

LOSSLESS TEXT DATA COMPRESSION USING MODIFIED HUFFMAN CODING

Harsimran Kaur¹, Balkrishan Jindal²

¹Research Scholar, Assistant Professor², YCoE, GKC, Punjabi University, Talwandi Sabo

ABSTRACT

In this paper, a new method for data compression is proposed. Data Compression is a technique to increase the storage capacity by eliminating redundancies that occur in text files. It converts a string of characters into a new string which have the same data in small length. In the proposed method, dynamic Bit Reduction algorithm and Huffman Coding is used to achieve better compression ratio and saving percentage as compared to the existing method. The accuracy of the proposed method is 60-70%, while the accuracy of existing method is 40-50%. The results of this study is quite promise.

Keywords: Data Compression, Data Compression Techniques, Lossless Text Data Compression.

I. INTRODUCTION

Data compression is a process by which a file (Text, Audio and Video) may be transformed into another file, such that the original file may be fully recovered from the original file without any loss of actual information [1]. This process may be useful if one wants to save the storage space. Fig 1.1 shows the compression and decompression process over the network.

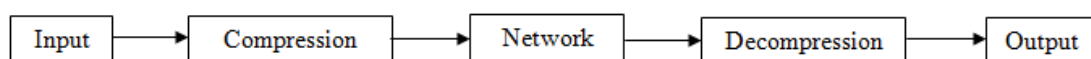


Fig. 1.1 compression and decompression process [2]

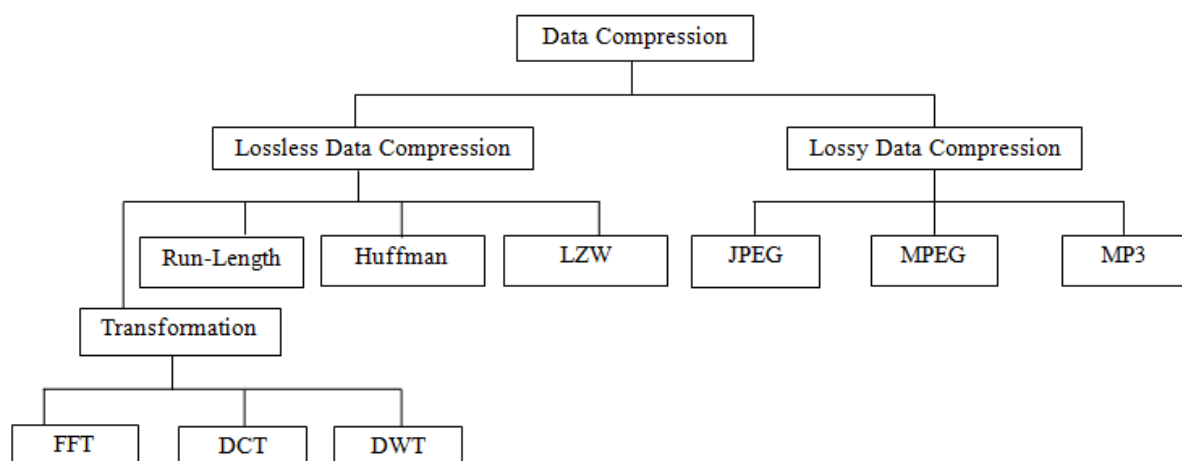


Fig: 1.2 classification of data compression [1]

Fig 1.2 shows the compression methods, lossless data compression and lossy data compression. The lossless data compression that is commonly used to transmit or archive text or binary files required to keep their

information intact at any time. Lossy data compression, which is widely used to compress image data files for communication or archives purposes. Lossless data compression is further divided into Run-Length, Huffman, Lempel Ziv algorithm and Transformation. Transformation is further divided into Fast Fourier Transform (FFT), Discrete Cosine Transform (DCT) and Discrete Wavelet Transform (DWT). Lossy data compression is divided into Joint Photographic Experts Group (JPEG), Moving Picture Experts Group (MPEG) and MPEG Audio Layer 3 (MP3).

