Volume 6, Issue 5, May 2016 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Analysis of Data Compression Techniques using Huffman Coding and Arithmetic Coding

M. Senthilkumar*
Research Scholar, AMET University, Chennai, Tamilnadu, India

V. Mathivanan
Research Supervisor, AMET University, Chennai, Tamilnadu, India

Abstract— The aim of multimedia data compression is the process of saving storage space or packing more data into the same size space. Data compression involves not only compression techniques can work on different type of digital data, such as character or byte in data file or whatever. Huffman Coding is a statistical data compression technique, which gives the reduction in the average code length used to represent the symbols of an alphabet. It is a Method for the Construction of Minimum-Redundancy Codes. Arithmetic coding completely bypasses the idea of replacing an input symbol with a specific code. Instead, it takes a stream of input symbols and replaces it with a single floating point output number. I consider a sample data and compare that data compression with the mentioned two methods.

Keywords— Compression, Digital data, input symbols, and minimum-redundancy

I. INTRODUCTION

The aim of multimedia data compression is the process of saving storage space or packing more data into the same size space. Data compression involves not only compression techniques can work on different type of digital data, such as character or byte in data file or whatever. In general data compression literature speaks in general terms of compression symbols.

Data compression literature also refers to data compression as data "encoding" and decompression as "decoding". Decompression means restoring the compressed data to its original form and it is also referred to as expansion. The term "message" is used to denote short data strings.

Data compression technique is used for saving disk space, reducing the time needed for communication or the time needed for data transfer and more. Data to be handled as well as software has been growing, and the amount of information communicated between systems has also been constantly increasing. In these circumstances, data compression technology is regarded as an important factor to support the information infrastructure.

Data compression technology is not only used in PCs, but also in many fields like modems, routers, digital cameras, facsimiles, CDs, MDs(Mini disk), video on demand(VOD), TV conference system, DVDs, Digital telephones and other fields yet the data compression technique generally refer to data stored in 'files' and file transfer over phone lines.

In MS-DOS, used programs like ARC and PKZIP and ubiquitous. UNIX used has the COMPRESS and COMPACT utilities and WINDOWS used have WINZIP utilities.

II. LITERATURE SURVEY

There are two types of data compression, "lossless" data compression and the "lossy" compression.

Lossless data compression is used when the data has to be uncompressed exactly as it was before compression. This type of compression used when storing database records, spreadsheets and word processing files. Text files are stored using lossless techniques, since losing a single character can in the worst case, make the text dangerously misleading.

Archival storage of master sources for images, videos data and audio data generally needs to be lossless as well. However there are strict limits to the amount of compression that can be obtained with lossless compression. Lossless compression ratio is generally in the range of 2:1 to 8:1.

Lossy compression, in contrast, works on the assumption, that the data does not have to be stored perfectly. Much information can be simply through away from images, video data and audio data, and when decompressed, the data will still be of acceptable quality. Lossy data compression concedes a certain loss of accuracy in exchange for greatly increased compression. Lossy methods can be used on speech and graphics, and they are capable of acting dramatically higher compression ratios.

The question of which is a 'better', lossless or lossy technique is pointless. Each has its own uses, with lossless techniques better in some cases and lossy techniques better in others. In fact, lossless and lossy technique are used together to obtain the higher compression ratio.

In general, data compression consists of talking a stream of symbols and transforming them into codes. If the compression is effective, the resulting stream of codes will be smaller than the original symbols. The decision is output a certain code for a symbol or set of symbols is based on a model. So, data compression consists of two components, modeling and coding.

Crypto graphic enabled Compression of Multimedia content [7][8] is the art of hiding and transmitting data through apparently innocuous carriers in an effort to conceal the existence of the data, the word Steganography literally means covered or hiding writing as derived from Greek. Steganography has its place in security. It is not intended to replace cryptography but supplement it.

Hiding a message with Steganography methods reduces the chance of a message being detected. If the message is also encrypted then it provides another layer of protection. Therefore, some steganography methods combine traditional Cryptography with Steganography; the sender encrypts the secret message prior to the overall communication process, as it is more difficult for an attacker to detect embedded cipher text in a cover. Hidden information in the cover data is known as the "embedded" data and information hiding is a general term encompassing many sub disciplines [9] [10], is a term around a wide range of problems beyond that of embedding message in content.

