A Survey of Compression Techniques

Sashikala, Melwin .Y., Arunodhayan Sam Solomon, M.N.Nachappa

Abstract - Digital data is owned, used and enjoyed by many people all over the world. Compressing data is mostly done when we face with problems of constraints in memory. In this paper we have attempted to discuss in general about compression and decompression, the different techniques of compression using 'Lossless method compression and decompression' on text data using a simple coding technique called Huffmann coding.

Index Words: Lossless text data compression, Huffman coding.

I. INTRODUCTION

In 1838 morse code used data compression for telegraphy which was based on using shorter code words for letters such as "e" and "t" that are more common in English . Modern work on data compression began in the late 1940s with the development of information theory. In 1949 Claude Shannon and Robert Fano devised a systematic way to assign code words based on probabilities of blocks. In 1951 David Huffmann found an optimal method for Data Compression. Early implementations were typically done in hardware, with specific choices of code words being made as compromises between compression and error correction. with online storage of text files becoming common, software compression programs began to be developed IN EARLY 1970S , almost all COMPRESSIONS were based on adaptive Huffman coding. In the late 1980s, digital images became more common, and standards for compressing them emerged, lossy compression methods also began to be widely used In the early 1990s. Current image compression standards include: FAX CCITT 3 (run-length encoding, with code words determined by Huffman coding from a definite distribution of run lengths); GIF (LZW); JPEG (lossy discrete cosine transform, then Huffman or arithmetic coding); BMP (run-length encoding, etc.); TIFF (FAX, JPEG, GIF, etc.). Typical compression ratios currently achieved for text are around 3:1, for line diagrams and text images around 3:1, and for photographic images around 2:1 lossless, and 20:1 lossy

Revised Manuscript Received on 30 March 2013.

* Correspondence Author
Prof. Sashikala, Department Of Computer Science, St. Joseph's College (Autonomous), Langford Road, Shanthinagar, Bangalore – 560027, India.
Prof. Melwin .Y., Department Of Computer Science, St. Joseph's College (Autonomous), Langford Road, Shanthinagar, Bangalore – 560027, India.
Prof. Arunodhayan Sam Solomon, Department Of Computer Science, St. Joseph's College (Autonomous), Langford Road, Shanthinagar, Bangalore – 560027, India.
Prof. M.N.Nachappa, Department Of Computer Science, St. Joseph's College (Autonomous), Langford Road, Shanthinagar, Bangalore – 560027, India.

Transferring data is one of the major problem all of us face. The more space the file occupies, the harder it is to transfer the data. Compression is used to solve this problem by reducing the file size without affecting the quality of the original Data. Data Compression is an important part of Computer Science mainly because of the reduced data communication and the Storage cost. [**1**] digital data is usually created by a word processor as text data, a digital camera to capture pictures of audio and video data, a scanner to scan documents of text as image data or other devices, once the data is captured it is edited on a computer, then stored on a optical medium or disk permanently based on the users need, sharing information Compressed data can be transmitted through e-mail, and downloaded.

II. COMPRESSION

Compression is a technology by which one or more files or directories size can be reduced so that it is easy to handle. The Goal of compression is to reduce the number of bits required to represent data and to decrease the transmission time [2].Compression is achieved through encoding data and the data is decompressed to its original form by decoding. Compression increases the capacity of communication channel by transmitting the compressed file.

The list below contains various compressed file extensions when the file is compressed [4].

A common compressed file which is used day-today has extensions which end with *.sit, .tar, .zip*; which indicates different types of software used to compress files. For Mac's .sit is used, Linux uses .tar and PCs, use .zip as the most common extensions. Compressed files may be either a one large file or a group of files bundled together.

III. DECOMPRESSION

In order to use a compressed file, you must first decompress it. The software used to decompress depends on how the file was compressed. To decompress a .zip file you need software, such as WinZip. To decompress a .sit file, you need the Stuffit Expander program. WinZip does not decompress .sit files, but one version of StuffIt Expander can decompress both .zip and .sit files. Files ending in .sea or .exe are called self-extracting files. These are compressed files that do not require any special software to decompress. Just click on the file and it will automatically be decompressed.

.arc	.cab	.lzh	.sea	.tgz
.arj	.cpt	.mim	.sit	.uu
.as	.gz	.mme	.sitx	.uue
.b64	.hqx	.pak	.tar.gz	.z
.btoa	.iso	.pf	.tbz	.zip
.bz	.lha	.rar	.tbz2	.zoo

IV. DATA COMPRESSION

The goal of data compression is to reduce redundancy in stored or communicated data, thus increasing effective data density, and to reduce the usage of resources [3]. Data compression has important application in the areas of file storage and distributed systems. Normally, it is used in mass storage, to lower the amount of data transmitted and reduce file size for improving speed of data transmission.