II. RELATED WORK

Brar and Singh [1] described different basic lossless and lossy data compression techniques. A bit reduction algorithm for compression of text data had been proposed by them. That was a simple compression and decompression technique free from time complexity. Sharma et al. [4] described the Improved Dynamic Bit Reduction Algorithm to compress and decompress the text data. They presented the results obtained by the proposed system and compare the results with the existing data compression techniques.

Sidhu and Garg [3] described the improved dynamic bit reduction algorithm to compress and decompress the text data based on lossless data compression approach. Various experiments had been conducted on different datasets such as Random, Alphanumeric, Numeral and Special Characters dataset. Sharma [4] had analyzed Huffman algorithm and compared it with other common compression techniques like Arithmetic, LZW and Run Length Encoding. They concluded that arithmetic coding was very efficient for bits and reduced the file size dramatically.

Porwal et al. [5] described the lossless data compression methodologies and compares their performance. Huffman and arithmetic coding were compared according to their performances. The author had found that the arithmetic encoding methodology was powerful as compared to Huffman encoding methodology. Shanmugasundaram and Lourdusamy [6] described a survey of different lossless data compression algorithms. They used Statistical compression technique and Dictionary based compression technique performed on text data. Lossy algorithms achieve better compression than lossless algorithms, but lossy compression was limited to audio, images and video, where some loss was acceptable.

Yellamma and Challa [7] described the data compression was most consideration thing of the recent world. They had to compress a huge amount of data so as to carry from one place to other or in a storage format. The proposed compression technique were improved the efficiency of compression on text. Huffman encoding Algorithm was suitable for the given text. Aarti [8] had reviewed various techniques of lossless image compression with different file formats and the need of compression. All the lossless image compression techniques had been discussed by her, which gave better compression ratio.

Kodituwakku and Amarasinghe [9] described the comparison of a number of different lossless data compression algorithms by considering the Shannon Fano algorithm. Btoush et al. [10] described the different data compression algorithms of text files, LZW, Huffman, Fixed-length code and Huffman after using Fixed-length code. They compared these algorithms on different text files of different sizes in terms of compression scales of size, ratio, time and entropy.

Katugampola [11] described how ternary representation of numbers can be utilized to compress text data with fixed-symbol-length coding techniques. He used binary map for ternary digits. He found a way to minimize the

length of the bits string which was only possible in ternary representation that reduced the length of the code. Kapoor and Chopra [12] described the several key issues to the dictionary based LZW algorithm. They would like to improve LZW algorithm in future which definitely get good results like, better compression ratio, time taken for searching in the dictionary for pattern matching. They also discussed that the LZ algorithm had various issues regarding its performance and internal structure.

III. PROPOSED METHOD

The proposed improved Huffman Coding algorithm works in two phases to compress the text data specially. This method also improved the numeric data and symbolic data. In the first phase, data is compressed with the help of dynamic bit reduction technique and in second phase, Huffman coding is used to compress the data further to produce the final output. In the First phase, when user enters an input data, the method will find out the occurrence of number of unique symbols in the input text string and then numeric code will be assigned to these unique symbols. For each numeric code, corresponding binary codes will be generated dynamically to obtain the compressed binary output. The performance of the proposed method is measured using parameters such as Compression Ratio and Saving Percentage. Fig. 1.3 shows the data compression process of the proposed method step by step.

3.1 Compression Ratio

It is the ratio between the result of the compressed file and the result of the source file [9] [10].

$$\text{Compression Ratio} = (\text{After Compression} / \text{Before Compression}) * 100 \quad (1.1)$$

3.2 Saving Percentage

Saving Percentage calculates the shrinkage of the source file as a percentage [9].

$$\text{Saving Percentage} = \{(\text{Before Compression} - \text{After Compression}) / \text{Before Compression}\} * 100 \quad (1.2)$$

