Data Compression using Huffman based LZW Encoding Technique

Md. Rubaiyat Hasan

ABSTRACT
Data compression is of interest in business data processing, both because of the cost savings it offers and because of the large volume of data manipulated in many business applications. A method and system for transmitting a digital image (i.e., an array of pixels) from a digital data source to a digital data receiver. More the size of the data be smaller, it provides better transmission speed and saves time. In this communication we always want to transmit data efficiently and noise free. Both the LZW and Huffman data compression methods are lossless in manner. These methods or some versions of them are very common in use of compressing different types of data. Even though on average Huffman gives better compression results, it determines the case in which the LZW performs best and when the compression efficiency gap between the LZW algorithm and its Huffman counterpart is the largest. In the case of Hybrid compression it gives better compression ratio than in single compression. So, at first I wanted to compress original data by Huffman Encoding Technique then by the LZW Encoding Technique .But it did not give better compression ratio than in single LZW compression. At that time I have found that if we compress the data by Huffman first and then by LZW all the cases it gives better compression ratio. Then it named as "Data compression using Huffman based LZW Encoding". Its compression ratio most of the cases above 2.55 and in some cases it becomes above 3.25 or more. It will provide cheap, reliable and efficient system for data compression in digital communication system.

Index Terms—Double Compression, Huffman based LZW Encoding, Data Compression, and Software Tools.

1 INTRODUCTION

The term "Data Compression" means compresses the data as much as possible from its original size. But in this communication we always want to transmit data efficiently and noise free. We try to find such data that are lossless. Huffman and LZW both are lossless technique of data compression. Hybrid compression is also helpful in data compression. There are various hybrid data compressions.

A brief introduction to information theory is provided in this chapter. The definitions and the assumptions necessary to a comprehensive discussion and evaluation of data compression methods are discussed. Data compression is of interest in business data processing, both because of the cost savings it offers and because of the large volume of data manipulated in many business applications. The types of local redundancy present in business data files include runs of zeros in numeric fields and sequences of blanks in alphanumeric fields which are present in some records and null in others.

Md. Rubaiyat Hasan
Bachelor in Computer Science and Engineering from Rajshahi University of Engineering and Technology (RUET), Bangladesh in 2009,
E-mail: mdrubaiyat@yahoo.com

Run length encoding can be used to compress sequences of zeros or blanks. Null suppression may be accomplished through the use of presence bits. Another class of methods exploits cases in which only a limited set of attribute values exist. Dictionary substitution entails replacing alphanumeric representations of information such as bank account type, insurance policy type, sex, month, etc. by the few bits necessary to represent the limited number of possible attribute values.

Cormack describes a data compression system which is designed for use with database files. The method, which is part of IBM's "Information Management System" (IMS), compresses individual records and is invoked each time a record, is stored in the database file; expansion is performed each time a record is retrieved. Since records may be retrieved in any order, context information used by the compression routine is limited to a single record. In order for the routine to be applicable to any database, it must be able to adapt to the format of the record. The fact that database records are usually heterogeneous collections of small fields indicates that the local properties of the data are more important than its global characteristics.

The compression routine in IMS is a hybrid method which attacks this local redundancy by using different coding schemes for different types of fields. The identified field

types in IMS are letters of the alphabet, numeric digits, packed decimal digit pairs, blank, and other. When compression begins, a default code is used to encode the first character of the record. For each subsequent character, the type of the previous character determines the code to be used.
For example, if the record 01870_ABCD___LMN were encoded with the letter code as default, the leading zero would be coded using the letter code; the 1, 8, 7, 0 and the first blank (_) would be coded by the numeric code. The A would be coded by the blank code; B, C, D, and the next blank by the letter code; the next blank and the L by the blank code; and the M and N by the letter code. Clearly, each code must define a codeword for every character; the letter code would assign the shortest code words to letters, the numeric code would favor the digits, etc. In the system Cormack describes, the types of the characters are stored in the encode/decode data structures.

