International Journal of Application or Innovation in Engineering & Management (IJAIEM)
Web Site: www.ijaiem.org Email: editor@ijaiem.org
Volume 5, Issue 7, July 2016 **ISSN 2319 - 4847**

Data Compression - Lossless and Lossy Techniques

Ruchi Gupta[1], Mukesh Kumar[2] , Rohit Bathla[3]

[1,2,3]A.P. in Department of Computer Science & Engineering. JMIETI, Radaur

ABSTRACT

Compression reduces the number of bits required to represent the data. Compression is useful because it helps in reducing the consumption of expensive resources, such as disk space and transmission bandwidth. Compression is built into a broad range of technologies like storage systems, databases operating systems and software applications. Hence selection of data compression algorithm should be appropriate. This paper presents different data compression methodologies. Mainly there are two forms of data compression :- Lossless and Lossy. In this paper, we discussed about some of the Lossless and Lossy data compression methods.

Keywords: Data Compression, Lossless compression, Lossy compression.

1. INTRODUCTION

Data compression refers to the process of converting an input stream (original raw data) into another output stream (compressed stream) that has a smaller size. A stream can be either a file or a buffer in memory. Data compression can also be defined as the science of representing information in a compact form. These compact representations can be created by identifying and using structures that exist in the data. Data can be in the form of characters in the text file, numbers that are samples of speech or image waveforms, or sequences of numbers that are generated by other processes. Data compression is required because more and more of the information that we generate and use is in digital form i.e. in the form of numbers represented by bytes of data and the number of bytes required to represent multimedia data can be huge[8].

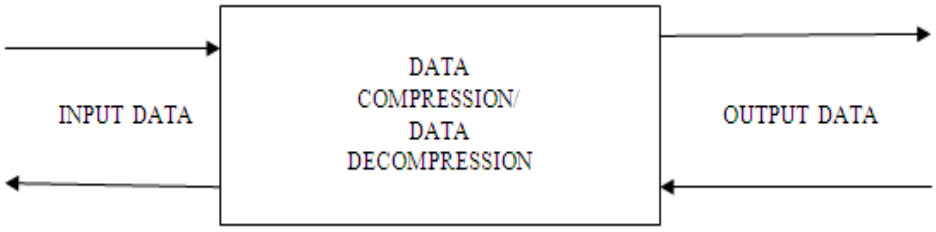

Fig.1.1 Data compression and decompression

There exists variety of techniques for data compression. All are based on different ideas and are suitable for different types of data. Different methods produce different results, but they are all based on the same basic principle i.e. they compress data by removing redundancy from the original data in the source file. There are different types of redundancies:

(1) Alphabetic Redundancy: For example, the letter 'E' appears very often, while 'Z' is rare. This is called alphabetic redundancy, and suggests assigning variable size codes to the letters with 'E' getting the shortest code and 'Z' the longest one.

(2) Contextual Redundancy: For example, the letter 'Q' is almost always followed by the letter 'U'.

(3) Image Redundancy: For example, in a non-random image, adjacent pixels tend to have similar colors.

Variable-size code has less redundancy than a fixed-size code (or no redundancy at all). Fixed size codes make it easier to work with test, so they are useful, but they are redundant.

The general law of data compression is to "assign short codes to common events (symbols or phrases) and long codes to rare events".[8]

2. TYPES OF COMPRESSION

Compression can be of two types:

- Lossless Compression
- Lossy Compression

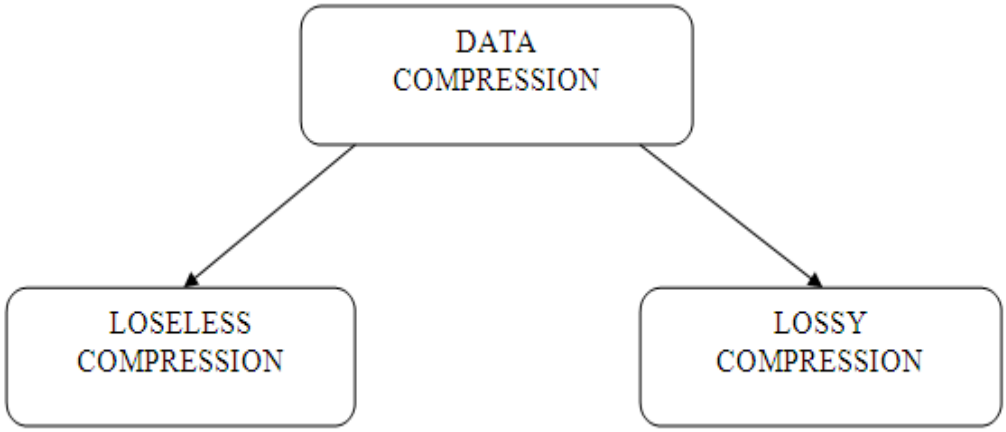

Fig.1.2 Types of Data Compression

2.1) 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.

