Performance Comparison of Huffman Coding and Double Huffman Coding

Rabia Arshad
Department of Electrical Engineering
The University of Lahore
Lahore, Pakistan
rabia.arshad@ee.uol.edu.pk

Adeel Saleem
Department of Electrical Engineering
The University of Lahore
Lahore, Pakistan
adeel.saleem@ee.uol.edu.pk

Danista Khan
Department of Electrical Engineering
The University of Lahore
Lahore, Pakistan
danista.khan@ee.uol.edu.pk

Abstract- Huffman coding [11] is a most popular technique for generating prefix-free codes [7, 10]. It is an efficient algorithm in the field of source coding. It produces the lowest possible number of code symbols of a single source symbol [1]. Huffman coding is a most widely used lossless compression technique [2]. However, there are some limitations that arise in Huffman coding [20, 21]. This method produces a code of few bits for a symbol having high probability of occurrence and large number of bits for a symbol having low probability of occurrence [3]. Instead of this, in Double Huffman Coding when the code word of the symbol has been generated it will be compressed on binary basis. Through this technique a better result be achieved. In this paper we discussed the technique of Huffman Coding and Double Huffman coding and compare their performance analysis.

Keywords: **Compression, Huffman Coding, Huffman Tree, Entropy, Redundancy**.

I. Introduction

Lossless compression [9] means the method of data compression [12, 17] in which we can recover original uncompressed data [8] from the compressed file. In the field of computer science, this compression method is widely used for compressing text, images and many other data. In general, overall compression procedure might be a mixture of lossy compression and lossless compression [4]. Huffman algorithm is an optimal lossless method of data compression [5]. In Huffman coding the symbols with high probability of occurrence are more emphasized [6]. This method of compression is efficient for data for which frequency of occurrence is high. Instead through Double Huffman Coding a string of symbol can be created after concatenation. Concatenation can be ordered alphabetically or numerically. Double Huffman Coding then compresses this string using 0 and 1 probability.

II. Huffman Coding And Double Huffman Coding

The Huffman algorithm [15, 16] works from leaves to the root in the opposite direction.

i. Make a leaf node for each symbol given. Assign it to frequency of occurrence.
ii. Combine two nodes of lowest frequency.
iii. Assign one node a code of 0 and other node with code 1.
iv. Calculate the sum of probabilities of these two nodes combined in step ii.
v. Add another node to the tree taking into account the steps ii, iii and iv.

vi. Now after the tree is complete by combining all the nodes then compute the Entropy, Average Length and Redundancy.
vii. After making the Huffman Tree we get a code word for each symbol. Now concatenate these code words to get a string.
viii. The resultant string in step vii. is in the form of 1 and 0 so now we find the probability of 0 and 1.
ix. Repeat Step form i to vi.

A. Entropy

Entropy is the average value of information contained in the message. Entropy of a random variable [19] Y can be written as:

$$H(Y) = -\sum_{i=1}^{n} P(y_i)\, log_b\, P(y_i)$$

where b is the base of the logarithm used.

B. Average Length

If we multiply the probability with its code word [18] length then we can get the average length of Huffman coding. Average length of code is defined as:

$$Avg.\ Length = \sum_{i=1}^{n} P_i * L_i$$

C. Redundancy

In information theory [14], redundancy [13] means the number of bits in the transmitted message minus number of bits in actual message.

$$Redundancy = Entropy - Avg.\ Length$$

III. Construction of Huffman Coding And Double Huffman Coding

We will illustrate this section with the help of an example. Suppose we have six different symbols having probabilities as follows:

TABLE I. Symbols and their Frequency of Occurrence

Symbol	Probability
A	1/6
B	1/6
C	1/6
D	2/6
E	1/6

Now we apply the algorithm discussed in section and get the following tree.