When a character c is received, the decoder checks type(c) to detect which code table will be used in transmitting the next character. The compression algorithm might be more efficient if a special bit string were used to alert the receiver to a change in code table. Particularly if fields were reasonably long, decoding would be more rapid and the extra bits in the transmission would not be excessive. Cormack reports that the performance of the IMS compression routines is very good; at least fifty sites are currently using the system. He cites a case of a database containing student records whose size was reduced by 42.1%, as a side effect the number of disk operations required to load the database was reduced by 32.7%.

A variety of approaches to data compression designed with text files in mind include use of a Dictionary either representing all of the words in the file so that the file itself is coded as a list of pointers to dictionary or representing common words and word endings so that the file consists of pointers to dictionary and encodings of the less common words. Hand-selection of common phrases, programmed selection of prefixes and suffixes and programmed selection of common character pairs has also been investigated.
The remainder of the paper is organized as follows. In Section 2, the relevant literature on data compression is reviewed. Section 3, shows several possible ways to parallelize the BWT algorithm. Section 4, presents the parallel bzip2 algorithm. Section 5 gives the performance results using the algorithm on several parallel architectures. Section 6 concludes the paper.

1.1 Scope of Research

We know about the term compression ratio. This means as:
Compression Ratio , *Cr*= ((Data size of original message)/(Data size of Encoded message))

Now-a-days, compression ratio is a great factor in transmission of data. By this research we can have a better solution about how to make compression ratio higher, because data transmission mostly depends on compression ratio.

This discussion of semantic dependent data compression techniques represents a limited sample of a very large body of research. These methods and others of a like nature are interesting and of great value in their intended domains. Their obvious drawback lies in their limited utility. It should be noted, however, that much of the efficiency gained through the use of semantic dependent techniques can be achieved through more general methods, albeit to a lesser degree. For example, the dictionary approaches can be implemented through either Huffman coding or Lempel-Ziv codes. Cormack's database scheme is a special case of the codebook approach, and run length encoding is one of the effects of Lempel-Ziv codes.

1.2 Objective

The main Objective is to provide better compression ratio. For these here I have used Huffman compression and LZW compression. The objective can be specified as follows:

Ø Study the different data compression techniques
Ø Implementation of the Huffman and LZW Encoding
Ø Measure the performance of Huffman and LZW by my measuring various size of data
Ø Then measure relative improved performance in the case of hybrid compression for both Huffman based LZW and LZW based Huffman
Ø Provide recommendation for future aspect

2 DATA COMPRESSION TECHNIQUES

2.1 Data Compression Types

Mainly there are two different ways that data compression algorithms can be categorized. In (table 2.1), the methods have been classified as either **lossless** or **lossy**. These are lossless and lossy(does loss any single data). In (table 2.2), the methods are classified according to a fixed or variable size of group taken from the original file and written to the compressed file.

Lossless	Lossy
run-length	CS&Q
Huffman	JPEG
delta	MPEG
LZW	

Table 2.1: Lossless or lossy.

Method	Input	output
CS&Q	Fixed	fixed
Huffma	Fixed	variable
Arithmet	Variable	variable
run-length, LZW		Variable

Table 2.2: Fixed or variable group size for the source messages.

2.1.1 Huffman Encoding

In computer science and information theory, Huffman coding is an entropy encoding algorithm used for lossless data compression. The term refers to the use of a variable length code table for encoding a source symbol (such as a character in a file) where the variable-length code table has been derived in a particular way based on the estimated probability of occurrence for each possible value of the source symbol. It was developed by David A. Huffman while he was a Ph.D. student at MIT, and published in the 1952 paper "A Method for the Construction of Minimum-Redundancy Codes." Huffman became a member of the MIT faculty upon graduation and was later the founding member of the Computer Science Department at the University of California, Santa Cruz, now a part of the Baskin School of Engineering.

2.1.2 Huffman Encoding Algorithm

Huffman (W, n) //Here, W means weight and n is the no. of inputs

Input: A list W of n (Positive) Weights.

Output: An Extended Binary Tree T with Weights Taken from W that gives the minimum weighted path length.

Procedure: Create list F from singleton trees formed from elements of W.