2.2) 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.[3]

3. TERMS ASSOCIATED WITH DATA COMPRESSION

3.1 Compressor or Encoder: It is the program that compresses the raw data in the input stream and creates an output stream with compressed (low redundancy data).

Fig. 1.3 Compressor or Encoder

3.2 Decompressor or Decoder: It is the program that converts the compressed data into the original data.

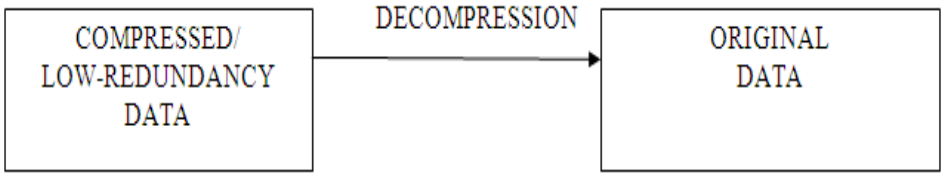

Fig. 1.4 Decompressor or Decoder

4. Measurement Parameters

The measurement parameter can differ and depends on the use of the compressed file. Performance of the compression algorithm largely depends on the redundancy on the source data. The different measurement parameters are as follows:

4.1 Compression Ratio: It is defined as the ratio between the compressed file and the original file.
Compression Ratio=*Compressed file size*/*Original File size*.

4.2 Compression Factor: It is defined as the ratio between the original file and the compressed file and is the inverse of the Compression Ratio.

Compression Factor=1/*Compression Ratio* OR *Original File size*/*Compressed file size* [3]

4.3) Compression Gain: It is defined as
Compression Gain= 100 loge (reference size) / compressed size)
where the reference size is either the size of the input stream or the size of the compressed stream produced by some standard lossless compression method[27].

4.4) Saving Percentage: It refers to the percentage of the size reduction of the file, after the compression.
Saving Percentage=*Original file size*−*Compressed file size*/*Original file size*(%)

4.5) Compression Time: It is defined as the amount of time taken by the algorithm to compress the file. It is calculated in milliseconds (ms).

4.6) Decompression Time: It is defined as the amount of time taken by the algorithm to decompress and retrieve the original file from compressed file. It is also calculated in milliseconds. The compression time and decompression time is important in case of the applications where the algorithms are used to transmit the data, or to store the data in a secondary storage and retrieve it as required.[3]

5. Lossless Compression Techniques

In this section, we will give a short review and explanation for each one of the lossless compression methods that can be used on any text files. Compression algorithms have a long history, the roots of compression algorithms goes back to earlier twentieth century. [4]

5.1 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[3].
The following example bases on a data source using a set of five different symbols. The symbol's frequencies are: [10]

Table 1.1

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.

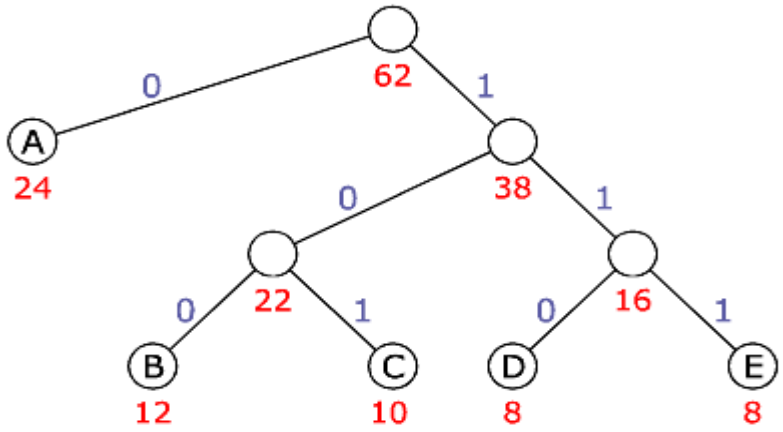

Fig. 1.5 Huffman Tree

Table 1.2

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 CODE)				TOTAL 138 BIT

5.2 Shannon Fano Compression : This technique is named after Claude Shannon and Robert Fano and is a variable length code for encoding a source symbol. It is a lossless data compression scheme. According to Shannon's source coding theorem, the optimal code length for a symbol is –log b P, where b is the number of symbols used to make output codes and P is the probability of the input symbol. Similar to the Huffman coding, initially a frequency table is generated and then a particular procedure is followed to produce the code table from frequency.[3]
The following example illustrates the compression algorithm: [10]
To create a code tree according to Shannon and Fano an ordered table is required which provides the frequency of any symbol. Each part of the table will be divided into two segments. The algorithm has to ensure that either the upper and the lower part of the segment have nearly the same sum of frequencies. This procedure will be repeated until only single symbols are left.

