المجلة الليبية لعلوم التعليم العدد الرابع أغسطس– 2021 – ISSN 202153

Analysis and comparison of data compression Techniques and their application to text files

Jalal M. Mehalhal[1*], $AdemA.bensaid^{2}$, $Muhmed\ F.Agbisha^{1}$

Abstract

Due to the rapid development in information technology in terms of information exchange and transmission through different transmission media, and the provision of storage places. when the volume of data is smaller, this means that, it provides better transmission speed. and saves time, which led to the emergence of data compression techniques to reduce its size without compromising the quality of the data.

Data compression is still an important topic of research and has many applications and required uses.

This paper presents a study of some of the data compression methods: Huffmann and Huffmann shift code, binary shift code algorithm, and the LZW method, analyzing and comparing between them, using a fixed text for all methods.

keywords: Data compression, compression techniques, Huffmann, Huffmann shift code, binary shift code , LZW.

1. Introduction

Data compression technology is primarily a branch of information theory that deals with techniques related to minimizing the amount of data to be transferred and preserved. Data compression is a method of representing data using fewer bits than the original data [6].

Data compression is of great importance in business data processing, as it helps us reduce resource usage such as data storage space or transmission capacity [5].

With the increasing development of technology and internet networks supported by programs and devices that facilitate the spread of information very quickly over the Internet around the world. The information obtained can be easily sent over the Internet as a means of communication for IT experts. However, not all information can be sent easily. There is a large volume that can hinder the fast data transfer and save on the storage in the computer. There are a number of different data compression methodologies that use technologies mainly for speed, efficiency, performance as well as cost savings [2].

Compression is the process of converting a data set into a code to save the need for storage and transmission of data making it easier to transmit a data.

With the compression of a can save in terms of time and storage that exist in memory. The data process of data compression is shown in figure 1[2].

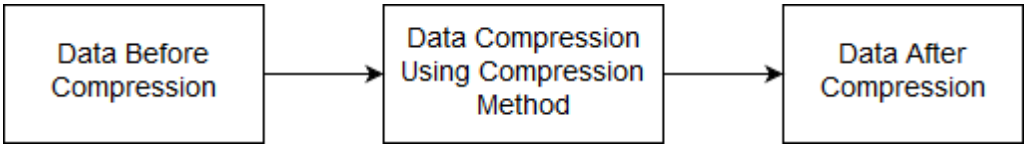

Figure 1: The data process of data compression [2]

In figure 1, explain the process of data compression in general. how the data when not compressed then uncompressed data will be continued and processed by compression method that is lossless compression then the data has been compressed will produce a size smaller than the size of the file before it is compressed.

1.1 CLASSIFICATION METHODS :

Data compression methods can also be categorized into static and dynamic compression methods. In static method, mapping from

the set of messages to the set of codeword is fixed before transmission begins. Huffman coding [Huffman 1952] is the example of classic static defined word scheme. On the other hand, dynamic method changes the mapping of set of messages to the set of codeword over a period of time. For instance dynamic Huffman coding computes approximate probability of occurrence of a set of characters in a message[6].

1.2 FUNDAMENTALS FOR COMPRESSION

Compression can be divided into two categories, as Lossy and Lossless compression.

Lossy compression means that some data is lost when it is decompressed. Lossy compression bases on the assumption that the current data files save more information than human beings can "perceive". Thus the irrelevant data can be removed.

Lossless compression means that when the data is decompressed, the result is a bit-for-bit perfect match with the original one. The name lossless means "no data is lost", the data is only saved more efficiently in its compressed state, but nothing of it is removed.