While (F has more than 1 element) **do**

Find T1, T2 in F that have minimum values associated with their roots // T1 and T2 are sub tree

Construct new tree T by creating a new node and setting T1 and T2 as its children

Let, the sum of the values associated with the roots of T1 and T2 be associated with the root of T Add T to F

Do

Huffman-Tree stored in F

2.2.1 LZW Encoding

LZW compression is named after its developers, A. Lempel and J. Ziv, with later modifications by Terry A. Welch. It is the foremost technique for general purpose data compression due to its simplicity and versatility. Typically, you can expect LZW to compress text, executable code, and similar data files to about one-half their original size. LZW also performs well when presented with extremely redundant data files, such as tabulated numbers, computer

LZW is the basis of several personal computer utilities that claim to "double the capacity of your hard drive." If the codeword length is not sufficiently large, Lempel-Ziv codes may also rise slowly to reasonable efficiency, maintain good performance briefly, and fail to make any gains once source code, and acquired signals. Compression ratios of 5:1 are common for these cases.

2.2.2 LZW Encoding Algorithm

Step 1: At the start, the dictionary contains all possible roots, and P is empty;

Step 2: C: = next character in the char stream;

Step 3: Is the string P+C present in the dictionary?

(a) if it is, P := P+C (extend P with C);

(b) if not,

–output the code word which denotes P to the code stream;

– add the string P+C to the dictionary;

–P := C (P now contains only the character C); (c) Are there more characters in the charstream?

–if yes, go back to step 2;

–if not:

Step 4: Output the code word which denotes P to the code stream;

Step 5: END.

3 RESULTS AND DISCUSSION

3.1 Result of Encoded data and compression ratio using Huffman

Table 3.1: Result of Data compression using Huffman Encoding

No.of Experiment	Original Data Size	Huffman	Compr ession Ratio
01	4.70 MB	2.67 MB	1.7
02	2.19 MB	1.20 MB	1.8
03	20	18.5 KB	1.4
04	71.6 KB	48	1.4
05	2.44 MB	1.40 MB	1.7
06	4.63 MB	2.61 MB	1.7
07	9.53 MB	5.42 MB	1.7
08	18.8 MB	10.6 MB	1.7
09	7.08 MB	4.02 MB	1.7
10	25.9 MB	14.6 MB	1.7

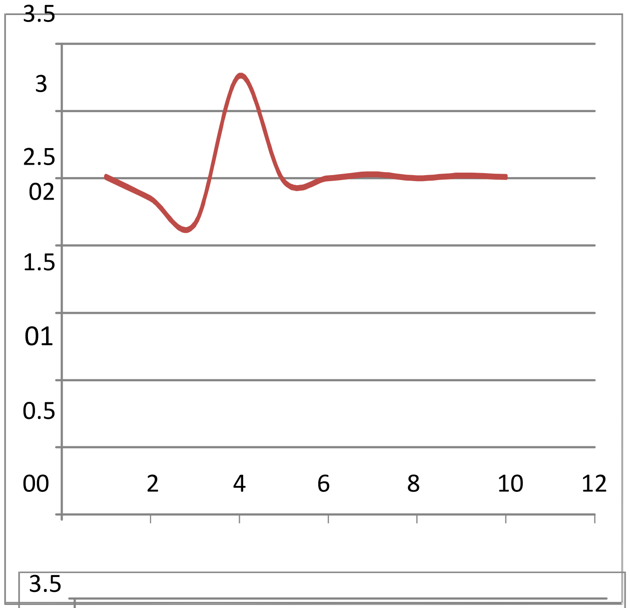
From the above experiment this is found that every data has decreased from its original size. For the case of both MB and KB data the compression ratio is above 1.7 in average case. There are lower and higher compression ratios in some cases. This is the result of compression of text file. The compressed files were in the 0, 1 binary code formats. After that when decompressed that file it becomes same.

3.2 Result of Encoded data and Compression ratio using LZW

No. of Exp.	Original Data Size	LZW	Compression Ratio(Cr)
01	4.70 MB	1.87 MB	2.51
02	2.19 MB	933 KB	2.35
03	26 KB	12 K	2.16
04	71.6 KB	21.9 KB	3.26
05	2.44 MB	980 KB	2.48
06	4.63 MB	1.85 MB	2.50
07	9.53 MB	3.77 MB	2.53
08	18.8 MB	7.51 MB	2.50
09	7.08 MB	2.81 MB	2.52
10	25.9 MB	10.3 MB	2.51

Table3.2: Data compression using LZW Encoding

From the above experiment it is found that every data has decreased from its original size. For the case of both MB and KB data the compression ratio is above 2.5 in average case. There are lower and higher compression ratios in some cases. This is the result of compression of text file. After that when decompressed that file it becomes same as the original file. It gives better compression ratio than the case of Huffman Encoding. It compresses a file almost 60% and we can have only 40% size of the original file size. To have better file size we may use LZW Encoding technique.