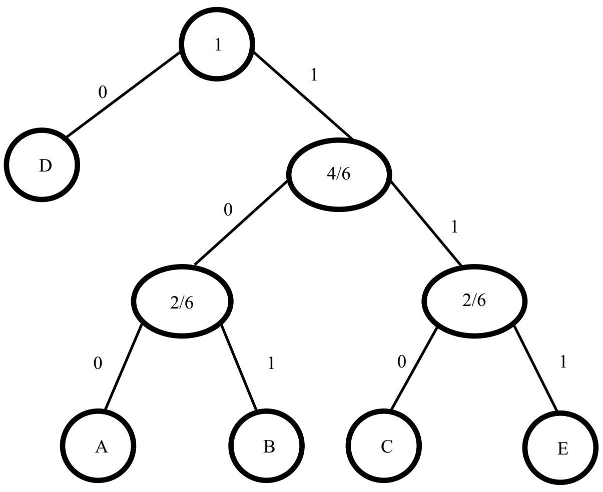

Fig 1 Huffman Tree

After applying the algorithm on above data we get the following code words for each symbol.

Table2: Code Word for Each Symbol by applying Huffman Coding

TABLE II. Code Word for Each Symbol by Applying Huffman Coding

Symbol	Code Word
A	100
B	101
C	110
D	0
E	111

Now we calculate Entropy, Average Length and Redundancy. So we get the following data.

TABLE III. Entropy, Length and Redundancy by Applying Huffman Coding

Entropy	2.2516 bits/sec
Average Length	2.3333 bits/sec
Redundancy	0.0817 bits/sec

Now, after concatenating the symbol with the code word, the string is "1001011100111". This string contains data only in the form of 0 and 1. So, probability for these symbols is given in TABLE IV.

TABLE IV. Binary Symbols and their Frequency of Occurrence

Symbol	Probability
0	5/13
1	8/13

Now we will repeat the procedure from step 1- 4 and get the tree in the figure.

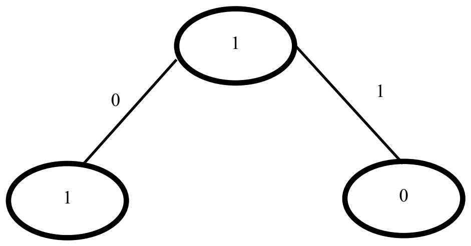

Fig 2. Binary Huffman Tree

Now we calculate Entropy, Average Length and Redundancy for the tree in figure. Calculated values are given in TABLE V.

TABLE V. Entropy, Length and Redundancy by Applying Huffman Coding

Entropy	0.9612 bits/sec
Average Length	1 bits/sec
Redundancy	0.0387 bits/sec

Therefore after applying Huffman Algorithm and Double Huffman Algorithm we get the information given in TABLE VI and TABLE VII respectively.

TABLE VI. Results of Huffman Coding

Symbols	A	B	C	D	E
Probabilities	1/6	1/6	1/6	2/6	1/6
Code Word	100	101	110	0	111
Code Word Length	3	3	3	1	3
Entropy	2.2516 bits/sec				
Average Length	2.3333 bits/sec				
Redundancy	0.0817 bits/sec				

TABLE VII. Results of Double Huffman Coding

Symbols	1	0
Probabilities	8/13	5/13
Code Word	0	1
Code Word Length	1	1
Entropy	0.9612 bits/sec	
Avg. Length	1 bits/sec	
Redundancy	0.0387 bits/sec	

It can be seen from Table and Table that redundancy in Double Huffman Coding is very less as compared to Huffman Coding. Figure represents a comparison between Huffman and Double Huffman Coding on basis of redundancy.

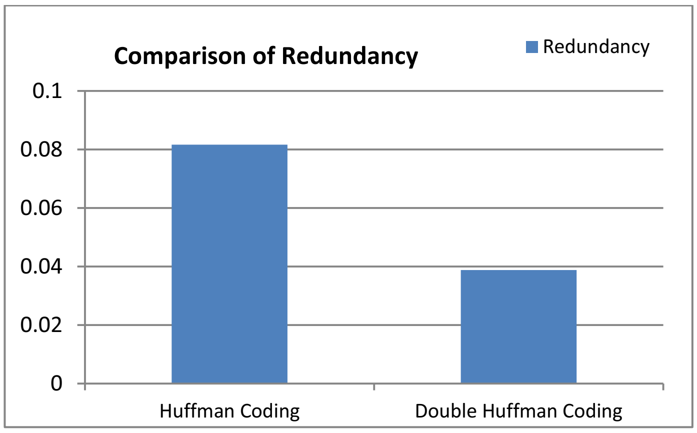

Fig 3. Comparison of Redundancy in Huffman Coding and Double Huffman Coding

IV. Conclusions

In this paper we discussed two important compression techniques i.e. Huffman Coding and Double Huffman Coding. Huffman coding is widely used in field of data compression but for a better performance another algorithm named Double Huffman coding was introduced. We conclude that Double Huffman Coding performs better than Huffman Coding as redundancy in Double Huffman Coding is less than redundancy in Huffman Coding. So Double Huffman Coding is more efficient method of compression than Huffman Coding.