1.3 Data Compression

Data compression is a procedure through which a file (text, Audio, and Video) could also be modified to one more (compressed) file, such that the normal file could also be completely recovered from the long-established file without any loss of exact knowledge. This process may be subsidiary if one wishes to save lots of the storage space. For instance if one wishes to retailer a 4MB file, it is usually top-rated to first compress it to a smaller size to save the storage space. Additionally compressed files are much more effectively exchanged over the web for the reason that they add and down load much faster. We require the potential to reconstitute the original file from the compressed

variation at any time. Data compression is a method of encoding rules that sanctions substantial reduction in the total number of bits to store or transmit a file. The more data being handled, the more it costs in phrases of storage and transmission costs. In short, Data Compression is the method of encoding data to fewer bits than the customary illustration in order that it takes less storage space and not more transmission time even as communicating over a network. There are two mainly two types of Data Compression:

1. Lossy Compression
2. Lossless Compression[7]

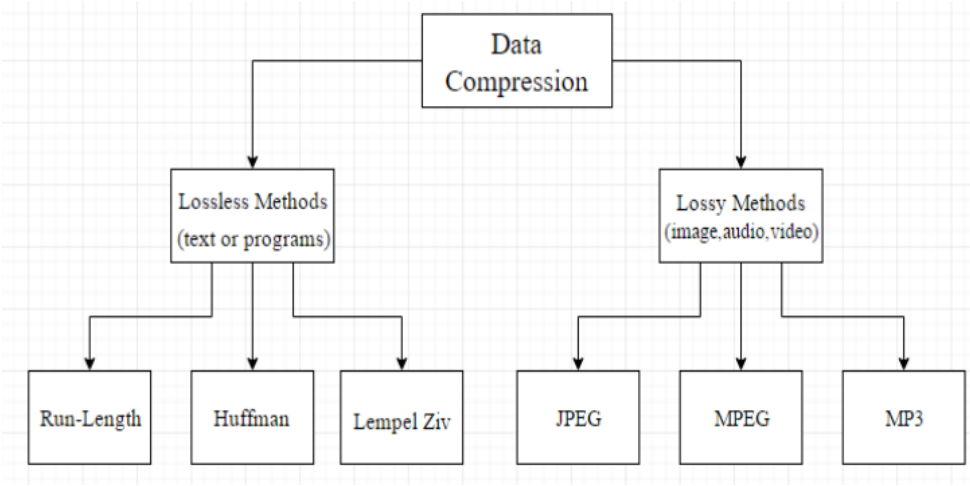

Figure 2: Types of Data Compression

2. Related works

In Paper [1] , researchers conducted a study about compression methods (Huffman, Shannon Fano, Tunstall, Lempel Ziv Welch and run-length encoding) and applied them to a text file and how each method works and compares between them.

In Paper[2], researchers studied compression methods (LZW, Huffman, Fixed-length code (FLC), and Huffman after using Fixed-length code (HFLC)).

And applied them to a group of different files and comparing between them, and as a result of that it was concluded that the LZW method is the best, especially for large files and then huffman respectively.

3. Data Compression Techniques

There are two different ways that data compression methods and algorithms can be categorized as the lossless and lossy, the methods are classified according to a fixed or variable. Lossless compressions are run-length; Huffman, delta, LZW etc.[5].

3.1 Huffman Compression Technique

There are many types of Huffman coding, some are use a Huffman-like algorithm, and others find optimal prefix codes (while, for example, putting different restrictions on the output). In the latter case, the method need not be Huffman-like, and, indeed, need not even be polynomial time. An exhaustive list of papers on Huffman coding and it's variations are given here as follows:

The n-ary Huffman algorithm uses the {0, 1, ... , n − 1} alphabet to encode message and build an n-ary tree. This approach was considered by Huffman in his original paper. The same algorithm applies as for binary (n equals 2) codes, except that the n least probable symbols are taken together, instead of just the 2 least probable. Note that for n greater than 2, not all sets of source words can properly form an n-ary tree for Huffman coding. In this case, additional 0-probability place holders must be added.

This is because the tree must form an n to 1 contractor; for binary coding, this is a 2 to 1 contractor, and any sized set can form such a contractor. If the number of source words is congruent to 1 modulo n-1, then the set of source words will form a proper Huffman tree.