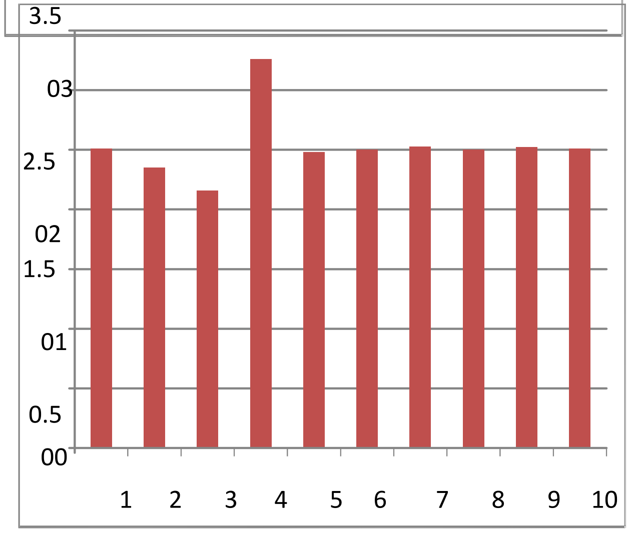


Figure 3.1: Graphical view of result of Data compression using LZW Encoding

3.3 Result of Encoded data and Compression ratio using Huffman based LZW

No. of Experiment	Original Data Size	Huffman based LZW	Compression Ratio(Cr)
01	4.70 MB	1.83 MB	2.56
02	2.19 MB	912 KB	2.40
03	26 KB	11.5 KB	2.26
04	71.6 KB	21 KB	3.41
05	2.44 MB	959 KB	2.54
06	4.63 MB	1.82 MB	2.54
07	9.53 MB	3.69 MB	2.58
08	18.8 MB	7.35 MB	2.55
09	7.08 MB	2.75 MB	2.57
10	25.9 MB	10.1 MB	2.56

Table 3.3: Result of Data compression using Huffman based LZW Encoding

From the above experiment it is found that every data has decreased from its original size. For the case of both MB and KB data the compression ratio is above 1.80 in average case. There are lower and higher compression ratios in some cases. This is the result of compression of text file.

After that when decompressed that file it becomes same as the original file. It gives better compression ratio than the case of Huffman Encoding but lower compression ratio in the case of LZW. It compresses a file almost 55% and we can have only 45% size of the original file size. We think that Hybrid compression gives better output but here we find different case. So, LZW performs better than LZW based Huffman Encoding .To have better file size we may use LZW Encoding technique.

3.4 Result of Encoded data and Compression ratio using LZW base Huffman

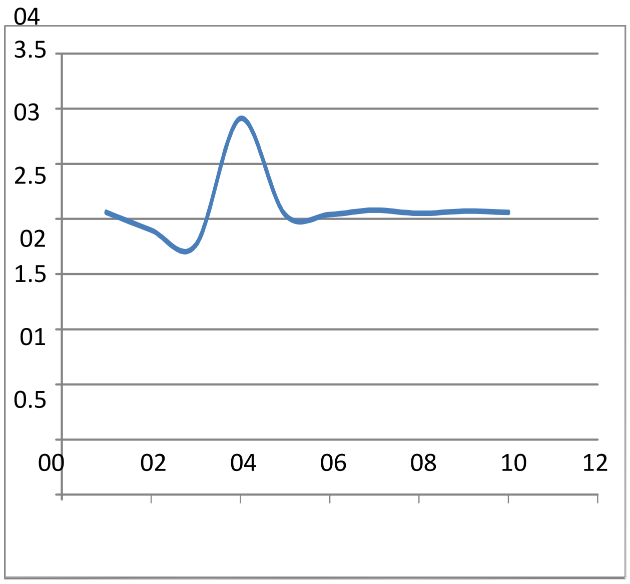
No.of Experiment	Original Data Size	LZW based Huffman	Compression Ratio(Cr)
01	4.70 MB	2.59 MB	1.81
02	2.19 MB	1.24 MB	1.76
03	26 KB	21.1 KB	1.23
04	71.6 KB	39.1 KB	1.83
05	2.44 MB	1.32 MB	1.85
06	4.63 MB	2.54 MB	1.82
07	9.53 MB	5.30 MB	1.79
08	18.8 MB	10.5 MB	1.79
09	7.08 MB	3.93 MB	1.80
10	25.9 MB	14.4 MB	1.79

