A Comparative Study of Lossless Data Compression Techniques

Elham Yakhlef Abushwashi
Technical electronic department
Technical faculty engineering, Zwara
Elham_abuelshwashi@yahoo.com

Hamida Aboulqasim Oushah
Software development department
College of Computer Technology, Zawia
e_hamida@yahoo.com

Abstract

Data compression is a process that reduces the data size, removing the excessive information and redundancy. It is a common and important requirement for most of the computerized applications, it can shorter the data size, which lead to cost reduction. The main purpose of data compression is to remove data redundancy from the store or transmitting data, it is also an important application in file storage field and distributed system. Data compression techniques are can be used in different data formats such as text, audio, video and image files. The aim of the study is to compare between many of the Lossless data compression techniques and compare their performance. There are many techniques of data compression and they can be categorized as Lossy and Lossless Compression methods. In this study a Run-length encoding, Huffman Coding, Shannon-Fano Coding, and LZW Encoding algorithm were used, their performance were compared by using data compression in the text format, the compression ratio, compression factor and saving percentage were calculated. Compression ratio in Huffman coding and Shannon-Fano

coding where less than Run-length encoding and LZW encoding (38%, 40%, %81, 74%) respectively, while compression factor where higher than Run-length encoding and LZW encoding (2.63, 2.48, 1.23, 1.35) respectively, the results of saving percentage by using Huffman coding and Shannon-Fano coding where higher than Run-length encoding and LZW encoding (62%,60%, 19%, 26%) respectively. The study pointed to the effectiveness of Huffman and Shannon-Fano coding reducing the size of the files compare to other algorithms.

Key Keywords: Data compression, Lossless data compression technique, Huffman Coding, Run length encoding, Shannon-Fano coding, LZW encoding.

1. INTRODUCTION

Data compression is a way to reduce storage cost by eliminating redundancies that happen in most files. There are two types of compression, lossy and lossless. Lossy compression reduced file size by eliminating some unneeded data that won't be recognize by human after decoding, this often used by video and audio compression[1]. Lossless compression on the other hand, manipulates each bit of data inside file to minimize the size without losing any data after decoding. This is important because if file lost even a single bit after decoding, that mean the file is corrupted. Lossless data compression is a technique that allows the use of data compression algorithms to compress the text data and also allows the exact original data to be reconstructed from the compressed data. This is in contrary to the lossy data compression in which the exact original data cannot be reconstructed from the compressed data. The popular ZIP file format that is being used for the compression of data files is also an application of lossless data compression approach. Lossless compression is used when it is important that the original data and the decompressed data be identical. Lossless text data compression algorithms usually exploit statistical redundancy in such a way so as

to represent the sender's data more concisely without any error or any sort of loss of important information contained within the text input data. Since most of the real-world data has statistical redundancy, therefore lossless data compression is possible. Lossless compression methods may be categorized according to the type of data they are designed to compress. Compression algorithms are basically used for the compression of text, images and sound. Most lossless compression programs use two different kinds of algorithms: one which generates a statistical model for the input data and another which maps the input data to bit strings using this model in such a way that frequently encountered data will produce shorter output than improbable(less frequent) data. The advantage of lossless methods over lossy methods is that Lossless compression results are in a closer representation of the original input data. The performance of algorithms can be compared using the parameters such as Compression Ratio and Saving Percentage. In a lossless data compression file the original message can be exactly decoded. Lossless data compression works by finding repeated patterns in a message and encoding those patterns in an efficient manner. For this reason, lossless data compression is also referred to as redundancy reduction[2]. Because redundancy reduction is dependent on patterns in the message, it does not work well on random messages. Lossless data compression is ideal for text.

2. COMPRESSION TECHNIQUES

There are two categories of compression techniques, lossy and lossless. Whilst each uses different techniques to compress files, both have the same aim: To look for duplicate data in the data.

2.1 Lossless Compression

Lossless data compression is a class of data compression algorithms that allows the original data to be perfectly reconstructed from the compressed data. In lossless data compression, the integrity of the data is preserved. Redundant data is removed in compression and added during decompression. Lossless compression methods are normally

used in cases where it is important that the original and the decompressed data be identical [3].