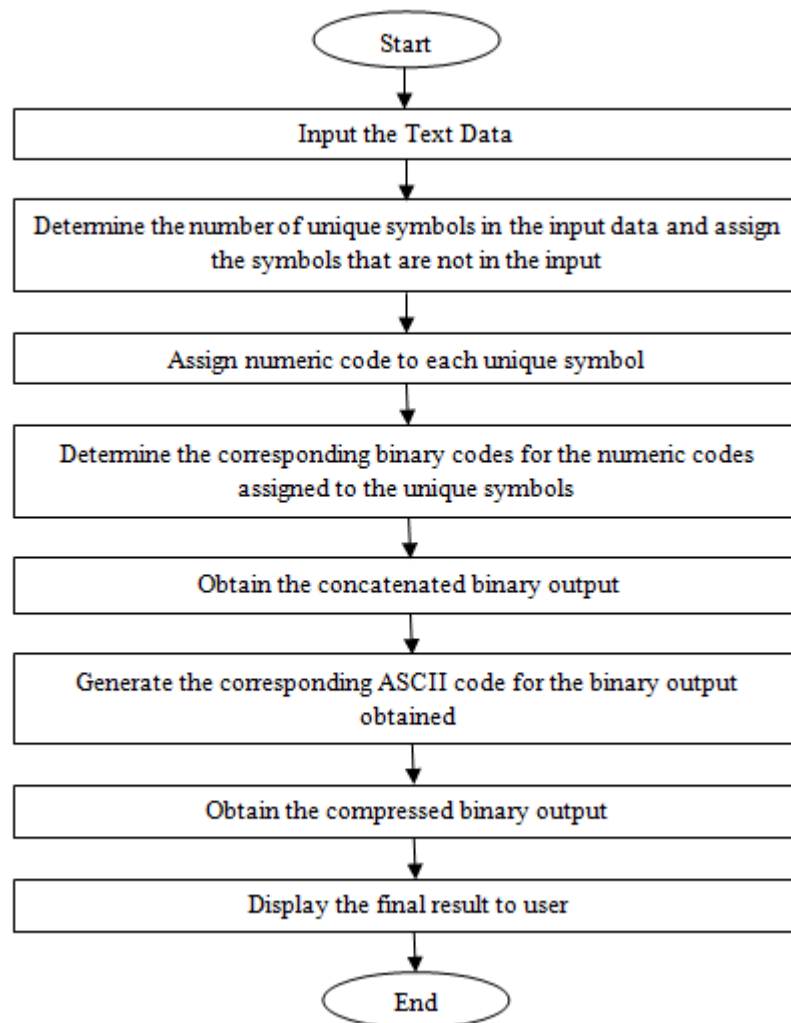


Fig. 1.3 data compression process of the proposed method

IV. PROPOSED DATA COMPRESSION ALGORITHM

Step I: Input the text data to be compressed.

Step II: Find the number of unique words in the input text data and assign the symbols that are not in the input.

Step III: Now find the unique characters from the step II.

Step IV: Find the number of bits required to assign bit code to the characters.

Step V: Assign the numeric code to the unique characters found in the step II according to the number of bits calculated in step IV.

Step VI: Starting from first symbol in the input find the binary code corresponding to that symbols from assigned numerical codes and concatenate them to obtain binary output.

Step VII: Add number of 0's in MSB of Binary output until it is divisible by 8.

Step VIII: Generate the ASCII code for every 8 bits for the binary output obtained in step VI and concatenate them to create input for second phase.

[Step VI is the result of dynamic bit Reduction method in ASCII format]

Step IX: Compressed data is obtained.

