Survey of Lossless Data Compression Algorithms

Himali Patel
Dept of Information Technology
SVMIT Engineering College
Bharuch, India

Unnati Itwala
Dept of Information Technology
SVMIT Engineering College
Bharuch, India

Roshni Rana
Dept of Information Technology
SVMIT Engineering College
Bharuch, India

Kruti Dangarwala
Associate Professor
Dept of Information Technology
SVMIT Engineering College
Bharuch, India

Abstract:- **The main goal of data compression is to decrease redundancy in warehouse or communicated data, so growing effective data density. It is a common necessary for most of the applications. Data compression is very important relevancy in the area of file storage and distributed system just because of in distributed system data have to send from and to all system. Two configuration of data compression are there “lossy” and “lossless”. But in this paper we only focus on Lossless data compression techniques. In lossless data compression, the wholeness of data is preserved. Data compression is a technique that decreases the data size, removing the extreme information. Data compression has many types of techniques that decrease redundancy. The methods which mentioned are Run Length Encoding, Shannon Fanon, Huffman, Arithmetic, adaptive Huffman, LZ77, LZ78 and LZW with its performance.**

Keywords: Data Compression, Lossy compression, Lossless compression, Run Length Encoding,Huffman, Shannon Fano, Arithmetic, Lz77,Lz78, LZW.

I. INTRODUCTION

Compression is the art of representing information in a compact form rather than its original or uncompressed form. Data Compression can be configure as lossy or lossless [1]. Lossless compression techniques rebuild the original data from the compacted file without any loss of data. Lossless technique is used when the original data of a source are so important that we can’t provide to lose any details. Examples of similar sources data are images, text etc that preserved for authorized reason, some computer practicable files, etc. Data compression [2] is a method of encoding rules that allows essential reduction in total number of bits to store or transmit a file. The algorithm which decreases some part of data is called lossy data compression but in this paper we are go for lossless data compression technique. Data compression methods can be classified in many ways [3, 4]. This paper examines the performance of the Run Length Encoding Algorithm, Huffman Encoding Algorithm, Shannon Fano Algorithm, Adaptive Huffman Encoding Algorithm, Arithmetic Encoding Algorithm and Lempel Ziv Welch Algorithm.

II. TYPES OF DATA COMPRESSION

Data Compression methods are divided into two categories.

a. Lossless Compression

b. Lossy Compression

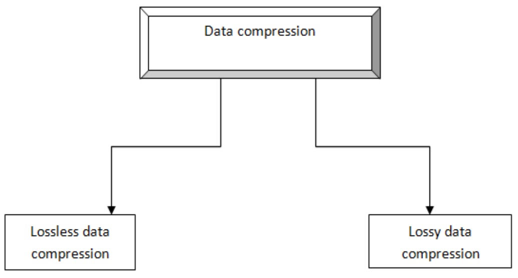

Fig. 1. Types of Data Compression Tehniques

A. Lossless Compression

Lossless technique is used when the original data of a source are so important that we can’t provide to lose any details. The main goal of this compression technique is to compress the file by decreasing the information in such a way that there is a no loss when decrypting any file back in to the original file [5, 8, 9]. Example of lossless data compression technique is text compression. the popular file format like ZIP file format that is used for compression of data files. It is an application of lossless data compression.

This Lossless data compression can be grouped into two categories [15],[16]:

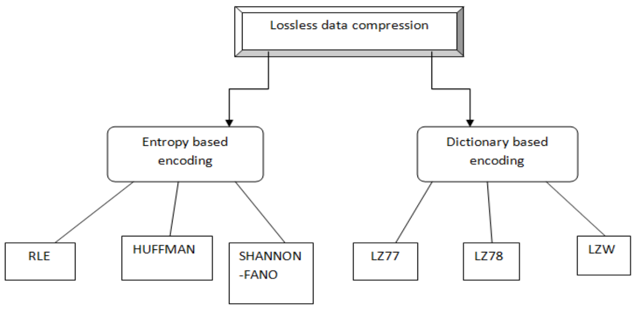

Fig. 2. Types of Lossless Compression

1) *Entropy based encoding*

This technique isn't dependent on definite characteristics of medium [6, 7]. In this method of data compression of the algorithm begins by first counting the frequency of every symbol according to its event in the file. For stated symbols of original file, these newly symbols are fixed and aren't dependent on the content of file. The new symbols length is variable and it varies on the frequency of include symbols of original file.

2) *Dictionary based encoding*

Dictionary based algorithms don't encode single symbols as variable length bit strings [10], they encode variable length strings of symbols as single tokens.The encoder finds the match pair of string in dictionary from original text and if this match is found it replaces with its regard in the dictionary.

