Volume 5, Issue 1, January 2015 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

A Review on Data Compression Techniques

Amarjit Kaur
Student of M.Tech
AIET, Faridkot
Punjab, India

Navdeep Singh Sethi
Assistant Professor
AIET, Faridkot
Punjab, India

Harinderpal Singh
Assistant Professor
AIET, Faridkot
Punjab, India

Abstract: ***Data compression is a technique of reducing the size of original data and it involves encoding information using fewer bits than the original representation.In this paper, there are different basic existing data compression techniques are considered.We try to propose improved dynamic bit reduction algorithim which is based on occurrence of no. of unique symbols in input string.Improved dynamic bit reduction algorithim is designed to improve the compression ratio and memory saving percentage for text data than existing techniques.***

Keywords: DSAD, JPEG, MPEG

I. INTRODUCTION

Data compression is a process by which a file (Text, Audio, Video) may be transformed to another (compressed) file, such that the original file may be fully recovered from the original file without any loss of actual information. This process may be useful if one wants to save the storage space.
For example if one wants to store a 4MB file, it may be preferable to first compress it to a smaller size to save the storage space.
Also compressed files are much more easily exchanged over the internet since they upload and download much faster. We require the ability to reconstitute the original file from the compressed version at any time. Data compression is a method of encoding rules that allows substantial reduction in the total number of bits to store or transmit a file. The more information being dealt with, the more it costs in terms of storage and transmission costs. In short, Data Compression is the process of encoding data to fewer bits than the original representation so that it takes less storage space and less transmission time while communicating over a network [1].

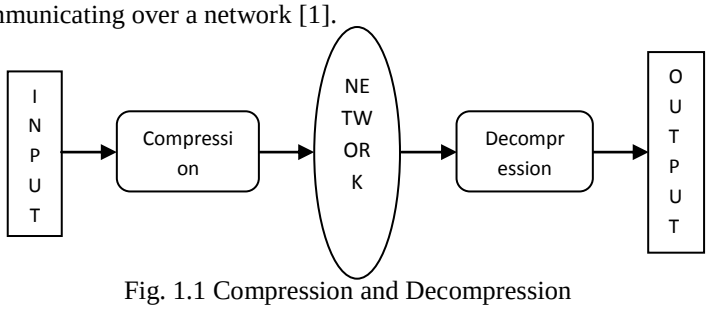

Fig. 1.1 Compression and Decompression

Data Compression is possible because most of the real world data is very redundant. Data Compression is basically defined as a technique that reduces the size of data by applying different methods that can either be Lossy or Lossless [1]. A compression program is used to convert data from an easy-to-use format to one optimized for compactness. Likewise, an uncompressing program returns the information to its original form.

II. TYPES OF DATA COMPRESSION

Currently, two basic classes of data compression are applied in different areas. One of these is lossy data compression, which is widely used to compress image data files for communication or archives purposes. The other is lossless data compression that is commonly used to transmit or archive text or binary files required to keep their information intact at any time.
There are two mainly two types of data compression:

- Lossy Compression
- Lossless Compression

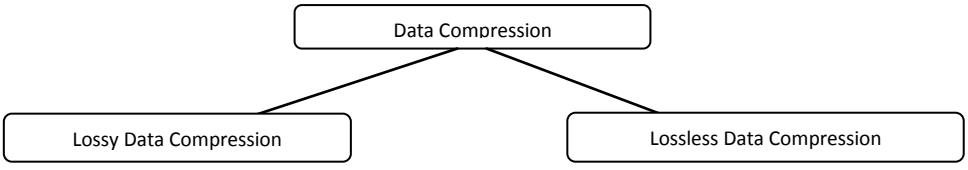

Fig: 1.3 Classification of Data Compression

© 2015, IJARCSSE All Rights Reserved

A. Lossy data compression

A lossy data compression method is one where the data retrieves after decompression may not be exactly same as the original data, but is "close enough" to be useful for specific purpose. 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 back the original message. Data has been lost. Because lossy compression cannot be decoded to yield the exact original message, it is not a good method of compression for critical data, such as textual data. It is most useful for Digitally Sampled Analog Data (DSAD). DSAD consists mostly of sound, video, graphics, or picture files. In a sound file, for example, the very high and low frequencies, which the human ear cannot hear, may be truncated from the file.