Table 3.4: Result of Data compression using LZW based Huffman Encoding

3.5 Result of Encoded data and Compression ratio using Huffman based LZW

From the above experiment we find that every data has decreased from its original size. For the case of both MB and KB data the compression ratio is above 2.55 in average case. There are lower and higher compression ratios in some cases. This is the result of compression of text file. After that when we have decompressed that file it becomes same as the original file. It gives better compression ratio than the case of Huffman, LZW and LZW based Huffman all three Encoding.

It compresses a file almost 62% and we can have only 38% size of the original file size. From this experiment we can say that Hybrid Data Compression is better than single compression. To have better file size we may use Huffman based LZW Encoding technique.



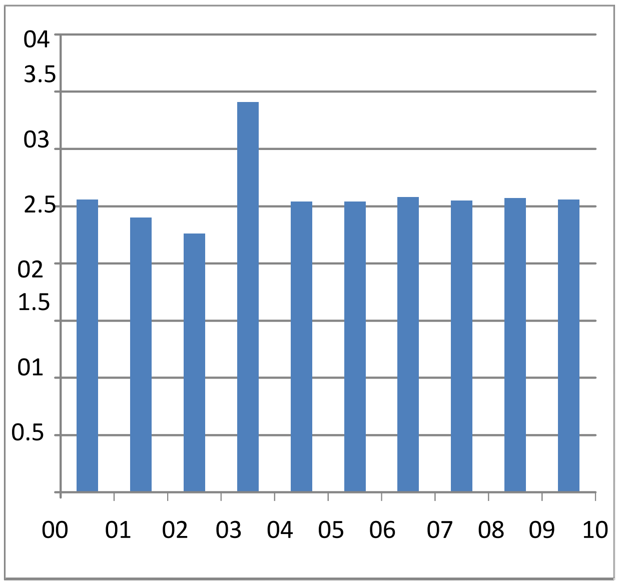


Figure 3.2: Graphical view of result of Data compression using Huffman based LZW Encoding

3.6 Comparison among relative compression ratio

No. of Exp.	Original Data Size	Huffman	LZW	LZW based Huffman	Huffman based LZW
01	4.70 MB	2.67 MB	1.87 MB	2.59 MB	1.83 MB
02	2.19 MB	1.20 MB	933 KB	1.24 MB	912 KB
03	26 KB	18.5 KB	12 KB	21.1 KB	11.5 KB
04	71.6 KB	48 KB	21.9 KB	39.1 KB	21 KB
05	2.44 MB	1.40 MB	980 KB	1.32 MB	959 KB
06	4.63 MB	2.61 MB	1.85 MB	2.54 MB	1.82 MB
07	9.53 MB	5.42 MB	3.77 MB	5.30 MB	3.69 MB
08	18.8 MB	10.6 MB	7.51 MB	10.5 MB	7.35 MB
09	7.08 MB	4.02 MB	2.81 MB	3.93 MB	2.75 MB
10	25.9 MB	14.6 MB	10.3 MB	14.4 MB	10.1 MB

Table 3.5: Result of Data compression in different cases

This is the result of relative data compression among Huffman, LZW, LZW based Huffman and Huffman based LZW. In all these cases of compression Huffman based LZW performs well. It compresses a data to almost 38% or less from its original size where the remaining three never gives such better compressed data.

4 CONCLUSION

Data compression is a topic of much importance and many applications. Methods of data compression have been studied for almost four decades. This paper has provided an overview of data compression methods of general utility. The algorithms have been evaluated in terms of the amount of compression they provide, algorithm efficiency, and susceptibility to error. While algorithm efficiency and susceptibility to error are relatively independent of the characteristics of the source ensemble, the amount of compression achieved depends upon the characteristics of the source to a great extent.