A variation called adaptive Huffman coding which is involved for calculating the probabilities dynamically based on recent actual frequencies in the sequence of source symbols, and changing the coding tree structure to match the updated probability estimates.
Most often, the weights used in implementations of Huffman coding represent numeric probabilities, but the algorithm which given above does not require this; it requires only the weights form a totally ordered commutative monoid, meaning a way to order weights and to add them. The Huffman template algorithm enables one to use any kind of weights (costs, frequencies, pairs of weights, nonnumerical weights) and one of many combining methods (not just addition). Such algorithms can solve other minimization problems, a problem first applied to the circuit design.
Length-limited Huffman coding is a variant where the goal is still to achieve a minimum weighted path length, but there is an additional restriction that the length of each code word must be less than a given constant. The package-merge algorithm solves this problem with a simple greedy approach which is very similar to that is used by Huffman's algorithm. It’s time complexity is O(nL), where L is the maximum length of a code word. No algorithm is known to solve this problem in linear or linearithmic time, unlike the pre-sorted and unsorted conventional Huffman problems, respectively[5].

3.2 LZW Compression Technique

LZW compression named after its developers, A. Lempel and J. Ziv, with later modifications by Terry A. Welch. It is the foremost technique for general purpose data compression due to its simplicity and versatility. Typically, you can expect LZW to compress text, executable code, and similar data files to

about one-half their original size. LZW also performs well when presented with an extremely redundant data files, such as tabulated numbers, computer LZW is the basis of several personal computer utilities that claim to "double the capacity of your hard drive." If the codeword length is not sufficiently large, Lempel-Ziv codes may also rise slowly to reasonable efficiency, maintain good performance briefly, and fail to make any gains once source code, and acquired signals. Compression ratios of 5:1 are common for these cases. [5]

LZW Encoding Algorithm

Step 1: At the start, the dictionary contains all possible roots, and P is empty

Step 2: C: = next character in the char stream;

Step 3: Is the string P+C present in the dictionary?

(a) if it is, P := P+C (extend P with C);

(b) if not,

–output the code word which denotes P to the code stream;

– add the string P+C to the dictionary;

–P: = C (P now contains only the character C); (c) Are there more characters in the char stream?

–if yes, go back to step 2;

–if not:

Step 4: Output the code word which denotes P to the code stream;

Step 5: END.

The fundamental theory of LZW compression algorithm is: any predictable data can demonstrate such predictability by certain mark and shorten the data length. During the process of LZW compression, as the length of each code in the code stream of a datum produced after compression is less than N, or the compression algorithm can represent number of 0~2n- l, therefore the string list can accommodate number of 2n at the maximum.

But each element of the input data stream after compression is a byte, which represents 0~255 possible assignments. Secondly, compression program takes one byte from
the input character stream. Two buffers of current prefix code and current string are used to store data.
Prefix position is for the code processed last time, current string is for the character string represented by prefix code and characters read just now. When the program starts, both the prefix code and current string are blank. Thirdly, program searches for current string in the string list after initialization.
Fourthly, read the next byte from input stream, and add this character behind current string, now the current string contains 2 bytes, and repeat the third step.

3.3 Huffman Shift Coding Algorithm

On Shift Huffman Coding, the symbol is divided into several blocks of the same size. Usually the block size is 2^k -1 symbols, where k is a positive integer. If k = 1, then the Shift Huffman Coding Huffman same as Shift Coding Standard.
The symbol of the first block to be encoded using Huffman Coding exactly standard. When encodes a symbol of the first block, were also coded symbol hypothesis frequency of occurrence equal to the number of frequency of occurrence of the symbols of the other blocks. The only difference between
one block to the next block is then the result of the addition of the prefix encoding the hypothesis symbols used to mark each block.
With Huffman Shift Coding algorithm, it could increase the use of less time and average - average length of the code more efficient.
Broadly speaking, the following compression algorithms Huffman Coding Shift work:
1. Source Symbol arranged so that the possibilities that arise from the largest to the smallest.

