ISSN: 2277-9655
(I2OR), Publication Impact Factor: 3.785

IJESRT

INTERNATIONAL JOURNAL OF ENGINEERING SCIENCES & RESEARCH TECHNOLOGY

A SURVEY OF DATA COMPRESSION TECHNIQUES

Ms.Anjikhane Manisha*, Ms.Hannure Asma
* M.Tech-CSE Department of Computer Science ,Maharashtra.
M.Tech-CSE Department of Computer Science ,Maharashtra.

ABSTRACT

Most digital data are not stored in the most compact form. Rather, they are stored in whatever way makes them easiest to use, such as: ASCII text from word processors, binary code that can be executed on a computer, individual samples from a data acquisition system, etc. Typically, these easy to use encoding methods require data files about twice as large as actually needed to represent the information. Data compression has important application in the areas of file storage and distributed system.Compression is used to reduce redundancy in stored or communicated data thus increasing effective data density & to reduce the useage of resources..In this Paper we shall discuss about different compression techniques using lossless compression techniques such as Huffman coding,Arithmatic coding and Run Length Encoding(RLE).A conclusion is derived on basis on these Methods.

KEYWORDS: Data Compression,lossless Compression,Huffman Coding,Arithmatic Coding,RLE.

INTRODUCTION

Modern work on data compression began in the late 1940 s with development of information theory.in 1949 claude Shannon & Robert fano devised a systematic way to assign a code word to based on probabilities of blocks.
The Primary Objective of Data Compression is to Minimize the amount of data to be transmitted.the goal of data compression is to reduce redundancy in stored or communication data,thus increasing effective data density and to reduce the useage of resources.

We show the Digramatically representation of compression Method:

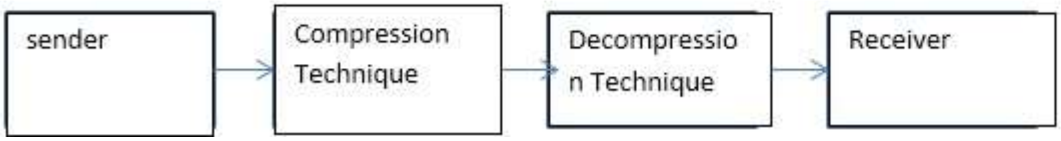


Fig 1.1: diagrammatic representation of Compression

A data compression is transforming a String of Characters in some representation (such as ASCII) into a new String which contains the same information but length is small as possible.it is important application in the area of data transmission and data storage.

The main purposes of this paper to shows the various lossless compression techniques and their comparative study .

Compression

In this Technology by which one or more files or directories size can be reduced so that it is easy to handle.
The main goal of compression is to reduce the no.of bits required to represent data and to decrease the transmission time.Compression is encoding the original data and decompressed to its original form by decoding.A common compressed file which is used today has extension which end with .sit,.tar,.zip.

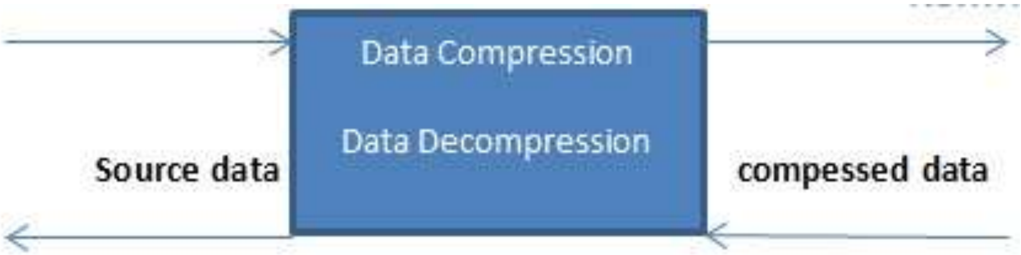


Fig. 1.2: Data Compression and Decompression

Decompression

The Software used to decompressed file depends upon how the form the file was compressed.to decompress .zip file you need software such as winzip.to decompress .sit file,you need a software stuffit expander program. Winzip does not decompress .sit file but one version of Stuffit expander can decompress both .zip and .sit files.file ending with .sea and .exe are called as self extracting files so such file do not required any special software to decompress just click on file it will automatically decompressed.

RELATED WORKS

There are two Compression Techniques Commonly used.

