A Review on Data Compression Techniques

Er. Mangi Lal [1], Er. Sammah Rasheed [2]

[1] *Assistant Professor, Department of Computer science & Engineering, SWIET, Ajmer Rajasthan., India*
[2]*Assistant Professor, Department of Computer science & Engineering, SWIET, Ajmer Rajasthan., India*

ABSTRACT

This review paper provides lossless data compression techniques and comparison between them. Data Compression is a process which reduces the size of data removing excessive information from it. It reduces the redundancy in data representation to decrease the storage required for that data and thus also reduces the communication cost by using the available bandwidth effectively. Data compression is important application in the area of file storage and distributed system. For different data formats like text, audio, video and image files there are different data compression techniques. Mainly there are two forms of data compression: - Lossy and Lossless. But in the lossless data compression, the integrity of data is to be preserved.

Keywords: *Data Compression, Huffman Coding, Shannon-Fano Coding, LZ Algorithm.*

I. INTRODUCTION:

Data compression has important application in the areas of data transmission and data storage. Many data processing applications are required for storage of large volumes of data, and the number of such applications is constantly increasing as the use of computers extends to new disciplines. At the same time, the proliferation of computer communication networks is resulting in massive transfer of data over communication links. Compressed data reduces storage and/or communication costs. When the amount of data to be transmitted is reduced, it effects increasing the capacity of communication channel. Similarly, compressing a file to half of its original size is equivalent to doubling the capacity of the storage medium. It may then become feasible to store the data at a higher, thus faster, level of the storage, hierarchy and reduce the load on the input/output channels of the computer system.

Data compression is necessary for both electronic storage and data transport to save storage space, conserve bandwidth and speed up communication. In electronic storage, a body of data is compressed before it is stored on some digital storage device, for example, a computer disk or tape. This process allows more data to be placed on a given device. When data is retrieved from the device, it is decompressed and In Data communications, A sender can compress data before transmitting it through communication channel and the receiver can decompress the data after receiving it.

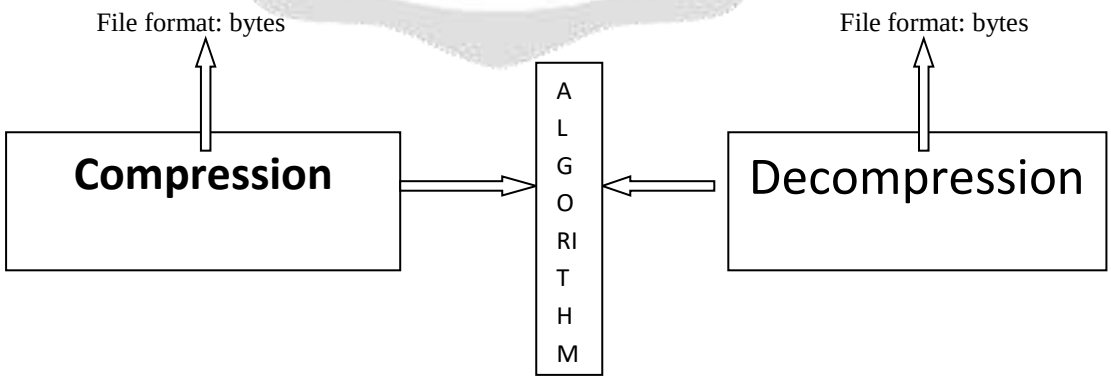


Fig 1: Data compression

II. TYPES OF DATA COMPRESSION:

There are two categories for compression of data: lossy and lossless. While each uses different techniques to compress files, both have the same aim. To look for duplicate data in the graphic and use a much more compact data representation.