2. The symbol of the first block will be encoded using Huffman Coding Standard. When encodes a symbol of the first block, were also coded symbol hypothesis frequency of occurrence equal to the number of frequency of occurrence of the symbols of the other blocks. The only difference between one block to the next block is the addition of one or more code prefix result of the encoding symbols to mark each block hypothesis.
3. Symbols hypothesis of Huffman coding, we think of as C.The total number of source symbols are divided into several blocks of symbols of the same size.
4. The second symbol of the code block is C^{K-1} coupled with the symbol of all I first block of Huffman Coding.
There are differences in the formation of the tree Huffman Coding Shift this. As shown in the following figure.[3]

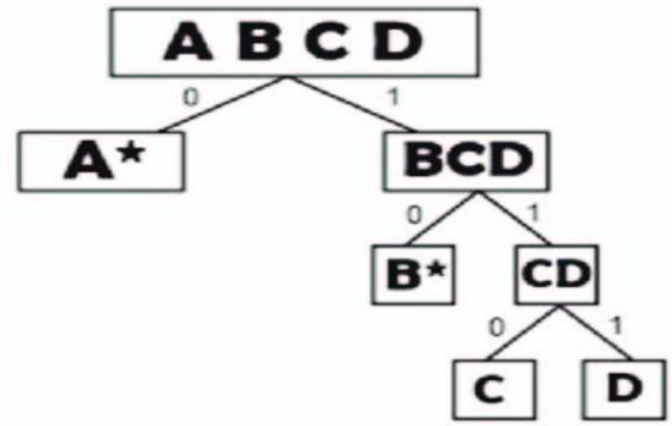

Figure 3:. Huffman Shift Coding process

4. Results and Discussion

Four compression algorithms are tested on a text file in a normal English language format and calculate the message size before and after the implementation of the algorithm and compare them.

The following message was chosen for compression analysis, and the selection was random and the results were as follows:

Jalal Mohammed And Adam Are Friends

1- Hoffmann's method of data compression:

An algorithm that does not give a code of equal length for every letter or symbol in the segment is called a variable length cipher algorithm.

The huffman coding technique is the most common method used to remove redundant data based on the following:

- The more prominent symbol is assigned a shorter code than the less visible symbols in the same segment.
- The two less-repeated symbols will have a code of the same length, and differ in the way it is represented.

To apply the method to the chosen text, we follow the following steps:

1- We do a statistic about the file whose size is to be reduced, after

repeating each letter.

J	A	L	Space	M	O	H	E	D	N	R	F	I	S
1	7	2	5	4	1	1	3	4	2	2	1	1	1

Then we arrange the letters by the number of repetitions

A	SP	M	D	E	L	N	R	J	O	H	F	I	S
7	5	4	4	3	2	2	2	1	1	1	1	1	1

2- We build a binary tree by selecting the letters with the least repetition and connecting them together.

3- We record the paths for each letter of the original letters of the file, by tracing the paths of binary tree from root to leaves.

When creating the compressed file, we replace the original characters with their paths computed in the previous step.

letter	J	A	L	Space	M
code	00110	000	0010	011	110
letter	O	H	E	D	N
code	00111	01010	111	101	0100
letter	R	F	I	S	
code	1000	01011	10010	10011	

We can calculate the size and percentage of the new file by multiplying the path length of each letter by the letter frequency.

letter	J	A	L	Space	M
code	00110	000	0010	011	110
No of bits	5	3	4	3	3
No of repetitions of the character	1	7	2	5	4
Total	5	21	8	15	12
Letter	O	H	E	D	N
Code	00111	01010	111	101	0100
No of bits	5	5	3	3	4
No of repetitions of the character	1	1	3	4	2
Total	5	5	9	12	8
Letter	R	F	I	S	
Code	1000	01011	10010	10011	
No of bits	3	5	5	5	

No of repetitions of the character	2	1	1	1	
Total	6	5	5	5	

From here we note :
The total size of the message: 8 * 35 =280 bit
The size of the message after applying the compression algorithm:121 bit
Data compression ratio = (total message size - message size after compression) / message size
=(280-121)/280=0.567
That is, reducing the size of the message to 56% of the original size.