Figure:

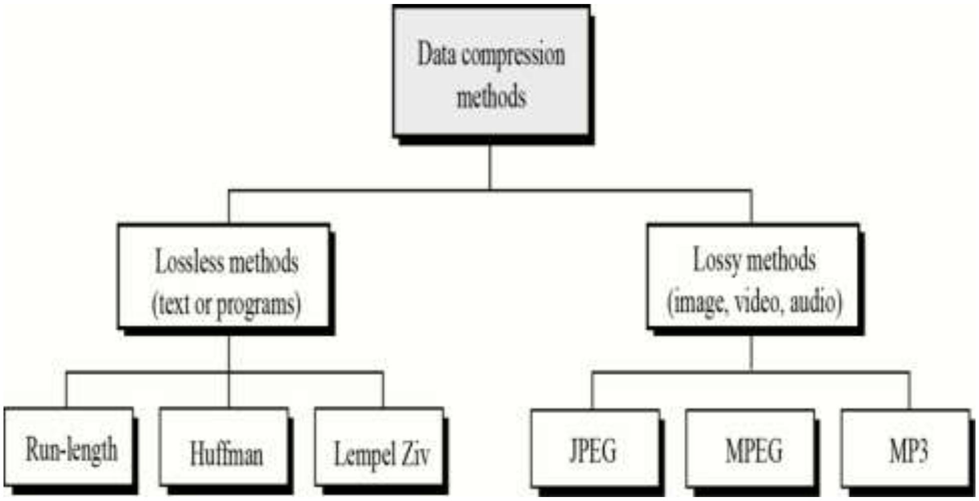

Data compression methods
Lossless methods (text or programs)
Lossy methods (image, video, audio)
Run-length
Huffman
Lempel Ziv
JPEG
MPEG
MP3

Fig2.1: - Tree representation of compression Method

Lossless Compression

It is used to reduce the amount of source information to be transmitted in such way that when we compressed information is decompressed there is not loss of any information. Lossless compression is used when it is important that the original data and the decompressed data be identical. Lossless text data compression algorithms usually exploit statistical redundancy in such a way so as to represent the sender's data more concisely without any error or any sort of loss of important information contained within the text input data. Since most of the real-world data has statistical redundancy, therefore lossless data compression is possible

Lossy Compression

Lossy compression creates smaller file by discarding (losing) some information about original image.

Lossy data compression is named for what it does after one applies lossy data compression to a message,the message can never be recovered exactly as it was before it was compressed.when the compressed message is decoded it does not give the original message.data has been lost.

In this paper we only discuss on the lossless compression technique which are used on the text data formatting.

There are different losseless compression techniques which we are explain in our paper are Huffman coding,Run length coding,Arithmatic coding method and compare their performance.

Huffman Coding

The Huffman coding is developed by David A.,while he was a Ph.D. student at MIT. A Huffman code is a particular type of optimal prefix code that is commonly used for lossless data compression. The Huffman algorithm is based on statistical coding, which means that the probability of a symbol has a direct bearing on the length of its representation.

Huffman Coding Algorithms

It is bottom-up approach the Huffman algorithm works from leaves to the root in the opposite direction.

```
    1.create a laef node for each symbol and add it to frequency of occurrence.
    2.while there is more than one nodes in the queue:
          1.Remove the two nodes of lower probability or frequency from the queue.
          2.assign 0 and 1 respectively to any node already ssigned to these nodes.
          3.create a new internal node with these two nodes as children & with probability equal to the sum of the two nodes probabilities.
          4.add the new nodes to the queue.
  3.The remaining node is the root node and the tree is complete.
```

Example:

File consisting of symbols (AAABAABABCDEDDBCDADA) required a file
size using Huffman coding and compare it with the original file size if you use a fixed length encoding
(3bit to represent each character)
Solution:

Probability of each symbol or number of frequency as follows:

Table 1 shows number of frequency for all symbols

Symbols	**Number of Frequency**	**probability**
A	8	0.4
B	4	0.2
C	2	o.1
D	5	0.25
E	1	0.0

Here,
Probability=(number of frequency / summation ofsymbols)
From the tablewe know that the E, C symbols are the least frequent, and so the huffman tree is configured as follows:

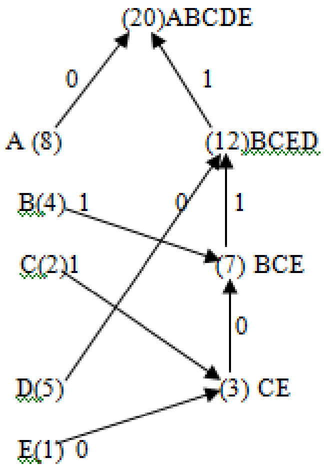


Figure2.2: Huffman Tree

From Huffman tree extract binary representation of each code as shown in the following table:

Table 2 each symbol has its own coding

symbols	Code
A	0
B	111
C	1101
D	10
E	1100

To calculate the file size using Huffman coding is as follows:

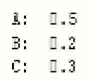
Table 3 shows the account file size

Symbols	Frequency value	Length of code	Overall size of the code in the file
A	8	1	8
B	4	3	12
C	2	4	8
D	5	2	10
E	1	4	4
total		42	

Hence the file size after compression algorithm using Huffman 42 bits, while the original file size before compressing (3 * 20 = 60 bits). Here, 20 is the number of characters.