2.2 Lossy Compression

In lossy data compression original data is not exactly restored after decompression and accuracy of reconstruction is traded with efficiency of compression. This type of compression used for image data compression. The decompression ratio is high compare to lossless data compression technique. Sometimes some loss of quality is acceptable. For example the human ear cannot hear all frequencies, people can't hear may end up with a smaller file, but it is not possible to get back to how exactly the original music sounded [2]. In such cases, we can use a lossy data compression methods. These methods are cheaper, they take less time and space when it comes to sending millions of bits per second for images and video.

3. LOSSLESS COMPRESSION TECHNIQUES

3.1 Run-length encoding(RLE)

Run Length Encoding (RLE) is the simplest of the data compression algorithms. It is created especially for data with strings of repeated symbols [3]. The consecutive sequences of symbols are identified as runs and the others are identified as non runs in this algorithm [4]. The general idea behind this algorithm is to replace consecutive repeating occurrences of a symbol by one occurrence of the symbol followed by the number of occurrences. The RLE algorithm uses those runs to compress the original source while keeping all the non-runs without using for the compression process [3].
For example, consider the following text string:

eelhhhaaammhhhaaammmiiddaaa = 27 characters
= 27 * 8 bits for each character
= 216 bits

Compressed string:

2l1h3a3m2h3a3m3i2d2a3 = 22 characters

= 22 * 8 bits for each character

= 176 bits

3.2 Huffman Coding

One of the most popular techniques for removing coding redundancy is due to Huffman [3]. Huffman coding was developed by Dr. David A. Huffman in 1952. Huffman coding assigns shorter codes to symbols that occur more frequently and longer codes to those that occur less frequently [5].

There are mainly two major parts in Huffman Coding

- Build a Huffman Tree from input characters.
- Traverse the Huffman Tree and assign codes to characters.

Huffman uses bottom-up approach, it is simple and can be described in terms of creating a Huffman code tree [3].
Example for Huffman coding:

Elhamabushwashihamidaoushah

Count of symbols stream as shown in Table 1:

Table 1: Huffman symbols stream

Symbol	**H**	**a**	**s**	**m**	**u**	**i**	**e**	**l**	**B**	**w**	**d**	**O**
Count	6	6	3	2	2	2	1	1	1	1	1	1

The tree of Huffman example is shown below in Fig.1:

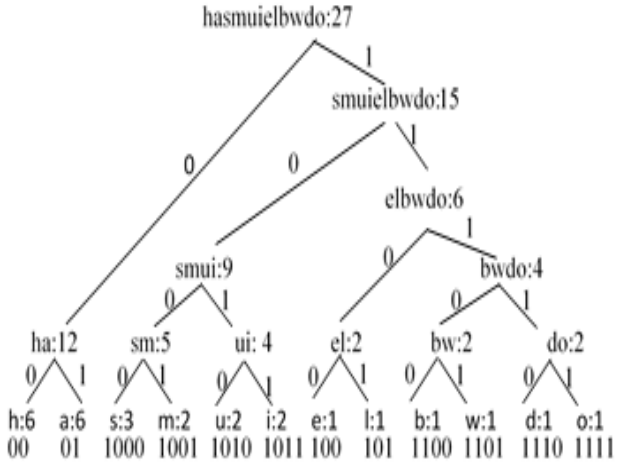


Fig. 1: Huffman Tree

The Table 2 illustrates the total length of compression output.

Table 2: Huffman total length

Symbol	Freq.	code	code length	total length
H	6	00	2	12
A	6	01	2	12
S	3	100	3	9
M	2	1001	4	8
U	2	1010	4	8
I	2	1011	4	8
E	1	100	3	3
L	1	101	3	3
B	1	1100	4	4
W	1	1101	4	4
D	1	1110	4	4
O	1	1111	4	4
Total				79 bits

Input:

elhamabushwashihamidaoushah = 27 characters

= 27 * 8 bits for each character

= 216 bits

Output:

100101000110010111001010100000110101100000101100011001101
111100111111010100000100 = 82 bits

3.3 Shannon-Fano Coding

This is one of an earliest technique for data compression that was invented by Claude Shannon and Robert Fano in 1949.In this technique, Shannon-Fano Algorithm is a top-down approach, 1 sort symbols according to their frequencies [4].