B. *Lossy Compression*

The goal of lossy compression is not to regenerate an complete copy of the source information after decomposition but prior a version of it which is feel by the receptive as a true copy.This technique doesn't generate the exact original file for example, the same copy but it gives the output with some lost information. An original message can't be rebuilded by decoding a process. Thus, this technique can't be appropriate to textual data but it can be apply on video, audio, images etc [11].

III. COMPRESSION ALGORITHM

A. *Run Length Encoding*

Run Length Encoding or simply RLE is the simplest technique of data compression algorithms. The successive sequences of symbols are identified as runs and the others are as non runs in this algorithm. This algorithm uses those runs to compress the original source file while keeping all the non-runs with out using for the compression process. For example, the text "aaabbccccd" is examined as a source to compress, taken the first 3 letters as a non-run having a length 3, and the next 7 letters taken as a run having length 7, since symbol a is repeated.

B. *Shannon Fano*

This coding process developed to create a binary code tree by Claude E. Shannon and Robert M. Fano in 1960. Shannon Fano coding is very the easiest method for implement as compared to any other methods. Based upon their probabilities it encodes the messages. There is Different codes have different numbers of bits. The algorithm is described as below[17]:-

Step1:
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.

Step2:
Sort the list of symbols according to frequency of their,
with the most frequently occurring symbols at the top and the least common at the bottom.

Step3:
Divide the list into two parts,
with the total frequency counts of the upper half being as close to the total of the bottom half as possible.

Step4:
The upper half of the list is assigned the binary digit 0, and the lower half is assigned the digit 1. This means that the codes for the symbols in the first half will all start with 0, and the codes in the second half will all start with 1.

Step5**:**
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.

C. *Huffman Coding*

Huffman Coding is developed by David Huffman in 1950. Many times Huffman Coding performs better than the Shannon Fano Coding. In this coding the characters in a data file are converted in to binary code. Two types categories of Huffman Encoding have been proposed: Static Huffman Algorithm and Adaptive Huffman Algorithm. Static Huffman Algorithms first count the frequencies and then generate a common tree for the compression and decompression processes. Adaptive Huffman algorithms generate the tree while counting the frequencies and there will be two trees in both the processes [12, 13]. The algorithm are as follows [17]:-

The nodes are arranged in ascending order.

Step1:
Two nodes with the lowest frequency is located.
Step2:
The two least node is added and an internal node of this two node is created and the added sum of the two node is given as it's weight.
Step3:
The internal node is now added to the list and the two node as it's child.
Step4:
One of the child node is assigned 1 and other as 0 during coding.
Step5:
Previous steps are repeated till there is no other node left in the tree. The free node is root of the tree.

D. *Adaptive huffman encoding*

Adaptive Huffman coding was first generate independently by Faller in 1973 and Gallager in 1978. Knuth assisted improvements in the original algorithm in 1985 and the resulting algorithm is known as algorithm FGK. A more current version of this coding is described by Vitter in 1987 and called as algorithm V [17].

```
ENCODER
Initialize_model();
do {
c = getc( input );
encode( c, output );
update_model( c );
} while ( c != eof)

DECODER
Initialize_model();
while ( c = decode (input)) != eof) {
putc( c,output)
update_model( c );
}
```

E. Arithmatic Encoding:

The main goal of Arithmetic coding is to appoint an interval to each possible symbol. Then a decimal number is appointed to this interval. The algorithm begins with an interval of 0 and 1. After every input symbol from the alphabet is read, the interval is divided into a smaller interval in apposite to the input symbol's probability. This interval starts the new interval and it is divided into parts according to probability of symbols the input alphabet. This is repeated for every input symbol. And, at the end, any floating point number from the final interval terminate the input data.

F. LZ77

LZ77 developed by Jacob Ziv and Abraham Lempel in 1977. A dictionary is a part of the previos encoded sequence. The encoder search into the input sequence within a sliding window. This algorithm assume patterns in the input stream appear close together. Any pattern that resort over period longer than the search buffer size will not be captured. A better compression method would save frequently appearing patterns in the dictionary.

Encoding-Pseudo code algorithms are as follows[17]:-

```
Step1:
   Check look-ahead buffer is empty or not
Step2:
   If not empty, look for the longest match in search buffer.
Step3:
   If match is found print output as (offset from window boundary, length of match, next symbol in lookahead buffer) and shift window by length+1 else print output as (0,0,first symbol in look-ahead buffer) and shift window by 1.
Step4:
   Loop until the look ahead buffer is empty.
```

G. LZ78

This compression algorithm developed by Jacob Ziv and Abraham Lempel in 1978. LZ78 inserts one- or multi-character, non-overlapping, clear patterns of the message to be encoded in a Dictionary [15]. It keeps an open dictionary.