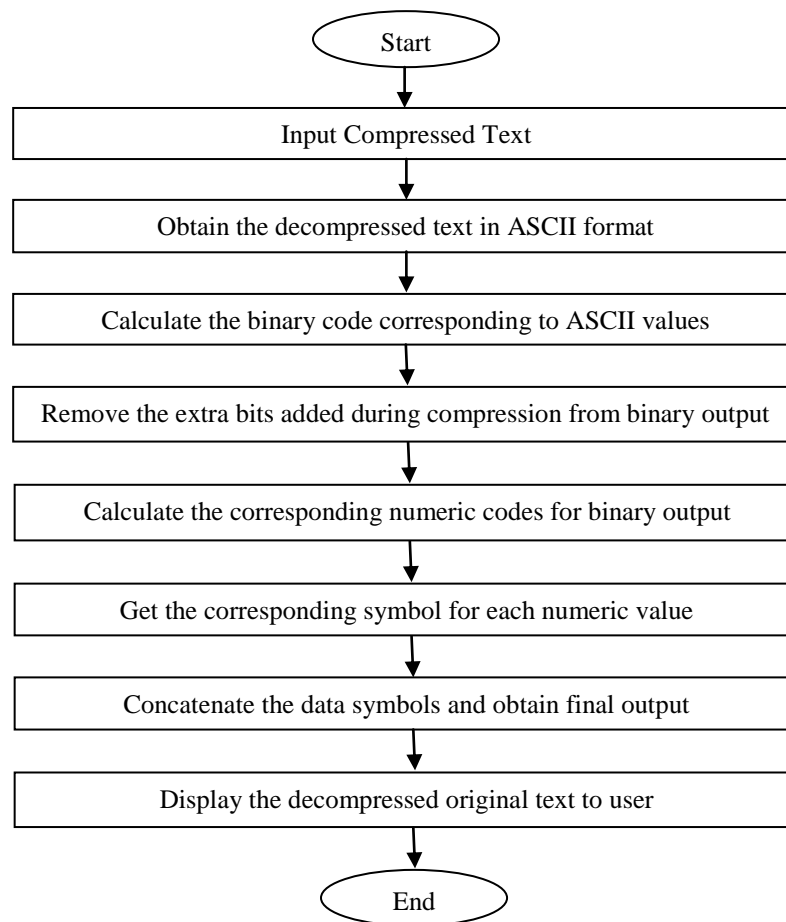


Fig. 1.4 data decompression process of the proposed method

V. PROPOSED DATA DECOMPRESSION ALGORITHM

Step I: Input the Final output from compressed phase.

Step II: Calculate the binary code corresponding to the ASCII values of input obtained in Step I.

Step III: Remove the extra bits from the binary output added in the compression phase.

Step IV: Calculate the numeric code for every 8 bits obtained in the Step IV.

Step V: For every numeric value obtained in the step V, find the corresponding symbol to get the final decompressed data.

Step VI: Concatenate the data symbols obtained in the step VI and obtain the final output.

Step VII: Display the final result to the user.

VI. ILLUSTRATION

Let the input string for compression: yadavindra_college

Then, find the unique characters and calculate the length to represent these characters as follows:

y=0, a=1, d=2, v=3, i=4, n=5, r=6, _=7, c=8, o=9, l=10, e=11 and g=12

After calculating the length, assign binary numbers to the input string as follows:

y=0000, a=0001, d=0010, v=0011, i=0010, n=0101, r=0110, _=0111, c=1000, o=1001, l=1010, e=1011 and g=1100

The output of the input string is as follows:

0000 0001 0010 0001 0011 0100 0101 0010 0110 0001 0111 1000 1001 1010 1010 1011 1100 1011