A simple example will be used to illustrate the algorithm:

Input stream:

elhamabushwashihamidaoushah

Frequency symbols in stream as shown in Table 3:

Table 3: Frequency of character

Symbol	**h**	**A**	**s**	**M**	**u**	**i**	**e**	**l**	**b**	**w**	**d**	**O**
Count	6	6	3	2	2	2	1	1	1	1	1	1

Recursively divide into two parts, each with approximately same number of counts, i.e. split in two so as to minimize difference in counts. Left group gets 0, right group gets 1 [6]. Fig. 2 shows the Shannon coding tree.

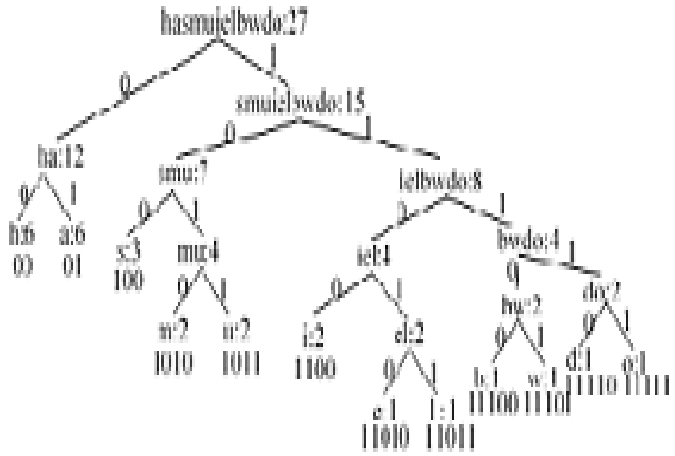


Fig. 2: Shannon-Fano Tree

The following table illustrates the complete operation for this algorithm.

Table 4: Shannnon-Fano total length

Symbol	Freq.	Code	code length	total length
H	6	00	2	12
A	6	01	2	12
S	3	100	3	9
M	2	1010	4	8
U	2	1011	4	8
I	2	1100	4	8
E	1	11010	5	5
L	1	11011	5	5
B	1	11100	5	5
W	1	11101	5	5
D	1	11110	5	5
O	1	11111	5	5
Total				87 bits

Input:

elhamabushwashihamidaoushah = 27 characters

= 27 * 8 bits for each character

= 216 bits

Output:

110101101100011010011110010111000011101001000011000001101
011001111001111111011100000100 = 87 bits

3.4 LZW Encoding Algorithm

LZW is the first letter of the names of the scientists Abraham Lempel, Jakob Ziv, and Terry Welch, who developed this algorithm. LZW compression is a lossless compression algorithm.

LZW algorithm is just like a greedy approach and divides text into substrings. LZW compression algorithm is dictionary based algorithm which always output a code for a character. Each character has a code and index number in dictionary. Input data which we want to compress is read from file. Initially data is entered in buffer for searching in dictionary to generate its code. If there is no matching character found in dictionary. Then it will be entered as new character in dictionary and assign a code. If character is in dictionary then its code will be generate. Output codes have less number of bits than input data. This technique is useful for both graphics images and digitized voice [7]. Table 4 shows the complete operation of this algorithm.

Table 5: LZW operations

Input string= elhamabushwashihamidaoushah			
Character input	***Code output***	***New code value***	***New string***
E	E	None	
E	L	256	EL
L	H	257	LH
H	A	258	HA
A	M	259	AM
M	A	260	MA
A	B	261	AB
B	U	262	BU
U	S	263	US
S	H	264	SH
H	W	265	HW
W	A	266	WA
A	S	267	AS
S	H	264	SH
SH	I	268	SHI
I	H	269	IH
H	A	258	HA
HA	M	270	HAM
M	I	271	MI
I	D	272	ID
D	A	273	DA
A	O	274	AO
O	U	275	OU
U	S	263	US
US	H	276	USH
H	A	258	HA
HA	H	277	HAH

Input: elhamabushwashihamidaoushah = 27 characters

= 27 * 8 bits for each character

= 216 bits

Output: ellhhaammaabbuusshhwwaasshiihhammiiddaauushhah = 87 bits

4. MEASURING COMPRESSION PERFORMANCES

Performance measure is use to find which technique is good according to some criteria. There are various criteria to measure the performance of compression algorithm. Since the compression behavior depends on the redundancy of symbols in the source file, it is difficult to measure performance of compression algorithm in general. The performance of data compression depends on the type of data and structure of input source. The compression behavior depends on the category of the compression algorithm: lossy or lossless [8]. Following are some measurements used to evaluate the performances of lossless algorithms.

