Comparative Study on Different Lossless Data Compression Methods

K.A. Ramya[1], M.Pushpa[2]

[1] M.Phil Student,
Quaid-E-Millath College for women (Autonomous), Anna Salai, Chennai-02.
ramyanndr@gmail.com

[2] Assistant Professor,
Quaid-E-Millath College for women (Autonomous), Anna Salai, Chennai-02.
push_surya@yahoo.co.in

Abstract

Compression is useful because it is reduce the number of bits needed to represent to data. Compressing can save storage capacity, speed file transfer, and decrease costs for storage hardware and network bandwidth. Compression is performed by a program that uses a formula or algorithm to determine how to shrink the size of data. There are number of data compression techniques used and they can be categorized as Lossy and Lossless Compression methods. In this paper, we discussed about some of the Lossless data compression methods and compare their performance.

Key-words: *Data compression, Lossless compression, Lossy compression, LZW, Shannon-Fano coding and Huffman coding.*

1.Introduction

Most of the real world data's are very redundant. Data Compression is basically defined as a technique that reduces the size of data by applying different methods that can either be Lossy or Lossless [1]. Data compression is a process that reduces the data size by removing the excessive information and redundant[2]. This is a common requirement for most of the computerized application[3]. Compression is a process by which the text, audio, video files may be transformed to another compressed file, such that the original file may be fully recovered from the original file without any loss of actual information[4].

There are two types of data compression namely lossy and lossless. In this paper we would like to discuss about LZW,Shannon-Fano coding and Huffman coding. Lossless data Compression with appropriate example and compare their performance with Compression Ratio, Compression Factor and Saving percentage.

2.Compression Techniques

Compression is built into a wide range of technologies like storage system, operating system, database and software applications. It is useful to reduce the redundancy in data representation thus increasing effective data density. The two basic classes of data compression are applied in different areas. One of these is lossy data compression, which is widely used to compress image data files for communication or archives purposes. The other is lossless data compression that is commonly used to transmit or archive text or binary files required to keep their information intact at any time [4]. The main purpose of this paper to shows the lossless compression techniques and their comparative study.

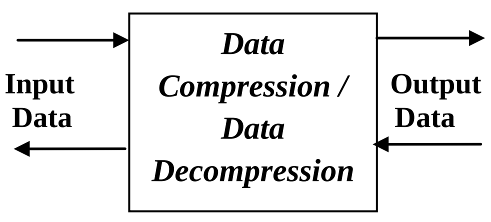

Fig.1 Data Compression and Decompression

2.1 Lossless Compression

Lossless data compression is a technique that allows the use of data compression algorithm to compress the data and also allows the exact the original data to be reconstructed from the compressed data [5]. It is used to reduce the amount of source information to be transmitted in such a way that when compressed information is decompressed, there won't be any loss of information [6]. It is mainly used for text data compression and decompression. It can also be applied to image compression[7]. Some of the popular algorithms are the Run length, Huffman coding, Shannon-Fano coding and LZW.

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[7]. 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[8]. 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 Lempel-Ziv-Welch

The Lempel-Ziv-Welch(LZW) algorithm was created in 1984 by Terry Welch. It removes redundant characters in the output and includes every character in the dictionary before starting compression and employs other techniques to improve compression[5]. The LZW algorithm stores strings in a "dictionary" with entries for 4,096 variable length strings. The 255 entries are used to contain the values for individual bytes, so the actual first string index is 256[9].

LZW Encoding Algorithm

Step 1: Initialize dictionary to contain entry for each byte. Initialize the encoded string with the first byte of the input stream.
Step 2: Read the next byte from the input stream.
Step 3: If the byte is an EOF go to step 6.
Step4: If concatenating the byte to the encoded string produces a string that is the dictionary:Concatenate the byte to the encoded string.
go to step 2
Step 5: If concatenating the byte to the encoded string produces a string that is not in the dictionary:
add the new string to the dictionary.Write the code for the encoded string to the output stream.Set the encoded string equal to the new byte.
go to step 2.
Step 6: Write out code for encoded string and text.

Example of LZW encoding

Input: LOSSY LOSSLESS = 14 character
Uncompressed bit = 14 X 8 = 112 bits

Table 1. Encoding for the LZW

input	next	output	Index	symbol
Nil	L			
L	O	L	256	LO
O	S	O	257	OS
S	S	S	258	SS
S	Y	S	259	SY
Y	ƀ	Y	260	Y ƀ
ƀ	L	ƀ	261	ƀ L
L	O			
LO	S	256	262	LOS
S	S			
SS	L	258	263	SSL
L	E	L	264	LE
E	S	E	265	ES
S	S			
SS	EOF	258		

Output =LOSSYƀLE
Output = 8 character
Compressed bit = 8 X 8 = 64 bits

3.2 Huffman Coding

The Huffman coding algorithm named after its inventor, David Huffman, who developed the method as a student in a class on information theory at MIT in 1950.