Arithmatic Coding Technique

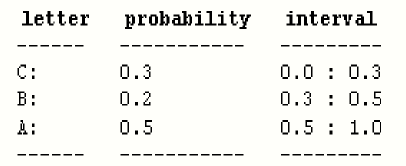
Huffman coding looks pretty slick, and it is, but there's a way to improve on it, known as "arithmetic coding". The idea is subtle and best explained by example [4, 8, 10]. Suppose we have a message that only contains the characters A, B, and C, with the following frequencies, expressed as fractions:



A:	0.5
B:	0.2
C:	0.3

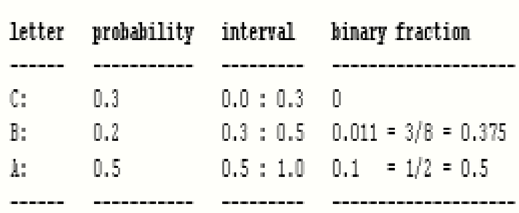
To show how arithmetic compression works, we first set up a table, listing characters with their probabilities along with the cumulative sum of those probabilities. The cumulative sum defines "intervals", ranging from the bottom value to less than, but not equal to, the top value. The order does not seem to be important.

To show how arithmetic compression works, we first set up a table, listing characters with their probabilities along with the cumulative sum of those probabilities. The cumulative sum defines "intervals", ranging from the bottom value to less than, but not equal to, the top value. The order does not seem to be important.



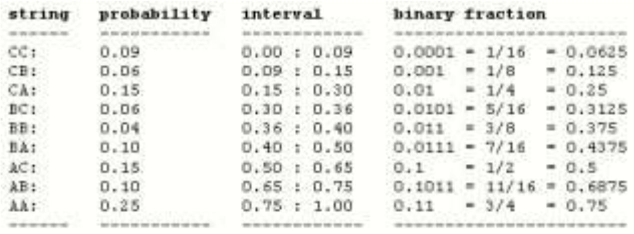
letter	probability	interval
C:	0.3	0.0 : 0.3
B:	0.2	0.3 : 0.5
A:	0.5	0.5 : 1.0

Now each character can be coded by the shortest binary fraction that falls in the character's probability interval:



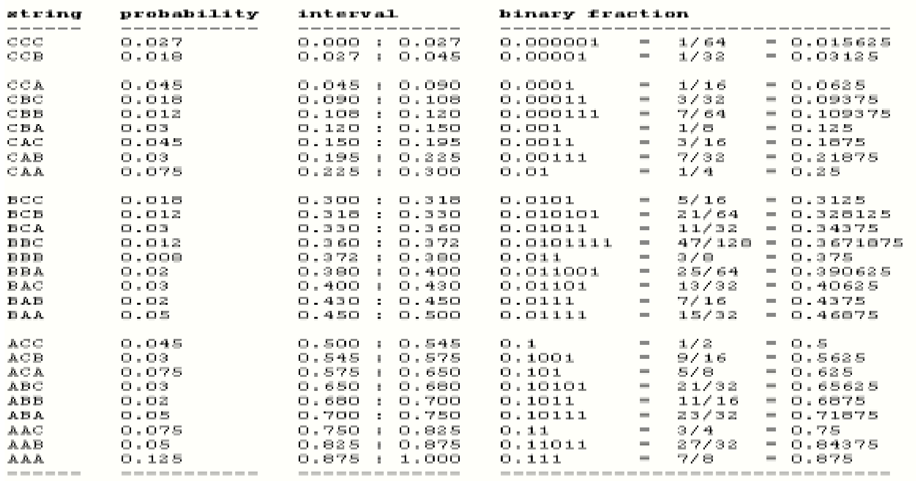
letter	probability	interval	binary fraction
C:	0.3	0.0 : 0.3	0
B:	0.2	0.3 : 0.5	0.011 = 3/8 = 0.375
A:	0.5	0.5 : 1.0	0.1 = 1/2 = 0.5