Compression Ratio: is the ratio between the size of the source file and the size of the compressed file.

$$compression\ ratio = \frac{Amount\ data\ bits\ compression}{Amount\ data\ bits\ before\ compression}$$

Compression Factor: is the inverse of the compression ratio. That is the ratio between the size of the source file and the size of the compressed file.

$$compression\ factor = \frac{Amount\ data\ bits\ before\ compression}{Amount\ data\ bits\ after\ compression}$$

Saving Percentage: calculates the shrinkage of the source file as a percentage.

$$saving\ percentage = \frac{Amount\ data\ bits\ before\ compression - Amount\ data\ bits\ after\ compression}{Amount\ data\ bits\ before\ compression}\%$$

5. RESULTS

In this work mainly focused on performance of four lossless compression algorithms.

The table 6 shows the comparative results (output size, compression ratio, compression factor, and saving percentage) between our selected algorithms.

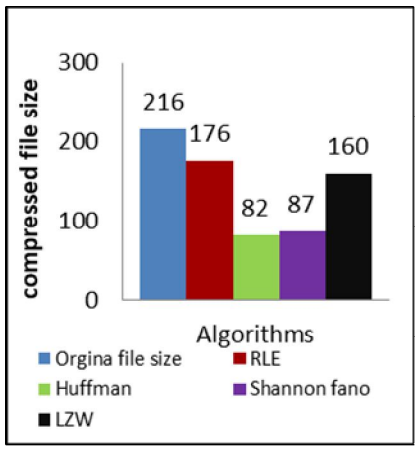
Table 6: Comparative results.

	Algorithm			
	RLE	***Huffman***	***Shannon-Fano***	***LZW***
Input size (bits)	216	216	216	216
Output size (bits)	176	82	87	160
Compression Ratio (%)	81	38	40	74
Compression Factor	1.23	2.63	2.48	1.35
Saving Percentage(%)	19	62	60	26

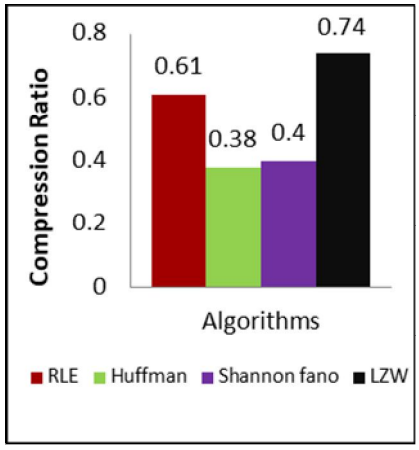
According to the results shown in table 6, Huffman and Shannon-Fano coding can reduce the file size around 50% of original file size, while the Shannon-Fano Coding, and LZW Encoding algorithm can reduce approximately (1.2%), This finding was consistent with study conducted Achinta,2016 [1]. Study, the compression ratio of Huffman and Shannon-Fano coding were more effective than the Shannon-Fano Coding, and LZW Encoding, this result compatible with study done by K.A. Ramya.2006 [2], compression factor is the inverse of compression ratio, and saving percentage of the Huffman and Shannon-Fano coding is better while compared to others.

The bar chart shows the original file size before and after compression, comparison between compressed file size, compression

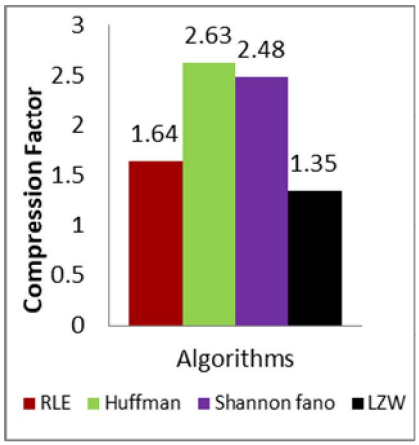
ratio, compression factor, and saving percentage, by using Huffman, Shannon-Fano coding, Run Length encoding, and LZW encoding in Fig. 3. In Fig. 3a. the graph shows original file size before compression (216 bits), and the size after compression using Huffman (82 bits), Shannon-Fano (87 bits), Run Length (176 bits), and LZW (160 bits), it is clear that the file compressed near to the half by using by Huffman and Shannon-Fano coding.