The term hiding here can refer to either making the information undetectable or keeping the existence of the information secret. Information hiding is a technique of hiding secret using redundant cover data such as images, audios, movies, documents, etc. This technique has recently become important in a number of application areas. For example, digital video, audio, and images are increasingly embedded with imperceptible marks, which may contain hidden signatures or watermarks that help to prevent unauthorized copy [11] [12].

Many different methods enable hiding information in audio and image. These methods may include hiding information in unused space in file headers to hold 'extra' information. Embedding techniques can range from the placement of information in imperceptible level [13] [14] (noise), manipulation of compression algorithms, and the modification of carrier properties.

In audio, small echoes or slight delays can be added or subtle signals can be masked by sound of higher amplitude. Information (data) can be hidden in different ways in image. To hide information; straight message insertion may encode every bit of information in the image or selectively embed the message in busy areas where it would be less perceptible. A message may also be scattered randomly or repeated several times throughout the image

Since the advent of computers there has been a vast dissemination of information, some of which needs to be kept private, some of which doesn't. The information may be hidden in two basic ways (Cryptography and Steganography).The methods of Cryptography does not conceal the presence of secret information but render it unintelligible to outsider by various transformations of the information that is to be put into secret form, while methods of Steganography conceal the very existence of the secret information[16][17].

In Cryptography, a block cipher is a symmetric key cipher which operates on fixed-length groups of bits, termed blocks, with an unvarying transformation. When encrypting, a block cipher might take a (for example) 128-bit block of plaintext as input, and outputs a corresponding 128-bit block of cipher text. The exact transformation is controlled using a second input — the secret key.

Decryption is similar: the decryption algorithm takes, in this example, a 128-bit block of cipher text together with the secret key, and yields the original 128-bit block of plaintext. To encrypt messages longer than the block size (128 bits in the above example), a mode of operation is used [18] [19]. Block ciphers can be contrasted with stream ciphers; a stream cipher operates on individual digits one at a time and the transformation varies during the encryption. The distinction between the two types is not always clear-cut: a block cipher, when used in certain modes of operation, acts effectively as a stream cipher.

III. DATA HIDING TECHNIQUES

In this topic we discuss fundamental lossless data compression technique in detail. The discussion will focus on compressing text file, but they will work for other type of files as well.

3.1 Huffman Coding

'Huffman Coding' is a statistical data compression technique, which gives the reduction in the average code length used to represent the symbols of an alphabet. It creates variable length codes that are an integral length of bits. It is a Method for the Construction of Minimum-Redundancy Codes.

3.1.1 Compression

Huffman Coding is an efficient lossless compression technique in which the characters in a data file are converted to a binary code, when the most common characters in the file have the shortest binary codes and the least common have the longest. The technique works by creating a binary tree of nodes. These can be stored in a regular array, the size of which depends on the number of symbols (N).

A node can be either a leaf node or an internal node. Initially, all nodes are leaf nodes, which contain the symbol itself, the weight (frequency of appearance) of the symbol and optionally, a link to a parent node, which makes it easy to read the code (in reverse) starting from a leaf node.

Internal nodes contain symbol weight, links to two child nodes and the optional link to a parent node. As a common convention, bit '0' represents following the left child and bit '1' represents following the right child. A finished tree has N leaf nodes and N-1 internal nodes. In this technique a binary tree known as the 'Huffman tree' is built according to the following algorithm.

Algorithm:

1. For a given list of symbols, develop a corresponding list of probabilities or frequency count so that each symbol's relative frequency of occurrence is known.
2. Sort the list of symbols according to frequency, with the most frequency occurring symbols at the left and the least count at the right.

3. The two free nodes with the lowest weights are located and a parent node for the two nodes is created. It is assigned a weight equal to the sum of the two-child node.
4. The parent node is added to list of free nodes, and the two child nodes are remaining from the list.
5. On the child nodes is designated as the path taken from the parent node when decoding a 'o' bit. The other is arbitrarily set to the '1' bit.
6. Repeat step 3 to 5 until only one free node is list. This free node is designated as the root of the node.
7. Trace down the tree to get the 'Huffman Codes' with the shortest code assigned to the characters with the greatest frequency.