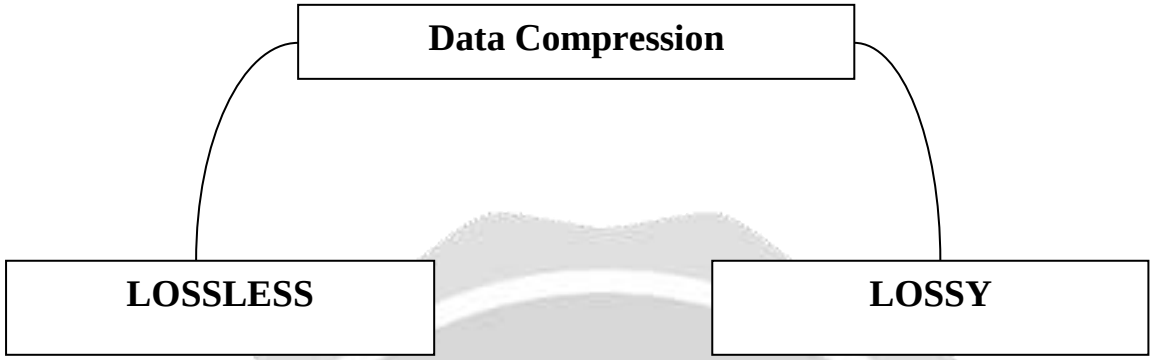


Fig 2: types of data compression

1. LOSSY COMPRESSION

A) Lossless Compression: In this compression technique, no data is lost. The exact replica of the original file can be retrieved by decrypting the encrypted file. Text compression is generally of lossless type. In this type of compression generally the encrypted file is used for storing or transmitting data. For general purpose use we need to decrypt the file.

B) Lossy Compression: Lossy Compression is generally used for image, audio, video. In this compression technique, the compression process ignores some less important data and the exact replica of the original file can't be retrieved from the compressed file. To decompress the compressed data we can get a closer approximation of the original file.

In lossy image compression, the reconstructed image after compression is approximation of the original image. A lossy compression method is termed visually lossless when the loss of information caused by compression method is invisible for an observer. In general, a lossy compression is implemented using spatial domain encoding and transform spatial domain encoding methods

III. LOSSLESS COMPRESSION TECHNIQES

A). Huffman Compression: Huffman coding is used for lossless data compression. It uses variable length code for encoding a source symbol (such as a character in a file) which is derived based on the estimated probability of occurrence for each possible value of the source symbol. In this compression technique, a table is created incorporating the no of occurrences of an individual symbol. This table is known as frequency table and is arranged in a certain order. Then a tree is generated from that table, in this tree high frequency symbols are assigned codes which have fewer bits, and less frequent symbols are assigned codes with many bits. In this way the code table is generated. The following example bases on a data source using a set of five different symbols. The symbol's frequencies are:

Table: 1

SYMBOL	FREQUENCY
A	24
B	12
C	10
D	08
E	08
TOTAL 186 BIT(WITH 3 BIT PER CODE WORD)	

The two rarest symbols 'E' and 'D' are connected first, followed by 'C' and 'D'. The new parent nodes have the frequency 16 and 22 respectively and are brought together in the next step. The resulting node and the remaining symbol 'A' are subordinated to the root node that is created in a final step.

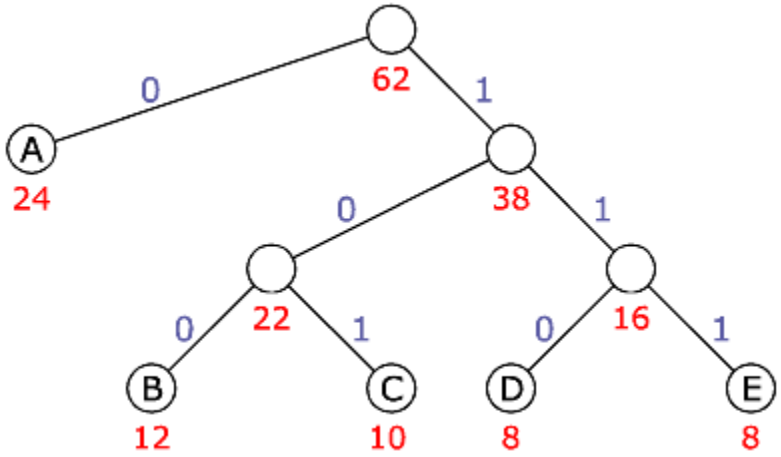


Fig 3: Huffman Tree

Table: 2

SYMBOL	FREQUENCY	CODE	CODE LENGTH	TOTAL LENGTH
A	24	0	1	24
B	12	100	3	36
C	10	101	3	30
D	08	110	3	24
E	08	111	3	24
GES. 186 BIT (3 BIT COD BIT			TOTAL 138	

B) **Shannon – Fano Code**

Shannon–Fano coding, named after Claude Elwood Shannon and Robert Fano, is a technique for constructing a prefix code based on a set of symbols and their probabilities. It is suboptimal in the sense that it does not achieve the lowest possible expected codeword length like Huffman coding; however unlike Huffman coding, it does guarantee that all codeword lengths are within one bit of their theoretical ideal $I(x) = -\log P(x)$.