Sending one character is trivial and uninteresting. Let's consider sending messages consisting of all possible permutations of two of these three characters, using the same approach:



string	probability	interval	binary fraction
CC:	0.09	0.00 : 0.09	0.0001 = 1/16 = 0.0625
CB:	0.06	0.09 : 0.15	0.001 = 1/8 = 0.125
CA:	0.15	0.15 : 0.30	0.01 = 1/4 = 0.25
BC:	0.06	0.30 : 0.36	0.0101 = 5/16 = 0.3125
BB:	0.04	0.36 : 0.40	0.011 = 3/8 = 0.375
BA:	0.10	0.40 : 0.50	0.0111 = 7/16 = 0.4375
AC:	0.15	0.50 : 0.65	0.1 = 1/2 = 0.5
AB:	0.10	0.65 : 0.75	0.1011 = 11/16 = 0.6875
AA:	0.25	0.75 : 1.00	0.11 = 3/4 = 0.75

The higher the probability of the string, in general the shorter the binary fraction needed to represent it. Let's build a similar table for three characters now:



string	probability	interval	binary fraction
CCC	0.027	0.000 : 0.027	0.000001 = 1/64 = 0.015625
CCB	0.018	0.027 : 0.045	0.00001 = 1/32 = 0.03125
CCA	0.045	0.045 : 0.090	0.0001 = 1/16 = 0.0625
CBC	0.018	0.090 : 0.108	0.00011 = 3/32 = 0.09375
CBB	0.012	0.108 : 0.120	0.000111 = 7/64 = 0.109375
CBA	0.03	0.120 : 0.150	0.001 = 1/8 = 0.125
CAC	0.045	0.150 : 0.195	0.0011 = 3/16 = 0.1875
CAB	0.03	0.195 : 0.225	0.00111 = 7/32 = 0.21875
CAA	0.075	0.225 : 0.300	0.01 = 1/4 = 0.25
BCC	0.018	0.300 : 0.318	0.0101 = 5/16 = 0.3125
BCB	0.012	0.318 : 0.330	0.010101 = 21/64 = 0.328125
BCA	0.03	0.330 : 0.360	0.01011 = 11/32 = 0.34375
BBC	0.012	0.360 : 0.372	0.0101111 = 47/128 = 0.3671875
BBB	0.008	0.372 : 0.380	0.011 = 3/8 = 0.375
BBA	0.02	0.380 : 0.400	0.011001 = 25/64 = 0.390625
BAC	0.03	0.400 : 0.430	0.01101 = 13/32 = 0.40625
BAB	0.02	0.430 : 0.450	0.0111 = 7/16 = 0.4375
BAA	0.05	0.450 : 0.500	0.01111 = 15/32 = 0.46875
ACC	0.045	0.500 : 0.545	0.1 = 1/2 = 0.5
ACB	0.03	0.545 : 0.575	0.1001 = 9/16 = 0.5625
ACA	0.075	0.575 : 0.650	0.101 = 5/8 = 0.625
ABC	0.03	0.650 : 0.680	0.10101 = 21/32 = 0.65625
ABB	0.02	0.680 : 0.700	0.1011 = 11/16 = 0.6875
ABA	0.05	0.700 : 0.750	0.10111 = 23/32 = 0.71875
AAC	0.075	0.750 : 0.825	0.11 = 3/4 = 0.75
AAB	0.05	0.825 : 0.875	0.11011 = 27/32 = 0.84375
AAA	0.125	0.875 : 1.000	0.111 = 7/8 = 0.875

Obviously, this same procedure can be followed for more characters, resulting in a longer binary fractional value. What arithmetic coding does is find the probability value of a particular message, and arrange it as part of a numerical order that allows its unique identification.

Run-length encoding (RLE)

Run-length encoding is a data compression technique that is supported by the bitmap file formats, like TIFF, BMP. RLE is easy to implement and quick to execute. RLE works by reducing the physical size of a repeating string of characters. This repeating string, called a run, is encoded into two bytes. The first byte represents the number of characters in the run and is called the run count. The second byte is the value of the character in the run, which is in the range of 0 to 255, and is called the run value.

Ex: Uncompressed, a character run of 10 A characters would normally require 10 bytes to store:

AAAAAAAAAA

The same string after RLE encoding would require only two bytes: 10A

The 10A code generated to represent the character string is called an RLE packet.

Here, the first byte, 10, is the run count and contains the number of repetitions. The second byte, A, is the run value and contains the actual repeated value in the run.

A new packet is generated each time the run character changes, or each time the number of characters in the run exceeds the maximum count.

Ex: AAAAABBBCC

Using run-length encoding this could be compressed into three 2-byte packets: 5A3B2C.