Fig: 1.4 Lossy Data Compression

The examples of frequent use of Lossy data compression are on the Internet and especially in the streaming media and telephony applications. Some examples of lossy data compression algorithms are JPEG, MPEG, MP3.Most of the lossy data compression techniques suffer from generation loss which means decreasing the quality of text because of repeatedly compressing and decompressing the file. Lossy image compression can be used in digital cameras to increase storage capacities with minimal degradation of picture quality.

B. Lossless data compression

Lossless data compression is a technique that allows the use of data compression algorithms to compress the text data and also allows the exact original data to be reconstructed from the compressed data. This is in contrary to the lossy data compression in which the exact original data cannot be reconstructed from the compressed data. The popular ZIP file format that is being used for the compression of data files is also an application of lossless data compression approach. 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. For instance, In English text, the letter 'a' is much more common than the letter 'z', and the probability that the letter 't' will be followed by the letter 'z' is very small. So this type of redundancy can be removed using lossless compression. Lossless compression methods may be categorized according to the type of data they are designed to compress. Compression algorithms are basically used for the compression of text, images and sound. Most lossless compression programs use two different kinds of algorithms: one which generates a statistical model for the input data and another which maps the input data to bit strings using this model in such a way that frequently encountered data will produce shorter output than improbable(less frequent) data.

Fig: 1.5 Lossless Data Compression

The advantage of lossless methods over lossy methods is that Lossless compression results are in a closer representation of the original input data. The performance of algorithms can be compared using the parameters such as Compression Ratio and Saving Percentage. In a lossless data compression file the original message can be exactly decoded. Lossless data compression works by finding repeated patterns in a message and encoding those patterns in an efficient manner. For this reason, lossless data compression is also referred to as redundancy reduction. Because redundancy reduction is dependent on patterns in the message, it does not work well on random messages. Lossless data compression is ideal for text.

III. EXISTING LOSSLESS DATA COMPRESSION TECHNIQUES

The existing data compression techniques are described as follow:

A. Bit Reduction algorithm

The main idea behind this program is to reduce the standard 7-bit encoding to some application specific 5-bit encoding system and then pack into a byte array. This method reduces the size of a string considerably when the string is lengthy and the compression ratio is not affected by the content of the string [1]. Bit Reduction Algorithm in steps-

- Select the frequently occurring characters from the text file which are to be encoded and obtain their corresponding ASCII code.
- Obtain the corresponding binary code of these ASCII key codes for each character.

© 2015, IJARCSSE All Rights Reserved

- Then put these binary numbers into an array of byte (8bit array).
- Remove extra bits from binary no like extra 3 bits from the front.
- Then rearrange these into array of byte and maintain the array.
- Final text will be encoded and as well as compression will be achieved.
- Now decompression will be achieved in reverse order at the client-side [1].

B. Huffman Coding

Huffman coding deals with data compression of ASCII characters. It follows top down approach means the binary tree is built from the top down to generate an optimal result. In Huffman Coding the characters in a data file are converted to binary code and the most common characters in the file have the shortest binary codes, and the characters which are least common have the longest binary code [2]. A Huffman code can be determined by successively constructing a binary tree, whereby the leaves represent the characters that are to be encoded. Every node contains the relative probability of occurrence of the characters belonging to the sub tree beneath the node. The edges are labeled with the bits 0 and 1. The algorithm to generate Huffman code is:

1. Parse the input and count the occurrence of each symbol.
2. Determine the probability of occurrence of each symbol using the symbol count.
3. Sort the symbols according to their probability of occurrence, with the most probable first.
4. Then generate leaf nodes for each symbol and add them to a queue.
5. Take two least frequent characters and then logically group them together to obtain their combined frequency that leads to the construction of a binary tree structure.
6. Repeat step 5 until all elements are reached and there remains only one parent for all nodes which is known as root.
7. Then label the edges from each parent to its left child with the digit 0 and the edge to right child with 1. Tracing down the tree yields to "Huffman codes" in which shortest codes are assigned to the character with greater frequency [2].