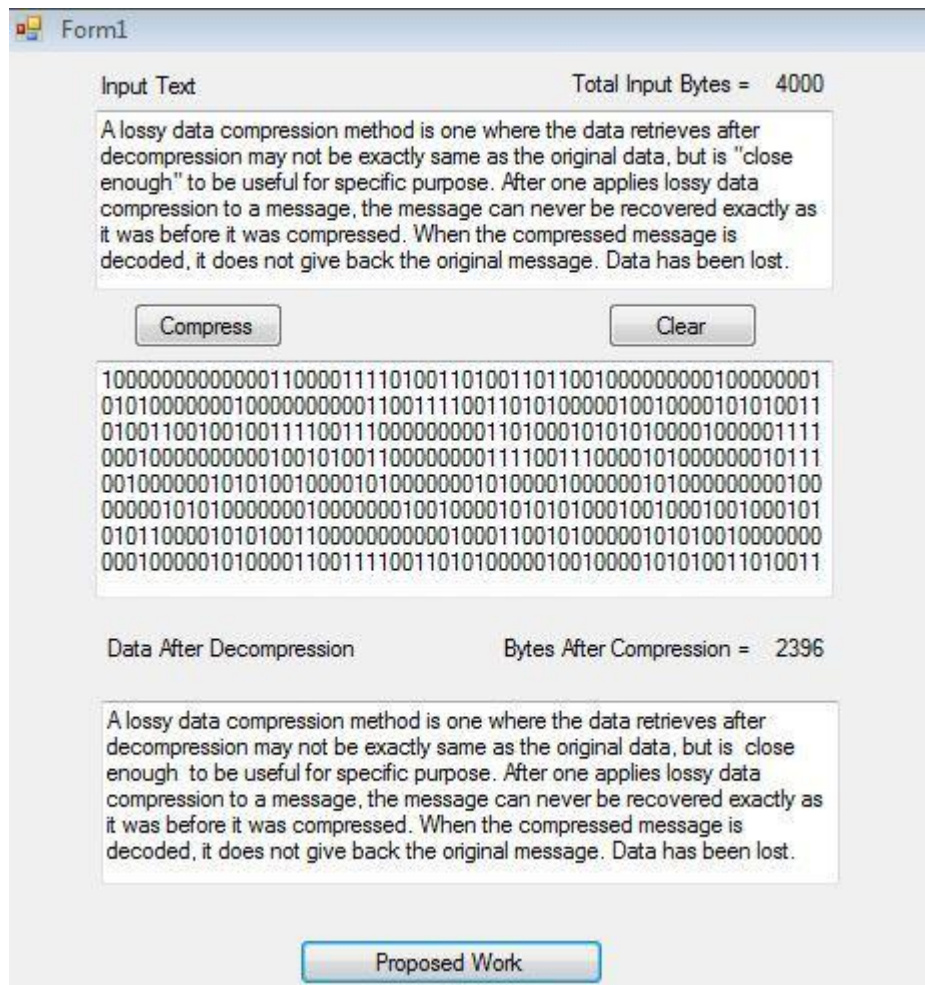
Above illustration shows that the total number of bits to represent the input string are 72.

VII. RESULTS

In this section, the results of the proposed method is presented and discussed. The algorithm has been implemented on c# platform with Visual Studio 2008 as an Integrated Development Environment (IDE). The proposed method is evaluated using different size of data bytes in terms of compression ratio & Saving Percentage and compared with the existing techniques i.e. Bit Reduction algorithm and Huffman coding. Table 1.1 demonstrates the analysis of proposed method using different size of data files and compared with existing methods.

Table 1.1 Comparison of compression ratio and saving percentage of the proposed method and the existing method[13]

Input (in bytes)	Output (in bytes)		Compression Ratio (in %)		Saving Percentage	
	Existing system	Proposed Method	Existing system	Proposed Method	Existing system	Proposed Method
100	51	36	51	36	49	64
200	107	77	53.5	38.5	46.5	61.5
300	167	123	55.6	41	44.3	59
400	229	191	57.2	47.7	42.7	52.2
500	288	243	57.6	48.6	42.4	51.4
1000	593	501	59.3	50.1	40.7	49.9
2000	1193	1005	59.6	50.2	40.3	49.7
3000	1790	1513	59.6	50.4	40.3	49.5
4000	2396	2024	59.9	50.6	40.1	49.4
5000	2975	2546	59.5	50.9	40.5	49.0
10000	5994	5746	59.9	57.4	40.0	42.5



Form1

Input Text Total Input Bytes = 4000

A lossy data compression method is one where the data retrieves after decompression may not be exactly same as the original data, but is "close enough" to be useful for specific purpose. After one applies lossy data compression to a message, the message can never be recovered exactly as it was before it was compressed. When the compressed message is decoded, it does not give back the original message. Data has been lost.

```
100000000000011000011110100110100110110010000000010000001
01010000000100000000011001111001101010000010010000101010011
01001100100100111100111000000001101000101010100001000001111
00010000000001001010011000000001111001110000101000000010111
00100000010101001000010100000001010000100000010100000000100
0000010101000000010000000100100001010100010010001001000101
01011000010101001100000000001000110010100000101010010000000
000100000101000011001111001101010000010010000101010011010011
```

Data After Decompression Bytes After Compression = 2396

A lossy data compression method is one where the data retrieves after decompression may not be exactly same as the original data, but is close enough to be useful for specific purpose. After one applies lossy data compression to a message, the message can never be recovered exactly as it was before it was compressed. When the compressed message is decoded, it does not give back the original message. Data has been lost.