In Shannon–Fano coding, the symbols are arranged in order from most probable to least probable, and then divided into two sets whose total probabilities are as close as possible to being equal. All symbols then have the first digits of their codes assigned; symbols in the first set receive "0" and symbols in the second set receive "1". As long as any sets with more than one member remain, the same process is repeated on those sets, to determine successive digits of their codes. When a set has been reduced to one symbol, of course, this means the symbol's code is complete and will not form the prefix of any other symbol's code.

The algorithm works, and it produces fairly efficient variable-length encodings; when the two smaller sets produced by a partitioning are in fact of equal probability, the one bit of information used to distinguish them is used most efficiently. Unfortunately, Shannon–Fano does not always produce optimal prefix codes.

1. **Shannon-Fano Algorithm:**

A Shannon–Fano tree is built according to a specification designed to define an effective code table. The actual algorithm is simple:

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

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

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.

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.

5. Recursively apply the steps 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 1:

The source of information A generates the symbols {A0, A1, A2, A3 and A4} with the corresponding probabilities {0.4, 0.3, 0.15, 0.1 and 0.05}. Encoding the source symbols using binary encoder and Shannon-Fano encoder gives:

Table: 3

Source Symbol	**Pi**	**Binary Code**	**Shannon-Fano**
A0	0.4	000	0
A1	0.3	001	10
A2	0.15	010	110
A3	0.1	011	1110
A4	0.05	100	1111
Lavg	**H = 2.0087**	**3**	**2.05**
Source Symbol	**Pi**	**Binary Code**	**Shannon-Fano**

The Entropy of the source is

$$H = -\sum_{i=0}^{4} Pi \log_2 Pi = 2.0087 \text{ bit/symbol}$$

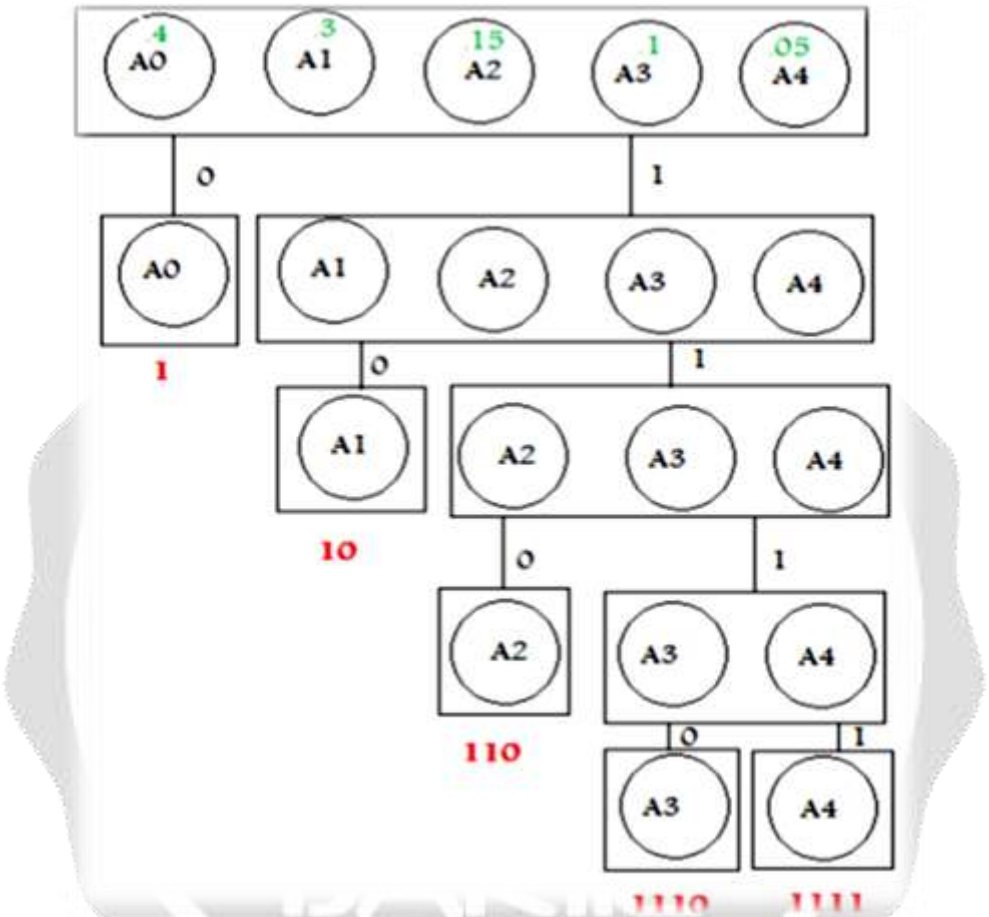
Since we have 5 symbols (5<8=23), we need 3 bits at least to represent each symbol in binary (fixed-length code). Hence the average length of the binary code is