2-The HUFFMAN SHIFT CODE algorithm

1- We arrange the symbols according to the most probability.
2 Divide the total number of message codes into blocks of a number of characters.
3- The letter is encoded according to the Hoffman code.
4- Encoding the elements inside each block with the same encoding for the previous group.
5- Adding a special code indicating each block, and this code is known to
the FAC code (decoding algorithm).

letter	A	Space	M	D	E
code	00	01	10	110	111
letter	L	N	R	J	O
code	0000	0001	0010	00110	00111
letter	H	F	I	S	
code	000000	000001	000010	0000110	

6-Calculate the size of the message after using the method:
The number of bits represented by each letter is counted and multiplied by the number of times the character is repeated:

letter	A	Space	M	D	E
code	00	01	10	110	111
No of bits	2	2	2	3	3
No of repetitions of the character	7	5	4	4	3
Total	14	10	8	12	9
letter	L	N	R	J	O
code	0000	0001	0010	00110	00111
No of bits	4	4	4	5	5
No of repetitions of the character	2	2	2	1	1
Total	8	8	8	5	5
letter	H	F	I	S	
code	000000	000001	000010	0000110	
No of bits	6	6	6	7	
No of repetitions of the character	1	1	1	1	
Total	6	6	6	7	

From here we note :
The total size of the message: 8 * 35 =280 bit
The size of the message after applying the compression algorithm:112 bit

Data compression ratio = (total message size - message size after compression) / message size

=(280-112)/280=0.6

That is, reducing the size of the message to 60% of the original size.

3- the binary shift code

We follow the same Hoffmann indentation method, but Hoffman's code is not used in Step # 3 and instead is used regular binary encoding.

According to the following steps:

1. Find the repetition of letters and their order.
2. Find the binary representation of the letter letters.
3. The letters are divided into three groups and the binary representation of the letters is done using the regular binary representation for one group of text, then adding the same code to the other groups with the addition of a code representing each group.

So we get the following table:

letter	A	Space	M	D	E
code	000	001	010	011	100
letter	L	N	R	J	O
code	111000	111001	111010	111011	111100
letter	H	F	I	S	
code	111111000	111111001	111111010	111111011	

To calculate the text size after using the method, we use the following table:

letter	A	Space	M	D	E
code	000	001	010	011	100
No of bits	3	3	3	3	3
No of repetitions of the character	7	5	4	4	3
Total	21	15	12	12	9
letter	L	N	R	J	O
code	111000	111001	111010	111011	111100
No of bits	6	6	6	6	6
No of repetitions of the character	2	2	2	1	1
Total	12	12	12	6	6
letter	H	F	I	S	
code	111111000	111111001	111111010	111111011	
No of bits	9	9	9	9	
No of repetitions of the character	1	1	1	1	
Total	9	9	9	9	

From here we note :

The total size of the message: 8 * 35 =280 bit

The size of the message after applying the compression algorithm:153 bit

Data compression ratio = (total message size - message size after compression) / message size

=(280-153)/280=0.454

That is, reducing the size of the message to 45% of the original size.