Data compression is also referred to as source coding. It is widely used in backup utilities, spreadsheet applications, communication and database management systems [1]. Certain types of data, such as bit-mapped graphics, software we get in the ***Internet*** are compressed. Without data compression, it would be virtually impossible to implement technologies such as streaming media, large database applications.

There are number of compression techniques designed to store graphic images in a much smaller space.

Image compression: [7] also improves throughput when images files are transmitted from one place to another.

Video compression: [7] in video compression each frame is an array of pixels that must be reduced by removing redundant information. Video compression is usually done with special integrated circuits, rather than with software, to gain performance. Standard video is normally about 30 frames/sec, but studies have found that 16 frames/sec is acceptable to many viewers, so frame dropping provides another form of compression. MPEG (Motion Picture Experts Group), DVI (Digital Video Interactive)

Voice/Telephone Compression:[7] The original telephone system was based on an analog infrastructure. But analog transmissions are subject to transmission problems like attenuation over distance and line noise. The more attenuation, the more the line noise masks the real signal. Periodic amplification solves the problem to some extent, but any line noise introduced is also amplified with the voice signal. To solve this problem, the trunk lines of the telephone system were converted to digital and PCM (pulse code modulation) was used to digitize the signals. Later, ADPCM (Adaptive Differential Pulse Code Modulation) was used to compress the signal.

There are two compression techniques commonly used

a. Lossless compression

Lossless compression methods may be categorized according to the type of data they are designed to compress. The algorithm which can handle all binary input can be used on any type of data. But, most are unable to achieve significant compression on data which is not able to deal with a particular type. For example Sound data cannot be compressed well with conventional text compression algorithms.

Some of the commonly used **Lossless coding techniques are** Run length encoding**,** Huffman encoding, Arithmetic encoding, Entropy coding, Area coding.

b. Lossy compression.

A lossy data compression method is one in which compressing data and then decompressing it retrieves data that will be different from the original, but it is enough to be useful in some way. Lossy data compression is used frequently on the Internet and mostly in streaming media and telephony applications. In lossy data repeated compressing and decompressing a file will cause it to lose quality.

Lossless when compared with lossy data compression will retain the original quality, an efficient and minimum hardware implementation for the data compression and decompression needs to be used even though there are so many compression techniques which is faster, memory efficient which suits the requirements of the user.

V. BASIC DATA COMPRESSION ALOGORITHMS

- **Adaptive Huffman coding and Lempel ziv algorithms**

These compression techniques use a symbol dictionary to represent recurring patterns. The dictionary is dynamically updated during a compression as new patterns occur. For data transmission, the dictionary is passed to a receiving system so it knows how to decode the characters. For file storage, the dictionary is stored with the compressed file.

- **Null compression** Replaces a series of blank spaces with a compression code, followed by a value that represents the number of spaces
- **Run-length compression** replaces any series of four or more repeating characters. The characters are replaced with a compression code, one of the characters and a value that represents the number of characters to repeat. Some synchronous data transmission can be compressed by as much as 98 percent using this scheme.
- **Keyword encoding creates** a table with values that represent common sets of characters. Frequently occurring words like *for* and *the* character pairs like *sh* or *th* are represented with tokens used to store or transmit the character.

VI. EVALUATING COMPRESSION ALGORITHMS

The criteria to evaluate compression algorithms are based on their compression speed, compression ratio and computational complexity

a. Data compression ratio also known as **compression power**[2]used to quantify the reduction in data-representation size produced by a data compression algorithm. Data compression ratio is defined as the ratio between the *compressed size* and the *uncompressed size*:

Data Compression Ratio(CR) = Compressed size/ Uncompressed size

Thus a representation that compresses a 10MB file to 2MB has a compression ratio of 2/10 = 0.2, often notated as an explicit ratio, 1:5 (read "one to five"), or as an implicit ratio, 1/5.

b. Data Compression speed: speed of data compression is always in terms of uncompressed data handled per second.

Data Compression Speed = Uncompressed bits / Seconds to compress

An illustration to compress 1000 bits in 2 seconds is 1000/2=0.002, has a compression speed of 0.002 bits per second.

c. Data compression complexity: the complexity of data compression depends on [4]

- The size of the file to be compressed.
- How much ROM or disk space does the software use.
- How long does it take to go through all the source code to find the right thing to change, and once changed.