$$Lavg = \sum_{i=0}^{4} Pi\, li = 3\,(0.4 + 0.3 + 0.15 + 0.1 + 0.05) = 3 \text{ bit/symbol}$$

Thus the efficiency of the binary code is

$$\eta = \frac{H}{Lavg} = \frac{2.0087}{3} = 67\%$$

Shannon-Fano code is a top-down approach. Constructing the code tree, we get



The average length of the Shannon-Fano code is

$$Lavg = \sum_{i=0}^{4} Pi\ li = 0.4 * 1 + 0.3 * 2 + 0.15 * 3 + 0.1 * 4 + 0.05 * 4 = 2.05\ bit/symbol$$

Thus the efficiency of the Shannon-Fano code is

$$\eta = \frac{H}{Lavg} = \frac{2.0087}{2.05} = 98\%$$

This example demonstrates that the efficiency of the Shannon-Fano encoder is much higher than that of the binary encoder.

c) Lempel ziv algorithms:

The Lempel Ziv Algorithm is an algorithm for lossless data compression. It is not a single algorithm, but stems from algorithms proposed by Jacob Ziv and Abraham Lempel in their landmark papers in 1977 and 1978 is shown in Figure 5.

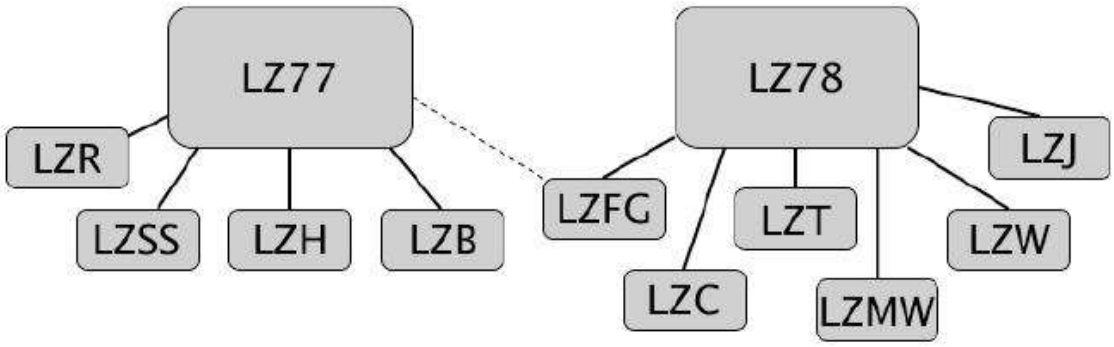


Figure 4: Classification of lempel ziv family

1) LZW:

Terry Welch has presented his LZW (Lempel–Ziv– Welch) algorithm (Welch 1984), which is based on LZ78. It basically applies the LZSS principle of not explicitly transmitting the next non-matching symbol to LZ78 algorithm. The dictionary has to be initialized with all possible symbols from the input alphabet. It guarantees that a match will always be found.

LZW would only send the index to the dictionary. The input to the encoder is accumulated in a pattern 'w' as long as 'w' is contained in the dictionary. If the addition of another letter 'K' results in a pattern 'w*K' that is not in the dictionary, then the index of 'w' is transmitted to the receiver, the pattern 'w*K' is added to the dictionary and another pattern is started with the letter 'K'.

2) Encoding Algorithm:

1. Initialize the dictionary to contain all blocks of length one (W= {a,b}).

2. Search for the longest block **W** which has appeared in the dictionary.

3. Encode **W** by its index in the dictionary shown in Table 3.4.

4. Add **W** followed by the first symbol of the next block to the dictionary.

5. Go to Step 2.

The following example illustrates how the encoding is performed.