C. Run length Encoding

Data often contains sequences of identical bytes. By replacing these repeated byte sequences with the number of occurrences, a substantial reduction of data can be achieved. This is known as Run-length Encoding. Run-Length Encoding is a simple data compression algorithm which is supported by bitmap file formats such as BMP. RLE basically compresses the data by reducing the physical size of a repeating string of characters. This repeating string is called a run [3] which is typically encoded into two bytes where the first byte represents the total number of characters in the run and is called the run count and it replaces runs of two or more of the same character with a number which represents the length of the run which will be followed by the original character and single characters are coded as runs of 1. Run-length coding is a generalization of zero suppression, which assumes that just one symbol appears particularly often in sequences. The blank (space) character in text is such a symbol; single blanks or pairs of blanks are ignored. Starting with sequences of three bytes, they are replaced by an M-byte and a byte specifying the number of blanks in the sequence. RLE is useful where redundancy of data is high or it can also be used in combination with other compression techniques also.

Here is an example of RLE:
Input: YYYBBCCCCDEEEEEERRRRRRRRRR
Output: 3Y2B4C1D6E10R
The drawback of RLE algorithm is that it cannot achieve the high compression ratios as compared to another advanced compression methods, but the advantage of RLE is that it is easy to implement and quick to execute thus making it a good alternative for a complex compression algorithm.

D. Shannon-Fano coding

This is one of an earliest technique for data compression that was invented by Claude Shannon and Robert Fano [3] in 1949.In this technique, a binary tree is generated that represent the probabilities of each symbol occurring. The symbols are ordered in a way such that the most frequent symbols appear at the top of the tree and the least likely symbols appear at the bottom .
The algorithm for Shanon- Fano coding is:

1. Parse the input and count the occurrence of each symbol.
2. Determine the probability of occurrence of each symbol using the symbol count.
3. Sort the symbols according to their probability of occurrence, with the most probable first.
4. Then generate leaf nodes for each symbol.
5. Divide the list in two while keeping the probability of the left branch roughly equal to those on the right branch.
6. Prepend 0 to the left node and 1 to the right node codes.
7. Recursively apply steps 5 and 6 to the left and right sub trees until each node is a leaf in the tree.

Generally, Shannon-Fano coding does not guarantee the generation of an optimal code. Shannon – Fano algorithm is more efficient when the probabilities are closer to inverses of powers of 2 [3].

© 2015, IJARCSSE All Rights Reserved

E. Arithmetic Coding

Arithmetic coding is an optimal entropy coding technique as it provides best compression ratio and usually achieves better results than Huffman Coding. It is quite complicated as compared to the other coding techniques. When a string is converted in to arithmetic encoding, the characters having maximum probability of occurrence will be stored with fewer bits and the characters that do not occur so frequently will be stored with more bits, resulting in fewer bits used overall. Arithmetic coding converts the stream of input symbols into a single floating point number as output [3]. Unlike Huffman coding, arithmetic coding does not code each symbol separately. Each symbol is instead coded by considering all prior data. Thus a data stream encoded in this fashion must always be read from the beginning. Consequently, random access is not possible. Here is an algorithm to generate the arithmetic code:

1. Calculate the number of unique symbols in the input. This number represents the base b (e.g. base 2 is binary) of the arithmetic code.
2. Assign values from 0 to b to each unique symbol in the order they appear.
3. Using the values from step 2, the symbols are replaced with their codes in the input.
4. Convert the result from step 3 from base b to a sufficiently long fixed-point binary number to preserve precision.
5. Record the length of the input string somewhere in the result as it is needed for decoding.

F. Lempel-Ziv-Welch (LZW) Algorithm

The Lempel-Ziv-Welch algorithm was created in 1984 by Terry Welch. LZW is a general compression algorithm capable of working on almost any type of data. LZW compression creates a table of strings commonly occurring in the data being compressed, and replaces the actual data with references into the table. The table is formed during compression at the same time at which the data is encoded and during decompression at the same time as the data is decoded [4].

IV. CONCLUSION

In this paper we have present the various technique to compress the text data in lossless manner. Various techniques along with their algorithms and disadvantages has been proposed in this paper. It is shown that no algorithm does not provide promising result that can be used in practical applications to compress the data. Hence in future there is a need to develop an lossless text compression algorithm that can compress the text data in the better way which can also be used in various practical application where compression of text data is required.