3.1.2 Decompression

Decoding a huffman encoding is just as easy: as we read bits in from our input stream we traverse the tree beginning at the root, taking the left hand path if we read a 0 and the right hand path if you read a 1. When we hit a leaf, we have found the code.

Generally, any huffman compression scheme also requires the huffman tree to be written out as part of the file, otherwise we cannot decode the data. For a static tree, we don't have to do this since the tree is known and fixed.

The easiest way to output the huffman tree itself is to, starting at the root, dump first the left hand side then the right hand side. For each node we output a **0**, for each leaf we output a **1** followed by N bits representing the value.

3.1.3 Merits and Demerits

This compression algorithm is mainly efficient in compressing text or program files. Images like they are often used in prepress are better handled by other compression algorithms. These best adaptive Huffman coding algorithms are still relatively time and memory consuming.

The problem with combining adaptive modeling with Huffman coding is that rebuilding the Huffman tree is a very expensive process. For an adaptive scheme to be efficient, it is necessary to adjust the Huffman tree after every character.

3.2 Arithmetic Coding

Statistical methods of data compression they operate by encoding symbols one at a time. The symbols are encoded into variable length output codes. The length of the output code varies based on the probability or frequency of the symbol. Low probability symbols are encoded using many bits, and high probability symbols are encoded using fewer bits.

Data compression operates in general by taking "symbols" from an input "text", processing them, and writing "codes" to a compressed file. To be effective, a data compression scheme needs to be able to transform the compressed file back into an identical copy of the input text. Needless to say, it also helps if the compressed file is smaller than the input text!

3.2.1 Compression

Arithmetic coding completely bypasses the idea of replacing an input symbol with a specific code. Instead, it takes a stream of input symbols and replaces it with a single floating point output number. The longer (and more complex) the message, the more bits are needed in the output number. It was not until recently that practical methods were found to implement this on computers with fixed sized registers.

The output from an arithmetic coding process is a single number less than 1 and greater than or equal to 0. This single number can be uniquely decoded to create the exact stream of symbols that went into its construction. In order to construct the output number, the symbols being encoded have to have a set probabilities assigned to them.

Algorithm:

The algorithm to accomplish this for a message of any length is shown below:

```
Set low to 0.0
Set high to 1.0
        While there are still input symbols do
            get an input symbol
            code_range = high - low.
            high = low + range*high_range(symbol)
            low = low + range*low_range(symbol)
        End of While
output low
```

3.2.2 Decompression

Given this encoding scheme, it is relatively easy to see how the decoding process will operate. We find the first symbol in the message by seeing which symbol owns the code space that our encoded message falls in. Since the number 0.2572167752 falls between 0.2 and 0.3, we know that the first character must be "B". We then need to remove the "B" from the encoded number. Since we know the low and high ranges of B, we can remove their effects by reversing the process that put them in. First, we subtract the low value of B from the number, giving 0.0572167752. Then we divide by the range of B, which is 0.1. This gives a value of 0.572167752. We can then calculate where that lands, which is in the range of the next letter, "I".

Algorithm:

The algorithm for decoding the incoming number looks like this:

```
get encoded number
Do
```


find symbol whose range straddles the encoded
number
output the symbol
range = symbol low value - symbol high value
subtract symbol low value from encoded number
divide encoded number by range
Until no more symbols

Note that it have conveniently ignored the problem of how to decide when there are no more symbols left to decode. This can be handled by either encoding a special EOF symbol, or carrying the stream length along with the encoded message.

3.2.3 Merits and Demerits

This algorithm could be applied today to circumstances where either storage or transmission costs are extremely high. It will increase the speed of compression and expansion, without affecting the compression ratio. The compression ratios might not be as good as more sophisticated models, but the memory consumption is minimized.