It does not matter how the characters are arranged. It arranged above so that the final code tree looks nice. It is used Turbo

C editor for coding. It does not matter how the final code tree are labeled (with 0s and 1s).It has chosen to label the upper branches with 0s and the lower branches with 1s.

In conclusion anybody can say that Huffman based LZW Encoding Compresses data more than Huffman Compression in most of the case. The Huffman based LZW compression is better some of the cases than LZW Compression. This is also possible to Decompress.

5 ACKNOWLEDGEMENT

My special thanks to Prof. Md. Robiul Islam, RUET, Bangladesh and Dr. Md. Shahid Uz Zaman for their suggestions.

6 APPENDIX

USED TOOLS AND PROGRAMS

I have used various tools for Compression purpose. I also used two different programs on LZW and Huffman to fulfill my Compression purpose. These are as follows:

Ø I have used Turbo C editor for coding.
Ø It does not matter how the final code tree are labeled. We have used 0 and 1 for code tree.
Ø I have chosen to label the upper branches with 0s and the lower branches with 1s for Huffman.
Ø For LZW we have used 0-256 for single word. For string or multi word we have used above the 256 numbers.
Ø I have used text files as the source file.
Ø I have compressed a text by LZW Encoding first then we apply Huffman to the LZW Compressed data. Finally, we obtain the double compressed data. It can be decompressed by Huffman first and then by LZW Decoding Technique.

7 REFERENCES

[1] **Digital Image Processing** (Second papers: Edition)-2006 –Rafael C. Gonzalez and Richard E. Woods. Pages: 411, 440-442, 459
[2] **Lecture 17on Data Compression**: Patrick Karlsson Patrick. karlsson@cb.uu.se Center for Image Analysis. Uppsala University, Computer Assisted Image Analysis May 19 2006
[3] **Chesnokov Yuriy-**Former Cambridge University post -doc(http://www-ucc.ch.cam.ac.uk/research/yc274-research.html)
currently lives in Krasnodar, Russia and doing some contract research for third parties.
[4] **Mark Nelson's** "LZW Data Compression" from the October, 1989 issue of *Dr. Dobb's Journal*.
[5] **Huffman's original article**:D.A. Huffman, "[3]" (PDF), Proceedings of the I.R.E., September 1952, pp1098-1102
[6] **Background story Profile:**David A. Huffman, Scientific American, Sept. 1991, pp. 54-58
[7] Thomas H. Cormen , Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. Introduction to http://en.wikipedia.org/wiki/Huffman_coding22-Aug-07
[8] **Abramson, N. 1963**. *Information Theory and Coding*. McGraw-Hill, New York.
[9] **Apostolico, A. and Fraenkel, A. S. 1985.** Robust Transmission of Unbounded Strings Using Fibonacci Representations. Tech. Rep. CS85-14, Dept. of Appl. Math., The Weizmann Institute of Science, Sept.
[10] **Elias, P. 1975.** Universal Code word Sets and Representations of the Integers. Trans. Inform.
Theory 21, 2 (Mar.), 194-203.
[11] **Cappellini, V., Ed. 1985.** Data Compression and Error Control Techniques with Applications. Academic
Press, London.
[12] **Connell, J. B. 1973.** A Huffman-Shannon-Fano Code. *Proc. IEEE 61*, 7 (July), 1046-1047.
[13] **Cormack, G. V. 1985.** Data Compression on a Database System. Commun. *ACM 28*, 12 (Dec.), 1336-42.
[14] **Cormack, G. V., and Horspool, R. N. 1984.** Algorithms for Adaptive Huffman Codes. Inform. Process. Letts. *18*, 3 (Mar.),159-65.
[15] **Cortesi, D. 1982.** An Effective Text-Compression Algorithm. *BYTE 7*, 1 (Jan.), 397-403.
[16] **Cot, N. 1977.** Characterization and Design of Optimal Prefix Codes. Ph. D. dissertation, Computer Science Dept.,StanfordUniv.,Stanford,Calif.