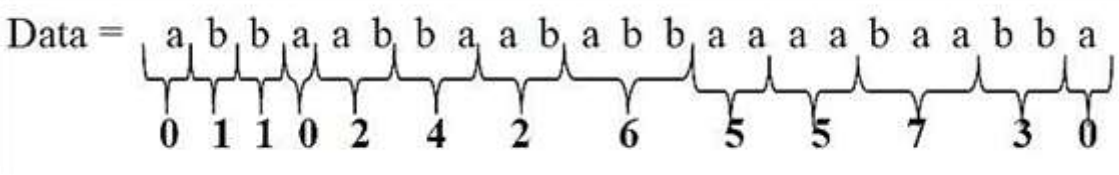


Table 4 Dictionary for lembel ziv algorithm

Dictionary			
Index	**Entry**	**Index**	**Entry**
0	a	7	Baa

1	b	8	Aba
2	ab	9	Abba
3	bb	10	Aaa
4	ba	11	Aab
5	aa	12	Baab
6	abb	13	bba

3) Advantages of Data Compression

(i) It reduces the data storage

(ii) The user can experience rich quality signals for data representation

(iii) Enhanced Data security can also be achieved by proper encryption methods.

(iv) The rate of input – output operations in a computing device can be greatly increased due to shorter presentation of data.

(v) Data Compression also reduces the cost of backup and recovery of data in computer systems by storing the backup of large database files in compressed format.

4) Disadvantages of Data Compression

(i) Extra overhead applied in encoding and decoding process is one of the most serious drawbacks of data compression, which discourages its use in some areas.

(ii) Reliability of the records gets reduced by Data compression.

(iii) Compressed, sensitive data transmitted through a noisy communication channel is risky because the burst errors introduced by the noisy channel can destroy the transmitted data.

(iv)Disorder of data properties of a compressed data will result in compressed data different from the original data.

(v) In many hardware and system implementation, the extra complexity added by data compression can increase cost of the system reduce its efficiency.

IV Future scope of data compression:

In the Our review paper's most significant long term goal is to examine Unity's entire algorithm stack and determine how to break down algorithm barriers that are preventing us from getting the best possible compression solutions. By studying and discussing all the techniques we find lossy compression techniques provides high compression ratio than lossless compression scheme.

Conclusion

Data compression is a topic of great importance in many applications. The methods of data compression have been studied for almost four decades. This paper provided overview of the various data compression methods both lossless and lossy of general utility. These algorithms are evaluated in terms of the amount of compression they provide, efficiency of algorithm and the susceptibility to error.

References:

1. Ruchi Gupta1, Mukesh Kumar2 and Rohit Bathla3,"Data Compression - Lossless and Lossy Techniques" International Journal of Application or Innovation in Engineering & Management (IJAIEM), Volume 5, Issue 7, July 2016 ISSN 2319 – 4847.

2. Prof. Dipti Mathpal and Prof. Mittal Darji," A Research Paper on Lossless Data Compression Techniques", IJIRST –International Journal for Innovative Research in Science & Technology| Volume 4 | Issue 1 | June 2017,ISSN (online): 2349-6010.
3. Introduction to Data Compression, Khalid Sayood, Ed Fox (Editor), March 2000.
4. Ken Huffman. Profile: David A. Huffman, Scientific American, September 1991, pp. 54–58.
5. Cormak, V. and S. Horspool, 1987.Data compression using dynamic Markov modeling, Comput. J., 30: 541–550.
6. Cleary, J., Witten, I., "Data Compression Using Adaptive Coding and Partial String Matching", IEEE Transactions on Communications, Vol. COM-32, No. 4, April 1984, pp 396-402.
7. Mahoney, M., "Adaptive Weighting of Context Models for Lossless Data Compression", Unknown, 2002.
8. P. Kumar and A.K Varshney, " Double Huffman Coding " International Journal of Advanced Research in Computer Science and Software Engineering (IJARCSSE) Volume 2, Issue 8, August 2012 S.Shanmugasundaram and R. Lourdusamy, " IIDBE: A Lossless Text Transform for Better Compression " International Journal of Wisdom Based Computing,Vol. 1 (2), August 2011.
9. 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.
10. 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.
11. Introduction to Data Compression, Khalid Sayood, Ed Fox (Editor), March 2000.
12. Haroon A, Mohammed A "Data Compression Techniques on Text Files: A Comparison Study" International Journal of Computer Applications (0975 –8887) Volume 26– No.5.
13. Mamta Sharma "Compression Using Huffman Coding". IJCSNS International Journal of Computer Science and Network Security, VOL.10 No.5, May 2010.