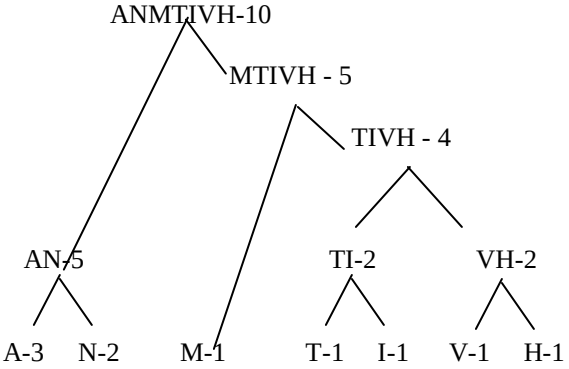
IV. HUFFMAN CODING VS ARITHMETIC CODING

4.1 Huffman Coding:

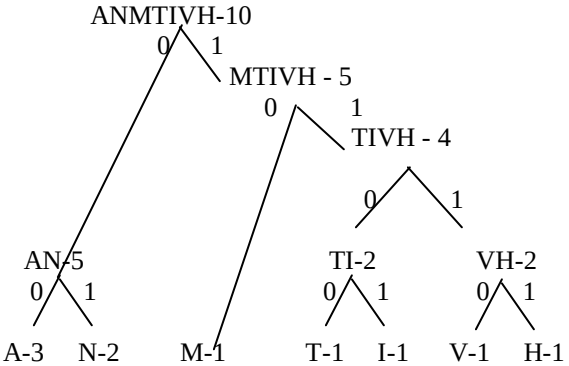
To see how Huffman code works, assume that a text file is to be compression.
Let us consider a Text "MATHIVANAN". Each letter containing 8 bits means, 10 * 8 = 80 bits. But when we are applying Huffman coding algorithm it may vary. Let us discuss that now.
In the above text we can calculate each letters count and mention as below.

A	N	M	T	I	V	H
3	2	1	1	1	1	1

Using Huffman coding algorithm make Huffman tree as below.



In the above tree we have to give the binary value for each node and assign the binary value for full tree.



After assign value, we can make the continuous binary value to the word NARAYAN MURTHY as follows.

A	N	M	T	I	V	H
00	01	10	1100	1101	1100	1111

Total Bit value = 22 bits.
When we compare with the Original word before compression,
(22 / 80) * 100 = 28 %.
So we can save 62% using Huffman coding compression.

4.2 Arithmetic Coding:

If we were going to encode the random message "MATHIVANAN", we would have a probability distribution that looks like this:

Character	Probability

A	3/10
H	1/10
I	1/10
M	1/10
N	2/10
T	1/10
V	1/10

Once the character probabilities are known, the individual symbols need to be assigned a range along a "probability line", which is nominally 0 to 1. It doesn't matter which characters are assigned which segment of the range, as long as it is done in the same manner by both the encoder and the decoder. The Nine-character symbol set use here would look like this:

Character	Probability	Range
A	3/10	0.00 - 0.30
H	1/10	0.30 - 0.40
I	1/10	0.40 - 0.50
M	1/10	0.50 - 0.60
N	2/10	0.60 - 0.80
T	1/10	0.80 - 0.90
V	1/10	0.90 - 1.00

Each character is assigned the portion of the 0-1 range that corresponds to its probability of appearance. Note also that the character "owns" everything up to, but not including the higher number. So the letter 'V' in fact has the range 0.90 - 0.9999....

The most significant portion of an arithmetic coded message belongs to the first symbol to be encoded. When encoding the message "MATHIVANAN", the first symbol is "M". In order for the first character to be decoded properly, the final coded message has to be a number greater than or equal to 0.50 and less than 0.60. What we do to encode this number is keep track of the range that this number could fall in. So after the first character is encoded, the low end for this range is 0.50 and the high end of the range is 0.60.

After the first character is encoded, we know that our range for our output number is now bounded by the low number and the high number. What happens during the rest of the encoding process is that each new symbol to be encoded will further restrict the possible range of the output number. The next character to be encoded, 'A', owns the range 0.00 through 0.30. If it was the first number in our message, we would set our low and high range values directly to those values. But 'A' is the second character.

So what we do instead is say that 'A' owns the range that corresponds to 0.10-0.30 in the new sub range of 0.2 - 0.3. This means that the new encoded number will have to fall somewhere in the 50th to 60th percentile of the currently established range. Applying this logic will further restrict our number to the range 0.25 to 0.26.