- How long does it take to test every part of the code to make sure the change doesn't break something else.

VII. TEXT DATA COMPRESSION

Compression software works by using mathematical equations to scan data file and to search for repeating patterns. Compression can be as simple as removing all extra space characters and inserting a single character to indicate a string of repeated characters.

Once the compression software has identified a repeating pattern, it replaces that pattern with a smaller code. For example the string *DDDDDAAAEE* can be reduced to *5D3A2E*. This kind of compression can reduce a text file to 50% of its original size, such as a 10 KB (Kilobyte or 1000 bytes) file can be converted to a 1 KB file.

Decompressing a file is the process by which a compressed file is converted back to its standard state, such as if our 1 KB file were converted back to a 10 KB file. The process of compression and decompression can be performed by various algorithms. These algorithms are programmed into **codecs** **(*CO*mpression/*DEC*ompression)**. The **codecs** are the mathematical formulas used to compress or decompress files so that it requires less disk space for storage and bandwidth on a transmission channel.

Figure-1 shows how compression and decompression is done on a text data. The work of the Compressor is to convert the original text at the sender's end into a compressed text (Zipping) which is then transferred to the receiver. The receiver then converts the compressed text into the original text with the help of the *DeCompressor* (Unzipping).

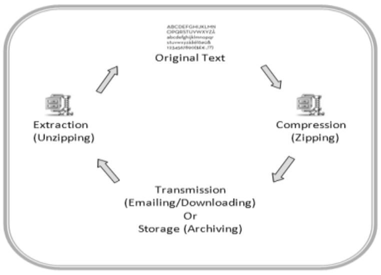


Figure-1 Text Data Compression

a. *Lossless Text Compression*

We use lossless compression for text where the compressed file must be exact bit per bit to the original file. Data is compressed without any loss of data during the compression process. When the compressed file is decompressed, the result is identical to the data before compression. This type of compression is used when storing database records, communication, spreadsheets or word processing files, computer programs, software applications, databases are compressed using lossless techniques, since a change in even one bit of data can make them useless or produce incorrect results. For most files, lossless methods reduce the file size by only 50-60%. Well known lossless compression methods include *Zip* and *RAR.*

b. *Codecs for Lossless compression*

Once these basic methods have been employed, much more intensive algorithms can be employed to reduce the amount of transmitted data. Mathematical algorithms can be used to encode and decode data these *codecs* (such as enCode and Decode).

The simplest encoding algorithm, called run-length encoding, represents strings of redundant values as a single value and a multiplier. For example, consider the following bit values:

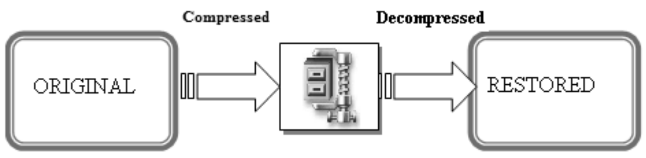
1111111000000000000000000000000111111111000000000000000000000000

Using run-length encoding on the bit values above can reduce the amount of information to:

1*7, 0 * 24, 1 * 9, 0 * 24

In the example above, the original 64 bits can be transmitted using only 18 bits.

Run-length encoding is lossless, because all the information is retained after decoding. This technique is particularly useful for computer graphics applications, because there are often large fields of identical colors.

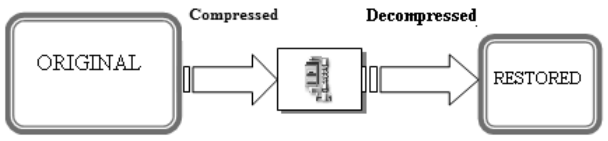


c. *Lossy text compression*

Lossy compression where some error is acceptable and in most of the cases is not detected. Lossy compression reduces file size by permanently eliminating redundant information, so that only a part of the original data is retained and reproduced[1]. Most video and audio compressors compress data in ways that are lossy, but produce very high compression levels. Properly compressed video and audio files are nearly indistinguishable from the original to the human eye or ear. Lossy compression techniques are known to produce high compression. In most of the cases signal compression goes of discarding as much data as possible but retaining as much quality as possible. Lossy compression proves effective when applied to graphics images and digitized voice. For example, WMA and MP3 are used to compress music, while JPEG and PGF are for image compression, and MPEG, H.261, H.263 and H.264 are designed for video. Many formats, including JPEG and all varieties of DV, use a fairly complicated algorithm called DCT encoding. Another method, called wavelet compression, is starting to be used for popular codecs, such as the Apple Pixlet video codec. DVDs, modern digital television, and formats such as HDV use MPEG-2 compression, which not only encodes single frames (intraframe, or spatial compression) but encodes multiple frames at once (interframe, or temporal compression) by throwing away data that is visually redundant over time.