4- Compression Algorithm LZW

To apply the LZW algorithm to a text we follow the following steps:

1. Define a index for each characters of the language in the dictionary.

Index	Char.	Index	Char.	Index	Char.	Index	Char.
1	A	8	H	15	O	22	V
2	B	9	I	16	P	23	W
3	C	10	J	17	Q	24	X
4	D	11	K	18	R	25	Y
5	E	12	L	19	S	26	Z
6	F	13	M	20	T	27	SPACE
7	G	14	N	21	U		

2. Define indexes to characters within the dictionary.

Index	Pattern	Derived as	Index	Pattern	Derived as
28	J a	17 a	45	A D	1 d
29	A l	1 l	46	D A	4 a
30	L a	12 a	47	A M Space	36 Space
31	A l Space	29 Space	48	Space A R	41 R
32	Space M	27 m	49	R E	18 e
33	M O	13 o	50	E Space	5 Space
34	O H	15 h	51	Space F	27 f
35	H A	8 a	52	F R	6 r
36	A M	1 m	53	R I	18 i
37	M M	13 m	54	I E	9 e
38	M E	13 e	55	E N	5 n
39	E D	5 d	56	N D S	43 s
40	D Space	4 Space			
41	Space A	27 a			
42	A N	1 n			
43	N D	14 d			
44	D Space A	4 Space			

Hence the resulting message from the lzw algorithm is the following string:

17 1 12 29 27 13 15 8 1 13 5 4 27 1 14 40 1 4 36 41 18 5 27 6 18 9 5 43

From here we note :

The total size of the message: 8 * 35 =280 bit

The size of the message after applying the compression algorithm: 224bit

Data compression ratio = (total message size - message size after compression) / message size

=(280-224)/280=0.2

That is, reducing the size of the message to 20% of the original size.

5.Conclusion

This paper provided an overview of general data compression methods and a comparison between them in terms of message size and compression ratio, and it was concluded that Hoffmann's displacement method gave better results compared to Hoffman's method and the binary displacement algorithm, but there is a complexity in representing the data, as for the LZW method, the message size was Large.

It was noted that the algorithm is more practical when the data is dealt with the fewest characters, and thus it is highly efficient when used with images that are represented by a matrix of one and zero, and there are also differences between message sizes after compressing it in different proportions, but each of these methods has Its advantages and disadvantages in pressure.

Future work can be done by using other compression methods and comparing them with the methods used in this paper, and to study the possibility of using data compression methods as data encryption methods.

6.REFERENCE

1. Luluk , Tito and Anggunmeka, " A Review of Data Compression Techniques" International Journal of Applied Engineering Research ISSN 0973-4562 Volume 12, Number 19 (2017) pp. 8956-8963

2. Mohammad Hjouj Btoush, Ziad E. Dawahdeh, "A Complexity Analysis and Entropy for Different Data Compression Algorithms on Text FilesJournal of Computer and Communications, 2018, 6, 301-315

3. Luthfi Firmansah and Erwin Budi Setiawan, "Data Audio Compression Lossless FLAC Format to
Lossy Audio MP3 format with Huffman Shift Coding
Algorithm2016 Fourth International Conference on Information and Communication Technologies (ICoICT)

4. Sanjay Kumar Gupta, " AN ALGORITHM FOR IMAGE COMPRESSION USING HUFFMAN CODING TECHNIQUES" International Journal of Advanced Research in Science and Engineering Vol. No. 5 Issue No.07, July 2016

5. M R Hasan, M I Ibrahimy, S M A Motakabber, M M Ferdaus and M N H Khan, "Comparative data compression techniques and multicompression results" 5th International Conference on Mechatronics (ICOM'13) IOP Conf. Series: Materials Science and Engineering 53 (2013) 012081

6. Gaurav Sethi , Sweta Shaw ,Vinutha K , Chandrani Chakravorty, "Data Compression Techniques" Gaurav Sethi et al, / (IJCSIT) International Journal of Computer Science and Information Technologies, Vol. 5 (4) , 2014, 5584-5586

7. Himadri Parikh1, Jay Amin2, " Data Compression and Steganography Using Shift LSB Algorithm" IJARIIE-ISSN(O)-2395-4396 Vol-2 Issue-3 2016

8. Jyotika Doshi and Savita Gandhi, " Computing Number of Bits to be Processed using Shift and Log in Arithmetic Coding"
International Journal of Computer Applications (0975 – 8887) Volume 62– No.15, January 2013