Following this process through to its natural conclusion with our chosen message looks like this:

New Character	Low value	High Value
	0.0	1.0
M	0.2	0.3
A	0.25	0.26
T	0.256	0.258
H	0.2572	0.2576
I	0.25720	0.25724
V	0.257216	0.257220
A	0.2572164	0.2572168
N	0.25721676	0.2572168
A	0.257216772	0.257216776
N	0.2572167752	0.2572167756

So the final low value, 0.2572167752 will uniquely encode the message "MATHIVANAN" using our present encoding scheme.

V. RESULTS COMPARISION

The statistics for compression ratio, compression time and expansion time are presented in this section in the form of tables and charts.

Table 1 – Compression Table

Techniques	Size of Input File	Size of Compressed File	Compression Ratio
Huffman Coding	589312 Bytes	340021 bytes	42.30%
Arithmetic Coding	589312 Bytes	327837 bytes	44.37%

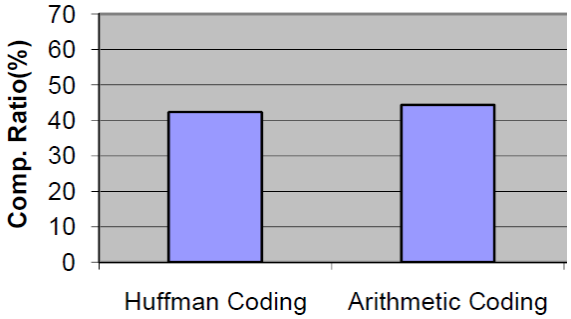


Figure 1 - Compression Ratio chart

Table 2 – Compression Time Table

Compression Techniques	Compression Time
Huffman Coding	0.08 Seconds
Arithmetic Coding	0.09 Seconds

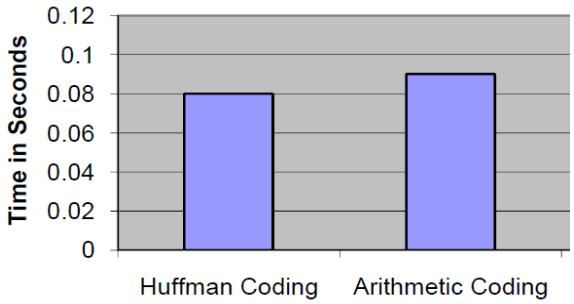


Figure 2 - Compression Time Chart

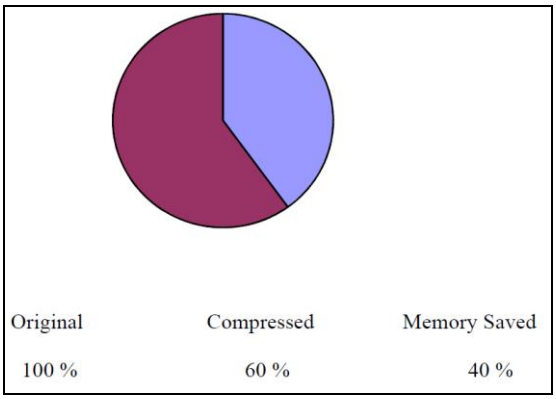


Figure 3 - Memory saved chart

VI. CONCLUSIONS

Compressed multimedia information hide is a way of encoding digital data, and it takes up less storage space and requires less network bandwidth to be transmitted. This is particularly useful in communication because it lets devices transmit the same amount of data in fewer bits. The objective of the work is to analyse the data compression techniques in different methods Huffman coding and Arithmetic coding. The compression methods are also to help to find the better method in an effective and efficient manner. One of the disadvantage for this comparison is not suitable for all type of compression methods. In, future this work can be evaluated using another two compression methods LZ78 coding and LZW coding.

ACKNOWLEDGMENT

The heading of the Acknowledgment section and the References section must not be numbered.

Causal Productions wishes to acknowledge Michael Shell and other contributors for developing and maintaining the IEEE LaTeX style files which have been used in the preparation of this template. To see the list of contributors, please refer to the top of file IEEETran.cls in the IEEE LaTeX distribution.