Table 1.3

SYMBOL	**FREQUENCY**	**CODE LENGTH**	**CODE**	**TOTAL LENGTH**
A	24	2	00	48
B	12	2	01	24
C	10	2	10	20
D	08	3	110	24
E	08	3	111	24
TOTAL: 62 SYMBOLS		SF CODED: 140 BIT		LINEAR (3 BIT/SYMBOL): 186 BIT

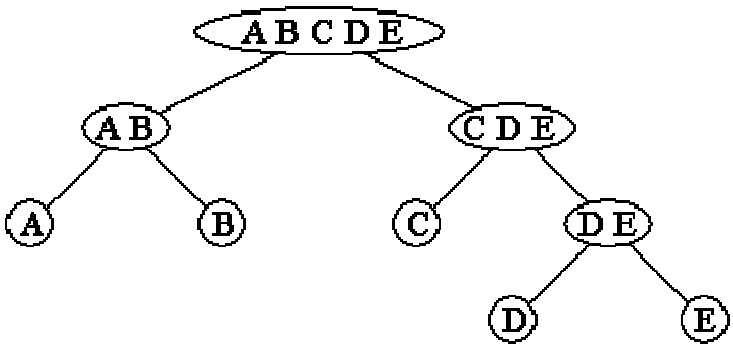

Fig. 1.6 Shannon Fano Compression

The original data can be coded with an average length of 2.26 bit. Linear coding of 5 symbols would require 3 bit per symbol. But, before generating a Shannon-Fano code tree the table must be known or it must be derived from preceding data.

5.3 LZW Compression: This technique is named after Abraham Lempel , Jacob Zev and Terry Welch. It is dictionary coder or substitution coder, which means a dynamic dictionary is created depending upon the presence of substring chosen from the original file. Then the substring is matched with the Dictionary, if the string is found then a reference of the dictionary is mentioned in the encoded file, if the string is not found then a new dictionary entry is made with a new reference. In all algorithms the encoded file contains the code table/ Dictionary and the encoded text; the encoder matches the codes with the directory (code table/ dictionary) and retrieves the original text iteratively.[3]

5.4 Run Length Encoding: Run Length Encoding (RLE) is a simple and popular data compression algorithm. It is based on the idea of replacing a long sequence of the same symbol by a shorter sequence. RLE requires only a small amount of hardware and software resources. Run-length encoding is a data compression algorithm that is supported by most bitmap file formats, such as TIFF, BMP, and PCX[2]. [9]
Consider a character run of 15 'A' characters which normally would require 15 bytes to store :

```
AAAAAAAAAAAAAAA
```

This can be compressed to the form:

```
15A
```

With RLE, this would only require two bytes to store, the count (15) is stored as the first byte and the symbol (A) as the second byte [10].

5..5 Arithmetic Coding: Arithmetic coding is the method of replacing each bit with a codeword. So it replaces a string of input data with a single floating point number as an output. The main purpose of this technique is to given an interval to each potential bit data. Arithmetic coding can treat the whole string data as one unit. A message is represented by a half-open interval [a, b) where a and b are real numbers between 0 and 1. Initially, the interval is [0, 1). When the message becomes longer, the length of the interval shorts and the number of bits needed to represent the interval increases [5].

5.6 Bit Reduction Algorithm: This algorithm was originally implemented for use in an SMS application and using this algorithm, about 256 characters per message (typically 160 characters per message) through the same 7-bit GSM network could be send. The idea is that this program reduces the standard 7-bit encoding to some application specific 5-bit encoding system and then pack into a byte array. This method will reduce the size of a string considerably when the string is lengthy and the compression ratio is not affected by the content of the string [5].

5.7 J-bit encoding(JBE) : It manipulates each bit of data inside file to minimize the size without losing any data after decoding. This basic algorithm is intended to be combined with other data compression algorithms in order to optimize the compression ratio. The performance of this algorithm is measured by comparing combination of different data compression algorithms. The main idea of this algorithm is to split the input data into two data where the first data will contain original non-zero byte and the second data will contain bit value explaining position of non-zero and zero bytes. Both of the data can be separately compressed with other data compression algorithm to achieve maximum compression ratio [6].