Huffman code procedure is based on the two observations. More frequently occurred symbols will have shorter code words than symbol that occur less frequently. The two symbols that occur less frequently will have the same length. The Huffman code is designed by merging the lowest probable symbols and this process is repeated until only two probabilities of two compound symbols are left and thus a code tree is generated and Huffman codes are obtained from labelling of the code tree[10].

Huffman Encoding Algorithm

Step 1. Initialization: Put all nodes in an sorted list, keep it sorted at all times

Step 2. Repeat the following steps until the sorted list has only one node left:

(a) From the list pick two nodes having the lowest frequencies/probabilities, create a parent node of them.

(b) Assign the sum of the children's frequencies/probabilities to the parent node and insert it into list such that the order is maintained.

(c) Delete the children from the sorted list.

Step 3. Assign code 0, 1 to the two branches of the tree on the path from the root.

After the Huffman tree, it creates a prefix code for each node from the alphabet by traversing the tree from the root to the node. It creates 0 for left node and 1 for a right node.

Example for Huffman coding:

Input: LOSSY LOSSLESS = 14 character
Uncompressed bit = 14 X 8 = 112 bits

Table 2. Frequencies in a character file

Symbol	S	L	O	Y	E	ƀ
Frequency	6	3	2	1	1	1

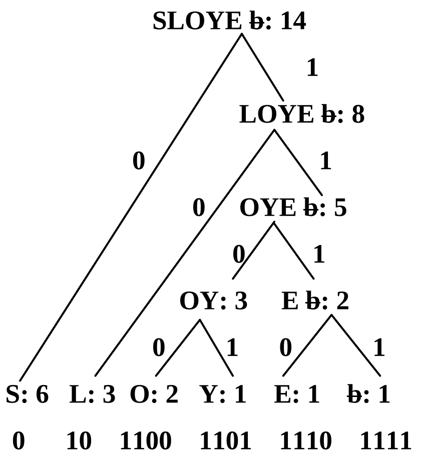

Fig.2 Encoding for the Huffman coding

Compressed bit = 32 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, a binary tree is generated that represent probabilities of each symbol occurring. The symbols are ordered in a way such that the most frequent symbol appear at the top of the tree and the least likely symbols appear at the bottom[11].

Shannon-Fano coding algorithm

Step 1: For a given list of symbols, develop a corresponding list of probabilities or frequency counts so that each symbols relative frequency of occurrence is known.

Step 2: Sort the list of symbols according to frequency, with the total frequency with the most frequently occurring symbols at the left and the least common at the right.

Step 3: Divide the list into two parts with the total frequency counts of the left part being as close to the total of the right as possible.

Step 4: The left part of the list is assigned the binary digit 0, and the right part is assigned the digit 1. This means that the codes for the symbols in the first part will all start with 0, and the codes in the second part will all start with 1.

Step 5: Recursively apply the step 3 and 4 to each of the two halves, subdividing groups and adding bits to the codes until each symbol has become a corresponding code leaf on the tree.

Example for Shannon-Fano coding:

Input: LOSSY LOSSLESS = 14 character
Uncompressed bit = 14 X 8 = 112 bits

Table 3. Frequencies in a character file

Symbol	S	L	O	Y	E	ƀ
Frequency	6	3	2	1	1	1

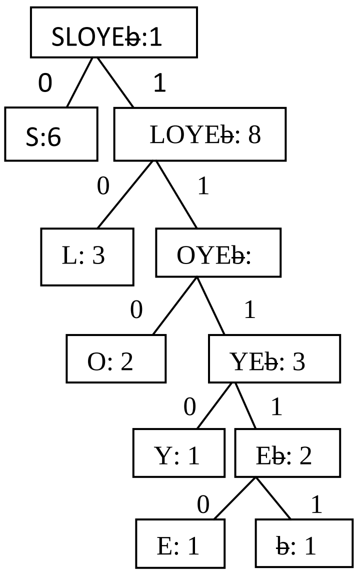

Fig. 3 Encoding for the Shannon-Fano encoding

Table 4. Output of Shannon-Fano Encoding

Symbol	**Count**	**Log(1/p)**	**Code**	**Subtotal**
S	6	1.22	0	6
L	3	2.22	10	6
O	2	2.81	110	4
Y	1	3.81	1110	4
E	1	3.81	11110	5
ƀ	1	3.81	11111	5
		Total # of bits		32

Compressed Bit = 32 bits

4. Measuring Compression Performance

Performance measure is use to find which technique is good according to some criteria. Depending on the nature of application there are various criteria to measure the performance of compression algorithm. When measuring the performance one of the main thing's to be considered is space efficiency and the time efficiency. Since the compression behaviours 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 and structure of input source. The compression behaviour depends on the category of the compression algorithm: lossy or lossless. Following are some measurements use to calculate the performance of lossless algorithms.