Fig. 1.5 input and output of the existing system

Fig. 1.5 shows the data compression for existing method by considering 4000 bytes as input and 2396 bytes are output after data compression. Fig. 1.6 shows the data compression process of the proposed method. by considering 4000 bytes as input and 2024 bytes are output after data compression.

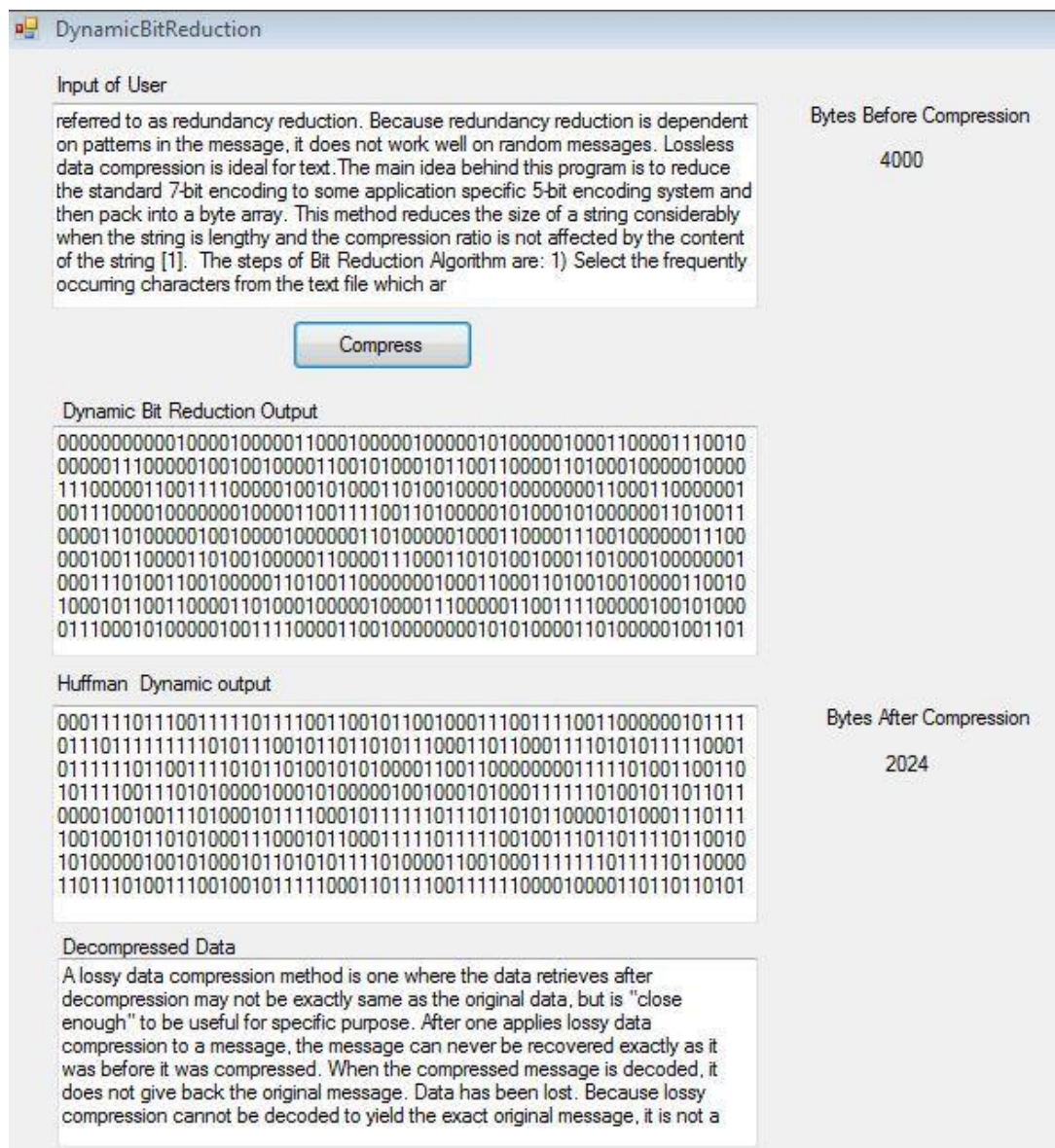


Fig. 1.6 input and output of the proposed method

VIII. CONCLUSION

In this study, a dynamic Bit Reduction algorithm and Huffman Coding is used to improve the compression ratio and saving percentage. Different size of data files is used to evaluate the performance of the proposed method. The results obtained by the proposed method are compared with the existing data compression techniques like Bit Reduction and Huffman Coding in terms of compression ratio and saving percentage.

REFERENCES

- [1] R. Brar, and B. Singh, A Survey on Different Compression Techniques and Bit Reduction Algorithm for Compression of Text/Lossless Data, International Journal of Advanced Research in Computer Science and Software Engineering, 3(3), 2013, 579-582.

- [2] N. Sharma et al. An Improved Dynamic Bit Reduction Algorithm for Lossless Text Data Compression, International Journal of Advanced Research in Computer Science and Software Engineering, 4(7), 2014, 1023-1029.
- [3] A.S. Sidhu, and M. Garg, Research Paper on Text Data Compression Algorithm using Hybrid Approach, International Journal of Computer Science and Mobile Computing, 3(12), 2014, 1-10.
- [4] A.S. Sidhu, and E.M. Garg, An Advanced Text Encryption & Compression System Based on ASCII Values & Arithmetic Encoding to Improve Data Security, International Journal of Computer Science and Mobile Computing, 3(10), 2014, 45-51.
- [5] S. Porwal, Y. Chaudhary, J. Joshi, and M. Jain, Data Compression Methodologies for Lossless Data and Comparison between Algorithms, International Journal of Engineering Science and Innovative Technology, 2(2), 2013, 142-147.