REFERENCES

[1] R.S. Brar and B.Singh, "A survey on different compression techniques and bit reduction algorithm for compression of text data" International Journal of Advanced Research in Computer Science and Software Engineering (IJARCSSE) Volume 3, Issue 3, March 2013

[2] S. Porwal, Y. Chaudhary, J. Joshi and M. 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] S. Shanmugasundaram and R. Lourdusamy, "A Comparative Study of Text Compression Algorithms" International Journal of Wisdom Based Computing, Vol. 1 (3), December 2011

[4] S. Kapoor and A. Chopra, "A Review of Lempel Ziv Compression Techniques" IJCST Vol. 4, Issue 2, April - June 2013

[5] I. M.A.D. Suarjaya, "A New Algorithm for Data Compression Optimization", (IJACSA) International Journal of Advanced Computer Science and Applications, Vol. 3, No. 8, 2012, pp.14-17

[6] S.R. Kodituwakku and U. S. Amarasinghe , "Comparison Of Lossless Data Compression Algorithms For Text Data" Indian Journal of Computer Science and Engineering Vol1No. 4 416-425

[7] R. Kaur and M. Goyal, "An Algorithm for Lossless Text Data Compression" International Journal of Engineering Research & Technology (IJERT), Vol. 2 Issue 7, July - 2013

[8] H.Altarawneh and M. Altarawneh, " Data Compression Techniques on Text Files: A Comparison Study " International Journal of Computer Applications, Volume 26– No.5, July 2011

[9] U. Khurana and A. Koul, "Text Compression And Superfast Searching" Thapar Institute Of Engineering and Technology, Patiala, Punjab, India-147004

[10] A. Singh and Y. Bhatnagar, " Enhancement of data compression using Incremental Encoding" International Journal of Scientific & Engineering Research, Volume 3, Issue 5, May-2012

[11] A.J Mann, "Analysis and Comparison of Algorithms for Lossless Data Compression" International Journal of Information and Computation Technology, ISSN 0974-2239 Volume 3, Number 3 (2013), pp. 139-146

[12] K. Rastogi, K. Sengar, "Analysis and Performance Comparison of Lossless Compression Techniques for Text Data" International Journal of Engineering Technology and Computer Research (IJETCR) 2 (1) 2014, 16-19

[13] M. Sharma, "Compression using Huffman Coding " IJCSNS International Journal of Computer Science and Network Security, VOL.10 No.5, May 2010

[14] S.Shanmugasundaram and R. Lourdusamy, " IIDBE: A Lossless Text Transform for Better Compression " International Journal of Wisdom Based Computing, Vol. 1 (2), August 2011

[15] P. Kumar and A.K Varshney, " Double Huffman Coding " International Journal of Advanced Research in Computer Science and Software Engineering (IJARCSSE) Volume 2, Issue 8, August 2012

© 2015, IJARCSSE All Rights Reserved

[16] R. Gupta, A. Gupta, S. Agarwal, "A Novel Data Compression Algorithm For Dynamic Data" IEEE REGION 8 SIBIRCON

[17] A. Kattan, "Universal Intelligent Data Compression Systems: A Review" 2010 IEEE

[18] M. H Btoush, J. Siddiqi and B. Akhgar, "Observations on Compressing Text Files of Varying Length " Fifth International Conference on Information Technology: New Generations, 2008 IEEE

[19] A.Jain, R.Patel, " An Efficient Compression Algorithm (ECA) for Text Data International Conference on Signal Processing Systems, 2009 IEEE

[20] Md. R. Hasan, "Data Compression using Huffman based LZW Encoding Technique" International Journal of Scientific & Engineering Research, Volume 2, Issue 11,November-2011, pp.1-7

[21] M. Gupta, B. Kumar, "Web Page Compression using Huffman Coding Technique" International Conference on Recent Advances and Future Trends in Information Technology (iRAFIT2012) Proceedings published in IJCA, International Journal of Computing Applications

[22] P. Yellamma, N. Challa, " Performance Analysis Of Different Data Compression Techniques On Text File" International Journal of Engineering Research & Technology (IJERT), Vol. 1 Issue 8, October – 2012, pp.1-6

© 2015, IJARCSSE All Rights Reserved