Compression ratio: Compression ratio is defined as the ratio of size of the compressed file and the size of the source file.

$$\textit{Compression ratio} = \frac{\textit{Size after compression}}{\textit{Size before compression}}$$

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

$$\textit{Compression factor} = \frac{\textit{Size before compression}}{\textit{Size after compression}}$$

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

$$\textit{saving percentage} = \frac{\textit{size before compression} - \textit{size after compression}}{\textit{size before compression}}\%$$

Table 5. Comparison between LZW, Huffman Coding and Shannon-Fano coding

Algorithm	**LZW**	**Huffman coding**	**Shannon -Fano coding**
Input size (Uncompressed bit)	112	112	112
Output size (Compressed bit)	64	32	32
Compression Ratio (In %)	57.14	28.5	28.5
Compression Factor	1.75	3.5	3.5
Saving Percentage (In %)	75	25	25

5. Conclusion

In this paper, we compare LZW, Huffman coding and Shannon-Fano coding techniques of data compression on English words in terms of compression size, compression ratio and saving percentage. After testing those algorithms, Huffman coding and Shannon-Fano coding methodologies are very powerful over LZW. Huffman coding and Shannon-Fano coding gives better results and reduces the size of the text.

6. REFERENCES

[1] K. Rastogi, K. Segar, "Analysis and performance Comparison of Lossless Compression Techniques for Text Data", International Journal of Engineering and Computer Research (IJETCR) 2(1) 2014,16-19.

[2] ShruthiPorwal, Yashi Chaudhary, Jitendra Joshi, Manish Jain "Data Compression Methodologies for Lossless Data and Comparison between Algorithms", International Journal of Engineering Science and Innovative Technology, Volume 2, Issue 2, March 2013.

[3] Introduction to Data Compression, Khalid Sayood, Ed Fox (Editor), March 2000.

[4] Amandeep Singh Sidhu, Er. MeenakshiGarg, "Research paper on Text Data Compression Algorithm using Hybrid Approach", International Journal of Computer Science and Mobile Computing, Vol. 3, Issue. 12, December 2014, Pg.01-10.

[5] Neha Sharma, Jasmet Kaur, Navmeet Kaur, " A Review on various Lossless Text Data Compression Techniques", An InternationaJournal Engineering Sciences, Vol. 2, Issue December 2014.

[6] Rajinder Kaur, Mrs. Monica Goyal, " A Survey on the different text data compression techniques", International Journal of Advanced Research in Computer Engineering & Technology, Volume 2, Issue 2, February 2013.

[7] www.rfwireless_world.com

[8] http://en.wikibooks.org

[9] Mohammed Al-laham&Ibrahiem, Proceedings of the world Congress on Engineering and Computer Science 2007 WCECS 2007, October 24-26, 2007, San Fancisco, USA

[10] Sahikal, Melvin.Y.,Arunodhayan Sam Solomon, M.N.Nachappa, "A Survey on Compression Techniques", International Journal of Recent Technology and Engineering (IJRTE), Volume 2, Issue 1, March 2013.

[11] Amarjit Kaur, Navdeep Singh Sethi, Harinderpal Singh, "A Review on Data Compression Techniques", International Journal of Advanced Research in Computer Science and Software Engineering, Volume 5, Issue 1, January 2015.

[12] Michael Heggeseth, Compression Algorithms: Huffman and LZW, CS 372: Data Structures, December 15,2003http://www.stolaf.edu/people/heggeset/compression/

[13] Lenat, doug, Lempel-Ziv compression, 1999 http://foldoc.doc.ic.ac.uk/foldoc/foldoc.cgi?Lempel-Ziv+compression

[14] HaroonAltarawneh, Mohammed altatawneh, "Data compression Techniques on Text Files: A Comparison Study", International Journal of Computer Applications, Volume 26-No.5, July 2011.

[15] A.D. Suarjaya, "A new algorithm for data compression Optimization" International Journal of Advanced Computer science Applications, Vol.3, No. 8, 2012.

K A Ramya received her MSc. Computer Science in 2011 from Anna University. She is pursuing her M.Phil Computer Science under the supervision of Ms. M.Pushpa in Quaid-E-Millath Government College for Women, Affiliated to University of Madras. She has presented papers in national and international conferences and published a paper in national level Journal. Her area of interest is Database Management System

M. Pushpa received her Post Graduation in Computer Applications from the University of Madras, Chennai and M.Phil. Degree in Computer Science from Mother Theresa's Women's University, Kodaikannal, Tamilnadu, and she has more than fifteen years of Teaching experience with both Undergraduate and Postgraduate Degrees Courses in Computer Science and Computer Application, with University of Madras. She had presented more than fifteen papers in national and international conferences and also published more than ten papers in National and International level Journal. Her area of interest is Data Mining and Artificial Intelligence.