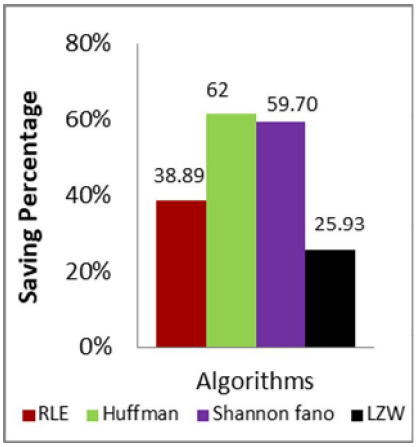
a. Compressed file size.



b. Compression ratio.



c. Compression factor.



d. Saving percentage.

Fig. 3. Compartive Results

In Fig. 3b compression ratio calculated by Huffman and Shannon-Fano coding are almost half the compression ratio calculated by Run Length encoding, and LZW encoding, in fig. 3c the compression factor calculated by Huffman and Shannon-Fano coding is almost double the compression factor calculated by the Run Length encoding, and LZW encoding.

In Fig. 3d the saving percentage is almost 60% by using Huffman and Shannon-Fano coding with and around 26% by using Run Length encoding, and LZW encoding.

6. CONCLUSION

Data text compression using Huffman and Shannon-Fano coding algorithms give small size of data in compare to Run-length and LZW encoding.

Compression Huffman and Shannon-Fano coding algorithms in data text resulting a smaller size of data than the size of the data by Run-length and LZW encoding

Huffman and Shannon-Fano coding are very powerful over Run-length and LZW encoding, they provide better results and reduce the size of the text. Run-length encoding more effective when data text with strings of continuous repeated symbols.

REFERENCES

1. Achinta Roy, Dr. Lakshmi Prasad Saikia, "A Comparative Study of Lossless Data Compression Techniques on Text Data" International Journal of Advanced Research in Computer Science and Software Engineering, Volume 6, Issue 2, February 2016.
2. K.A. Ramya1, M.Pushpa, ,M.Phil Student, Assistant Professor, "Comparative Study on Different Lossless Data Compression Methods", International Journal of Scientific Engineering and Applied Science (IJSEAS) - Volume-2, Issue-1,January 2016 ISSN: 2395-3470 www.ijseas.com.

3. P.RAVI1, Dr.A.ASHOKKUMAR, “A Study of Various Data Compression Techniques”, International journal of computer scinece & communication, Volume 6, Issue 2, April-September 2015, ISSN:0973-7391.
4. Amit Jain a * Kamaljit I. Lakhtariab, Prateek Srivastava, “A Comparative Study of Lossless Compression Algorithm on Text Data”, Proc. of Int. Conf. on Advances in Computer Science, AETACS, 2013.
5. Dr. AMIN MUBARK ALAMIN IBRAHIM, Dr. MUSTAFA ELGILI MUSTAFA, “Comparison Between (RLE And Huffman) Algorithmsfor Lossless Data Compression”, (IJITR) INTERNATIONAL JOURNAL OF INNOVATIVE TECHNOLOGY AND RESEARCH, Volume No.3, Issue No.1, December – January 2015, 1808 – 1812.
6. B.A. Al-hmeary, “Role of Run Length Encoding on Increasing Huffman Effect in Text Compression”, Journal of Kerbala University , Vol. 6 No.2 Scientific. 2008.
7. Mamta Sharma, S.L. Bawa D.A.V. college, “Compression Using Huffman Coding”, IJCSNS International Journal of Computer Science and Network Security”, VOL.10 No.5, May 2010.
8. Pooja Singh, “LOSSLESS DATA COMPRESSION TECHNIQUES AND COMPARISON BETWEEN THE ALGORITHMS”, International Research Journal of Engineering and Technology (IRJET), e-ISSN: 2395-0056, p-ISSN: 2395-0072, Volume: 02 Issue: 02 May-2015. www.irjet.net