engineering Research*, 10. 5 (2015): 1841–184.