d. *Codecs for Lossy Compression*

Since it is not possible to store and transmit uncompressed video signals, therefore most video codecs are lossy . In this case there may some loss of video signal but the goal is to make loss visually imperceptible. When lossy codecs algorithm are developed they are fine tuned based on analyses of human vision and perception. For example human eye cannot differentiate between lots of subtle variation in the color, a codec throw away some of the information that viewers may never notice.



e. Huffman Coding

Huffman code procedure is based on the two observations. More frequently occurred symbols will have shorter code words than symbol that occur less frequently. The two symbols that occur least frequently will have the same length[6]. The Huffman code is designed by merging the lowest probable symbols and this process is repeated until only two probabilities of two compound symbols are left and thus a code tree is generated and Huffman codes are obtained from labelling of the code tree.

f. Compressing files

A huffman tree[8] is a special binary tree called a trie. (pronounced try) A binary trie is a binary tree in which a 0 represents a left branch and a 1 represents a right branch. The numbers on the nodes of the binary trie represent the total frequency, F, of the tree below. The leaves of the trie represent the elements, e, to be encoded. The elements are assigned the encoding which corresponds to their place in the binary trie.

- The file is scanned from beginning to the end and the frequency of each character is stored in an array. The size of this array must be at least 2n-1 where n is the number of characters used in the file. Each element of this array represents a node of the tree.
- The first element of this array gives us the frequency of the first character. Similarly the second element gives the frequency of the second character.
- After the file is scanned, the first 256 elements will contain the frequencies of each of the 256 characters. The remaining 255 elements will be empty. If an array of 511 elements, need 2 * 256-1 = 511 elements. Since one byte can represent 256 different characters
- The next step is to find two nodes having the least frequency. Take one of the empty elements from the array and make it the parent of these two nodes.
- The frequency of the parent node is the sum of the frequencies of these two nodes. The left and right child nodes can be interchanged, it does not matter if node 10 is the left or right child of the parent node 21.
- Now find the next two nodes having least frequencies, after removing the two child nodes and adding the parent node to the search list. Continue this process until only one node is left in the search list. This node is the root nod e of the tree.

For example:

Message to be Encoded

Cad fae ead bead aec caf bad ade face café efe dea

g. Block Encoding

011 000 011 101 000 100 100 000 011 001 100 000 011 000 100 010 010 000 101

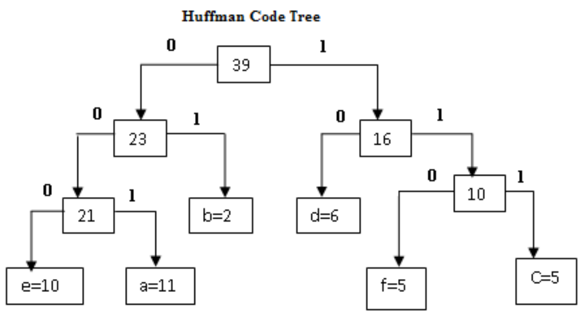
001 000 011 000 011 100 101 000 100 010 000 101 100 100 101 100 011 100 000

The block encoding above is a fixed length encoding. If a message contains i elements, block encoding requires **log i** bits to encode each element,

h. Huffman Encoding

001 10 01 110 10 111 111 10 01 000 111 10 01 10 111 001 001 10 110 000 10 01

10 01 111 110 10 001 111 001 10 110 111 111 110 111 01 111 10



To differentiate each character of string in both the Block Encoding and the Huffman Encoding Spaces have been inserted between the strings of bits they are included in the message and in the encodings for readability purposes only. They are not considered as a character in the message and therefore is not encoded as an element in the message.

The Huffman code is decoded using a Huffman tree[5]. The tree for this code is illustrated above.