The compression algorithm are as follows [17] :-

```
Dictionary ← empty ; Prefix ← empty ; DictionaryIndex ← 1;
while(characterStream is not empty)
{     Char ← next character in characterStream;
      if(Prefix + Char exists in the Dictionary)
           Prefix ← Prefix + Char ;
      else
       {   if(Prefix is empty)
                CodeWordForPrefix ← 0 ;
           else
                CodeWordForPrefix ← DictionaryIndex for Prefix ;
           Output: (CodeWordForPrefix, Char) ;
           insertInDictionary( ( DictionaryIndex , Prefix + Char) );
           DictionaryIndex++ ;
           Prefix ← empty ;
       }
}
if(Prefix is not empty)
      CodeWordForPrefix ← DictionaryIndex for Prefix;
   Output: (CodeWordForPrefix ,   ) ;
}
```

H. LZW

LZW algorithm is denoted by the Lempel–Ziv–Welch developed by Abraham Lampel , Jacob Zev and Terry Welch in 1984 and it is based on LZ78. A dictionary that is indexed by "codes" is used. The dictionary is supposed to be initialized with 256 entries (indexed with the ASCII codes & that is 0 within 255) representing the ASCII table.

The algorithm are as follows [17]:-

```
Set w=NIL
  loop
      read a character k
      if wk exists in the dictionary
            w=k
else
    Output the code for w add wk to the dictionary
w=k
end loop
```

IV. MEASURING COMPRESSION PERFOMANCE

A. Compression Ratio

Is a percentage that outputs from dividing the compression size(in bits) by the original file size(in bits) and then multiplying the result by 100%[4].

$$\text{Compression Ratio} = \frac{\text{Original Size}}{\text{Compressed Size}} *100\%$$

B. Compression Time

Time is taken for both compression and decompression must be taken into examined as in some cases decompression time and in some cases compression time to be considered is necessary and in some cases both of them are necessary.

C. Entropy

Entropy is the measurement of the totality of information in file. It is number that is conclusion from dividing the compression size in bits by number of symbols in the original file and scale as bits per symbol[4].

V.CONCLUSION

In this paper we need of compression in which of the situations lossy and lossless methods can be used. There are different compression techniques discussed in detail information. We done main focus in this paper is on various data compression methods like dictionary based and entropy based are described. In entropy based technique the RLE (Run length encoding) is not used in much but Shannon Fano and Huffman encoding algorithm are the two methods which are better than RLE algorithm. But Shannon Fano and Huffman compression both are almost same but the Huffman coding is better than Shanon Fano algorithm and in dictionary based algorithm there is LZW method gives best and accurate result.

REFERENCES

[1] S.R. Kodituwakku and U. S. Amarasinghe "Comparison of lossless data compression algorithms For text data" Indian Journal of Computer Science and Engineering Vol 1 No 4 416-425

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

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

[4] Haroon Altarawneh and Mohammad Altarawneh, "Data Compression Techniques on Text Files:A Comparison Study" International Journal of Computer Applications (0975 – 8887) Volume 26– No.5, July 2011

[5] Maninder Pal Singh and Navpreet Singh "A Study of various standards for text compression techniques"

[6] Senthil Shanmugasundaram and Robert Lourdusamy, "A Comparative Study Of Text Compression Algorithms". International Journal of Wisdom Based Computing, Vol. 1 (3), December 2011.

[7] Arup Kumar Bhattacharjee, Tanumon Bej, Saheb 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).

[8] Mark Nelson and Jean-Loup Gailly, "The Data Compression book" 2nd Edition.

[9] Kesheng, W., J. Otoo and S. Arie, 2006. Optimizing bitmap indices with efficient compression, ACM Trans. Database Systems, 31: 1-38.

[10] Data Compression Conference (DCC '00), March 28-30, 2000, Snowbird, Utah.

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

[12] S.R. Kodituwakku and U.S. AmaraSinghe "Compression of Lossless Data Compression Algorithms for Text Data" Indian Journal of Computer Science and Engineering Vol 1 No 4 416-425.

[13] Fano R.M., "The Transmission of Information", Technical Report No.65, Research Laboratory of Electronics, M.I.T., Cambridge, Mass.; 1949.

[14] Huffman D.A., "A method for the construction of minimum redundancy codes", Proceedings of the Institute of Radio Engineers, 40 (9), pp. 1098–1101, September 1952.

[15] M. Pal Singh and N. Singh, "A Study of Various Standards for Text Compression Techniques".

[16] D. Chudasama, K. Parmar, D. Patel, K. Dangarwala, S. Shah, " Survey of Image Compression Method Lossless Approach" IJERT ISSN 2278-0181 Vol, 4 Issue 03, March-2015.

[17] T.Patel, J. Anjela, P. Choudhary, K. Dangarwala , "Survey of Text Compression Algorithms", ISSN 2278-0181 Vol, 4 Issue 03, March-2015.