RLE techniques are simple and fast, but their compression efficiency depends on the type of image data being encoded. A black-and-white image that is mostly white, such as the page of a book, will encode very well, due to the large amount of contiguous data that is all the same colour.

Applications:

- Run-length encoding used for lossless data compression technique.
- RLE is suited for bitmapped images that had a finite set of colours (like computer icons).
- It does not suited for continuous-tone images like photographs.
- Run-length encoding is also used in fax machines.

CONCLUSION

There we talked about a need of data compression, and situations in which these lossless methods are useful. The algorithms used for lossless compression are described in brief.. In the Statistical compression techniques, Arithmetic coding technique performs with an improvement over Huffman coding, over Shannon-Fano coding and over Run Length Encoding technique. Another area of research would be to implement the compression scheme so that searching is faster

REFERENCES

[1] Xue Li, , Vasu D. Chakravarthy, , Bin Wang, and Zhiqiang Wu, “Spreading Code Design of Adaptive Non-Contiguous SOFDM for Dynamic Spectrum Access” in IEEE JOURNAL OF SELECTED TOPICS IN SIGNAL PROCESSING, VOL. 5, NO. 1, FEBRUARY 2011

[2] J. D. Poston and W. D. Horne, “Discontiguous OFDM considerations for dynamic spectrum access in idel TV channels,” in Proc. IEEE DySPAN, 2005.

[3] R. Rajbanshi, Q. Chen, A.Wyglinski, G. Minden, and J. Evans, “Quantitative comparison of agile modulation technique for cognitive radio tranceivers,” in Proc. IEEE CCNC, Jan. 2007, pp. 1144–1148.

[4] V. Chakravarthy, X. Li, Z. Wu, M. Temple, and F. Garber, “Novel overlay/underlay cognitive radio waveforms using SD-SMSE framework to enhance spectrum efficiency—Part I,” IEEE Trans. Commun., vol. 57, no. 12, pp. 3794–3804, Dec. 2009.

[5] V. Chakravarthy, Z. Wu, A. Shaw, M. Temple, R. Kannan, and F. Garber, “A general overlay/underlay analytic expression for cognitive radio waveforms,” in Proc. Int. Waveform Diversity Design Conf., 2007.

[6] V. Chakravarthy, Z. Wu, M. Temple, F. Garber, and X. Li, “Cognitive radio centric overlay-underlay waveform,” in Proc. 3rd IEEE Symp. New Frontiers Dynamic Spectrum Access Netw., 2008, pp. 1–10.

[7] X. Li, R. Zhou, V. Chakravarthy, and Z. Wu, “Intercarrier interference immune single carrier OFDM via magnitude shift keying modulation,” in Proc. IEEE Global Telecomm. Conf. GLOBECOM , Dec. 2009, pp. 1–6.

[8] Parsaee, G.; Yarali, A., "OFDMA for the 4th generation cellular networks" in Proc. IEEE Electrical and Computer Engineering, Vol.4, pp. 2325 - 2330, May 2004.

[9] 3GPP R1-050971,"R1-050971 Single Carrier Uplink Options for EUTRA: IFDMA/DFT-SOFDM Discussion and Initial Performance Results ",http://www.3GPP.org,Aug 2005

[10] IEEE P802.16e/D12,'Draft IEEE Standard for Local and metropolitan area networks-- Part 16: Air Interface for Fixed and Mobile Broadband Wireless Access Systems', October 2005

[11] 3GPP RP-040461, Study Item: Evolved UTRA and UTRAN, December 200

[12] R. Mirghani, and M. Ghavami, “Comparison between Wavelet-based and Fourier-based Multicarrier UWB Systems”, IET Communications, Vol. 2, Issue 2, pp. 353-358, 2008.

[13] R. Dilmirghani, M. Ghavami, “Wavelet Vs Fourier Based UWB Systems”, 18th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications, pp.1-5, Sep. 2007.

[14] M. Weeks, Digital Signal Processing Using Matlab and Wavelets, Infinity Science Press LLC, 2007.

[15] S. R. Baig, F. U. Rehman, and M. J. Mughal, “Performance Comparison of DFT, Discrete Wavelet Packet and Wavelet Transforms in an OFDM Transceiver for Multipath Fading Channel,”, 9th IEEE International Multitopic Conference, pp. 1-6, Dec. 2005.

[16] N. Ahmed, Joint Detection Strategies for Orthogonal Frequency Division Multiplexing, Dissertation for Master of Science, Rice University, Houston, Texas. pp. 1-51, Apr. 2000.