V. References

[1] http://ee.lamar.edu/gleb/dip/index.html

[2] Mamta Sharma, “Compression Using Huffman Coding”, IJCSNS International Journal of Computer Science and Network Security, VOL.10 No.5, May 2010

[3] M Nelson and J-L Gaily, “*The Data Compression Book*”, 2nd edn., MIS Press, 1995.

[4] V Bhaskaran and K Konstantinides. *“Image and Video Compression Standards: Algorithms Architectures”*. Kluwer Academic Publishers, 1997.

[5] R M Capocelli, R Giancarlo and I J Taneja, “*Bounds on the redundancy of Huffman codes*”, IEEE Transactions on Information Theory, IT-32, pp 854-857, 1986.

[6] A Turpin and A Moffat, “*Housekeeping for Prefix Coding*”, IEEE Transactions on Communications, 48, pp.622- 628, 2000.

[7] Aarti, “Performance Analysis of Huffman Coding Algorithm” International Journal of Advanced Research in Computer Science and Software Engineering, Volume 3, Issue 5, May 2013

[8] Pu, I.M., 2006, Fundamental Data Compression, Elsevier, Britain.

[9] Bentley, J.L., Sleator, D.D., Tarjan, R.E., and Wei, V.K. “A Locally Adaptive Data Compression Scheme.” CACM 29, 4 (April 1986), pp. 320-330.

[10] A. Moffat and A. Turpin, “On the implementation of minimum redundancy prefix codes,” *IEEE Trans. Commun.*, vol. 45, no. 10, pp. 1200–1207, Oct. 1997.

[11] T.C. Bell, J.G. Cleary, and I.H. Witten, Text Compression,1990: Prentice-Hall, Englewood Cliffs, NJ.

[12] D. A. Lelewer and D.S. Hirschberg, “Data Compression”, ACM Computing Survery (CSUR), 1987. 19(3): p. 261-296.

[13] Derek Partridge; “Information Theory and Redundancy”, *Philosophy of Science* Vol. 48, No. 2 (Jun., 1981), pp. 308-316

[14] Chaoli Wang, Han-Wei Shen, “Information Theory in Scientific Visualization” , Entropy 2011, 13, 254-273; doi:10.3390/e13010254

[15] Nidhi Dhawale, “Implementation of Huffman algorithm and study for optimization”, IEEE Trans. Advances in Communication and Computing Technologies (ICACACT), 2014 International Conference, Aug. 2014

[16] Asadollah Shahbahrami, Ramin Bahrampour, Mobin Sabbaghi Rostami, Mostafa Ayoubi Mobarhan, “Evaluation of Huffman and Arithmetic Algorithms for Multimedia Compression Standards”

[17] Rupinder Singh, Brar Bikramjeet singh, “A Survey on Different Compression Techniques and Bit Reduction Algorithm for Compression of Text/Lossless Data”, International Journal of Advanced Research in Computer Science and Software Engineering, Volume 3, Issue 3, March 2013.

[18] Parikshit Gopalan, Cheng Huang, Huseyin Simitci, Sergey Yekhanin, “On the Locality of Codeword Symbols”

[19] Debasis Kundu, Rameshwar D. Gupta, Anubhav Manglick, “A Convenient Way of Generating Normal Random Variables Using Generalized Exponential Distribution”.

[20] Rajeshree Naire, Ben Soh, “A Modified Ternary Tree for Adaptive Huffman Encoding Data Structure”, JOURNAL OF RESEARCH AND INNOVATION IN INFORMATION SYSTEMS.

[21]]X. Kavousianos, E. Kalligeros, D.Nikolos, “Optimal Selective Huffman Coding for Test-Data Compression,” IEEE Trans.Computers, vol. 56, no. 8, pp. 1146-1152, Aug. 2007.