- [6] S. Shanmugasundaram and R. Lourdasamy, a Comparative Study of Text Compression Algorithms, International Journal of Wisdom Based Computing, 1(3), 2011, 68-76.
- [7] P. Yellamma, and N. Challa, Performance Analysis of Different Data Compression Techniques on Text File, International Journal of Engineering Research & Technology, 1(8), 2012, 1-6.
- [8] Aarti. A Review on Lossless Image Compression Techniques, International Journal of Advanced Research in Computer Science and Software Engineering, 3(11), 2013, 1483-1495.
- [9] S.R. Kodituwakku, and U. S. Amarasinghe, Comparison of Lossless Data Compression Algorithms For Text Data, Indian Journal of Computer Science and Engineering, 11(4), 416-425.
- [10] M.H. Btoush, et al., Observations on Compressing Text Files of Varying Length, Proc. 5th IEEE International Conf. on Information Technology: New Generations, 2008, 224-228.
- [11] U. Katugampola, A New Technique for Text Data Compression, Proc. IEEE International Symposium on Computer, Consumer and Control, 2012, 405-409.
- [12] S. Kapoor, and A. Chopra, A Review of Lempel Ziv Compression Techniques, International Journal of Computer science and Telecommunications, 4(2), 2013, 246-248.
- [13] A. Kaur, and N. Sethi, Approach for Lossless Text data Compression using Advanced Bit Reduction Algorithm, International Journal of Advanced Research in Computer Science and Software Engineering, 5(7), 2013, 1172-1176.
- [14] A. Moffat, et al., Static Compression for Dynamic Texts, IEEE Computer, 1994, 126-135.
- [15] J. Ziv, and A. Lempel, a Universal Algorithm for Sequential Data Compression, Proc. IEEE Transactions on Information Theory, 23(3), 1977, 337-343.
- [16] M. Burtscher, et al., The VPC Trace-Compression Algorithms, Proc. IEEE Transaction on Computers, 54(11), 2005, 1329-1344.
- [17] M.Z. Zia, et al., Two-Level Dictionary-Based Text Compression Scheme, Proc. 11th IEEE International Conf. on Computer and Information Technology, 2008, 13-18.