REFERENCES

[1] Information hiding: a new approach in text steganography , l. y. por, b. delina , faculty of computer science and information technology, university of malaySIa, 50603, kualalumpur, malaysia porlip@um.edu.my, delinabeh@yahoo.com.

[2] Image information hiding –an survey, d. saravanan, a. ronald doni & a. abisha ajith sathyabama university, chennai, tamilnadu, india

[3] Information hiding codes and their applications to images and audioby mehmet kivanc¸ mihc¸ akb.s., bilkent university, 1996 m.s., university of illinois at urbana-champaign, 1999

[4] Information hiding techniques: a tutorial review , sabu m thampi ,assistant professor department of computer science & engineering ,lbs college of engineering, kasaragod kerala- 671542, s.india smtlbs@yahoo.co.in

[5] Ire journal on selected areas in communications .special issue on copyright and privacy protection, vol. 16, no. 4, may 1998.

[6] Proceedings of the ieee .special issue on identification and protection of multimedia information, vol. 87, no. 7, july 999.

[7] f. a. p. petitcolas, r. j. anderson, and m. g. kuhn, "information hiding - a survey," proceedings of the ieee .special issue on protection of multimedia content, vol. 87, no. 7, pp. 1062-1078, july 1999.

[8] w. bender, d. gruhl, n. morimoto, and a. liu, "techniques for data hiding," in proceedings of spie , 1995, pp. 2420-2440. [13] i. j. cox, j. killian, f. t. leighton, and t. shamo on, "secure spread sp ectrum watermarking for multimedia," ieee transactions on image processing, vol. 6, no. 12, pp. 1673-1687, dec. 1997.

[9] f. hartung and m. kutter, "multimedia watermarking techniques," proceedings of the ieee .sp ecial issue on protection of multimedia content, vol. 87, no. 7, pp. 1079-1107, july 1999.

[10] m. d. swanson, m. koyabashi, and a. h. tewfik, "multimedia data-emb edding and watermarking strategies," proceedings of the ieee , vol. 86, no. 6, pp. 1064-1087, june 1998.

[11] r. b. wolfgang, c. i. po dilchuk, and e. j. delp, "perceptual watermarks for digital images and video," proceedings of the ieee . sp ecial issue on protection of multi- media content, vol. 87, no. 7, pp. 1108-1126, july 1999.

[12] Read M. Saleh, (2001), "Information Hiding in Wave Media File by Using Low Bit Encoding", M.Sc thesis, University of Technology, Baghdad, Iraq.

[13] Provos N., (January 31, 2001) "Probabilistic Methods for Improving Information Hiding", Center for Information Technology Integration, University of Michigan, USA. Email: provos@citi.umich.edu

[14] Luis von Ahn,, Manuel Blum,, Nicholas J. Hopper, and John Langford, (2003),"Using Hard AI Problems for Security", Computer Science Dept., Carnegie Mellon University, Pittsburgh PA 15213, USA.

[15] T.J. Watson Research Center, Yorktown Heights NY 10598, USA. Jacob T. Jackson, Gregg H. Gunsch, Roger L. Claypoole, Jr., and Gary B.

[16] Lamont, (2003)" Novel Steganography Detection Using an Artificial Immune System Approach ", Air Force Institute of Technology Department of Electrical and computer Engineering 2950 Hobson Way, Bldg 640 Wright .

[17] Compressed Video over Networks, Editors Ming-Ting Sun and Amy R. Reibman, Marcel Dekker Publishers, 1st edition, 2001.

[18] R. Rajan, D. Verma, S. Kamat, E. Felstaine, S. Herzog, "A Policy Framework for Integrated and Differentiated Services in the Internet," IEEE Network Magazine, Sept./Oct. 1999.

[19] K. Stuhlmuller, N. Farber, M. Link, B. Girod, "Analysis of Video Transmission over Lossy Channels," IEEE Journal on Selected Areas in Communication, Vol. 18, No 6, June 2000.

[20] Vikas Singla, Rakesh Singla and Sandeep Gupta, "Data Compression Modeling: Huffman and Arithmetic," IJCIM, Vol. 16, No 3.