5.8 Two phase encoding: This technique compresses the sorted data more efficiently. It provides a way to enhance the compression technique by merging RLE compression algorithm and incremental compression algorithm. In first phase the data is compressed by applying RLE algorithm that compresses the frequent occur data bits by short bits. In the second phase incremental compression algorithm stores the prefix of previous symbol from the current symbol and replaces with integer value. This technique can reduce the size of sorted data by 50% using two phase encoding technique.[11]

5.9 Modified Huffman Data Compression algorithm: It works in three phases in order to compress the text data. In the first phase, data is compressed with the help of dynamic bit reduction technique and in second phase unique words are to be found to compress the data further and in third and final phase Huffman coding is used to compress the data further to produce the final output.[2]

5.10 Adaptive Huffman coding: We use Huffman coding for data compression but the limitation of Huffman coding is, to send the probability table with the compress information, because without the probability table decoding is not possible. To remove this disadvantage in Huffman coding, the adaptive Huffman coding was developed. This table requires the addition of 0 an extra bytes to the output table, and consequently it usually doesn't make much difference in the compression ratio.[7]

6. Lossy Compression Techniques

6.1 Transform coding: Transform coding is a type of data compression for natural data like audio signals or images. This technique is typically lossy, resulting in a lower quality copy of the original input. In transform coding , knowledge of the application is used to choose information to discard, thereby lowering its band width. The remaining information can be compressed via number of methods, when the output is decoded, the result may not be identical to the original input, but is expected to be close enough for the purpose of the application.[1]

6.2 Discrete Cosine Transform (DCT): A discrete expresses a finite sequence of data points in the terms of the sum of cosine functions oscillating at different frequencies. DCT is a lossy compression technique which is widely used in area of image and audio compression. DCTs are used to convert data in the summation of series of cosine waves oscillating at different frequencies. There are very similar to Fourier transforms but DCT involves uses of cosine functions are much more efficient as fewer function are needed to approximate a signal. [1]

6.3 Discrete Wavelet Transform(DWT): The DWT is an implementation of the wavelet transform using a discrete set of the values scales and translations obeying some defined rules. In other words, this transform decomposes the signal into mutually orthogonal set of wavelets which is the main differences from the continuous wavelet transform or its implementation from the discrete time series sometimes called Discrete –time continuous wavelet transform (DTCWT). DWT is applied to the image block generated by the pre-processor .[1]

7. 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] P.Kavitha, "A Survey on Lossless and Lossy Data Compression Methods" International Journal of Computer Science & Engineering Technology (IJCSET), Vol. 7 No. 03 Mar 2016, ISSN : 2229-3345
[2] M.Kaur and U.Garg " Lossless Text Data Compression Algorithm Using Modified Huffman Algorithm", International Journal of Advanced Research in Computer Science and Software Engineering, Volume 5, Issue 7, July 2015, ISSN: 2277 128X
[3] A.K.Bhattacharjee, T.Bej and S.Agarwal, "Comparison Study of Lossless Data Compression Algorithms for Text Data", IOSR Journal of Computer Engineering (IOSR-JCE), Volume 11, Issue 6 (May. - Jun. 2013), PP 15-19, e-ISSN: 2278-0661, p- ISSN: 2278-8727, www.iosrjournals.org
[4] S.Kapoor, A.Chopra, "A Review of Lempel Ziv Compression Techniques", IJCST Vol. 4, Issue 2, April - June 2013, ISSN : 0976-8491 (Online) | ISSN : 2229-4333 (Print)
[5] R.S.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, Volume 3, Issue 3, March 2013, ISSN: 2277 128X
[6] N.Sharma, J.Kaur and N.Kaur, "A Review on various Lossless Text Data Compression Techniques", An International Journal of Engineering Sciences, Vol.2, Issue December 2014, ISSN: 2229-6913 (Print), ISSN: 2320-0332 (Online)
[7] S.Mishra and S.Singh, "A Survey Paper on Different Data Compression Techniques", Indian Journal of Applied Research, Vol.6, Issue 5, May 2016, ISSN - 2249-555X
[8] S.Khalid, "Introduction to Data Compression", 1996, 2000 by Academic Press
[9] D.Kaur and K.Kaur, "Analysis of Lossless Data Compression Techniques", International Journal of Computational Engineering Research, Vol. 03, Issue 4, April 2013
[10] http://www.binaryessence.com/dct/en000046.htm
[11] A.Singh and Y.Bhatnagar, "Enhancement of Data Compression Using Incremental Encoding", International Journal of Scientific & Engineering Research, Volume 3, Issue 5, May-2012 1 ISSN 2229-5518.