The huffman encoding of the message is 100 bits long saving 17 bits over block encoding where block encoding is 117 bits long. The compression ratio is 1.17 to 1. The compression rate is 17.5%

After generating Huffman tree, the file is scanned again and character in the file is replaced by its corresponding code from the tree. Once this is done it is saved in a file in form of a table because it is required during decompression process. The size of this table depends on the file being compressed and usually ranges from 500 to 1200 bytes. The table includes character and codes for each character and is stored along with the length of each code .The frequency of the character do not have to be saved because they are not required during decompression process .

i. Decompressing Files

Before decompressing the file, we have to first regenerate the huffman tree from the table that was saved along with the compressed file.

The decompression process can be done by the following algorithm .

Step 1: The current node is set to the root node.

Step 2 : A sequence of zeros and ones are read from the compressed file. For every '0' read we move to the left child node of the current node and for every '1' read we move to the right child node of the current node and set it as the new current node.

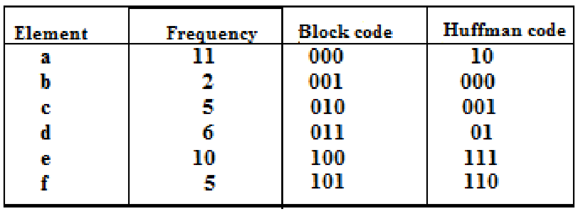
Step3 : If the current node is a leaf node then we output the character that is represented by this node. The current node is reset to the root node.

Step 4 : Steps 2 to 3 are repeated until all the bytes in the file are read.

The output file formed at the end of decompression is in uncompressed form. This compression ratio achieved by this algorithm is generally in the range of 20% to 40 %.

However recent study of compression algorithm shows LZW[11] compression algorithm which can produce much better compression ratios. So this compression algorithm is now mostly used in conjunction with other compression algorithms to achieve a better compression ratio.

VIII. CONCLUSION AND FUTURE ENHANCEMENT



Element	Frequency	Block code	Huffman code
a	11	000	10
b	2	001	000
c	5	010	001
d	6	011	01
e	10	100	111
f	5	101	110

This paper explores the design implementing generalized, lossless compression algorithms on data compression a full compression system will require three parts; an enhancement scheme, an algorithm that can predict or measure where the enhancement scheme will fail, and a system for including this information with the image file. Here we have described the last step, and we have used simple algorithms for the first two steps. In future research

ACKNOWLEDGMENT

A work of this nature requires the blessings and patience of a number of people. We would like to acknowledge the guidance of all our colleagues in this work. We would also like to thank god for his blessings and acknowledge his presence in all our efforts.

REFERENCES

1. Lynch, Thomas J., Data Compression: Techniques and Applications, Lifetime Learning Publications, Belmont, CA, 1985
2. Philip M Long., Text compression via alphabet representation
3. Cappellini, V., Ed. 1985. *Data Compression and Error Control Techniques with Applications.* Academic Press, London.
4. Cortesi, D. 1982. An Effective Text-Compression Algorithm. *BYTE 7*, 1 (Jan.), 397-403.
5. Glassey, C. R., and Karp, R. M. 1976. On the Optimality of Huffman Trees. *SIAM J. Appl. Math 31*, 2 (Sept.), 368-378.
6. Knuth, D. E. 1985. Dynamic Huffman Coding. *J. Algorithms 6*, 2 (June), 163-180.
7. Llewellyn, J. A. 1987. Data Compression for a Source with Markov Characteristics. *Computer J. 30*, 2, 149-156.
8. Pasco, R. 1976. Source Coding Algorithms for Fast Data Compression. Ph. D. dissertation, Dept. of Electrical Engineering, Stanford Univ., Stanford, Calif.
9. Rissanen, J. J. 1983. A Universal Data Compression System. *IEEE Trans. Inform. Theory 29*, 5 (Sept.), 656-664.
10. Tanaka, H. 1987. Data Structure of Huffman Codes and Its Application to Efficient Encoding and Decoding. *IEEE Trans. Inform. Theory 33*, 1 (Jan.), 154-156.
11. Ziv, J., and Lempel, A. 1977. A Universal Algorithm for Sequential Data Compression. *IEEE Trans. Inform. Theory 23*, 3 (May), 337-343.
12. Giancarlo, R., D. Scaturro, and F. Utro. 2009. Textual data compression in computational biology: a synopsis. *Bioinformatics* **25**(13): 1575-1586.
13. Operations on images in compressed domain : by Muthuraju.v & richa Sharma http://irnet.sg/irnet_journal/journal/IJCSEE/IJCSEE_Vol1Iss2/18.pdf
14. Stephen Wolfram, A New Kind of Science (Wolfram Media, 2002), page 1069.