The International Journal Of Engineering And Science (IJES)
|| Volume || 3 || Issue || 6 || Pages || 46-54 || 2014 ||
ISSN (e): 2319 – 1813 ISSN (p): 2319 – 1805

Analysing Forward Difference Scheme on Huffman to Encode and Decode Data Losslessly

[1,]Adamu M. Garba and [2,]P. B. Zirra
[1,2,]Computer Science Department, Adamawa State University, Mubi

ABSTRACT
Data can be characters in text file, numbers that are samples of speech or image waveforms, or sequences of numbers that are generated by other processes,[1]. This type of data can be saved or transmitted over the Internet from one point to another through a communication channel. Data usually sent is in the form of binary digits (bits) called packets. Communication channel, however, has some overheads, which includes speed of transmission and bandwidth. For this reason, there is need to reduce the size of data to be transmitted through the reduction of redundancies in the data to be transmitted, that is, compression. This paper proposed a new algorithm, which encodes data as well as decodes or regenerates the replica of the encoded data. Using Forward Difference Scheme on Huffman, the character positions(C) can be transformed to sets of delta differences (D). The D values are later regenerated into Fixed Length Code (FLC) values using the twos complement. The FLC values are going to be used to further compute the new probabilities (P_{new}), which could be used to encode data using Huffman's algorithm.

***KEYWORDS**: Data Compression, Forward Difference, Lossy , Lossless, Huffman*

Date of Submission: 19 June 2014 — Date of Publication: 30 June 2014

I. INTRODUCTION

Compression, according to Oxford Advanced Learners Dictionary, is pressing together, making smaller using pressure. But in computer science, it refers to reduction of a file containing data using a special algorithm. File sizes can be reduced to various percentages of their original sizes. This is called data compression. The compression techniques are of two types: Lossless and Lossy. According to [16], computers process miscellaneous data. Some data, such as pictures, voices or videos, are analogue. Present-day computers do not work with infinite-precise analogue values, so the data has to be converted to a digital form. Digitization, according to [3], is a process by which text, images, audio and video are converted to the digital form. Once the data is in the digital form, it can be converted to another digital form without loss of quality; unlike the analog types which degrades with each use and losses quality when copied,[21].That is, the infinite number of values is reduced to a finite number of quantized values. Therefore some information is always lost (i.e. lossy type).Regardless of the way data are gathered to computers, they usually are sequences of elements. The elements come from a finite ordered set, called an alphabet. The elements of the alphabet, representing all possible values, are called symbols or characters. One of the properties of a given alphabet is its number of symbols, which is called the size of the alphabet. For a Boolean sequence the alphabet consists of only two symbols: false and true, represented as 0 or 1 bit only, [13].[16]further added that a sequence of symbols can be stored in a file or transmitted over a network in a compressed form, since the sizes of modern databases, application files, or multimedia files can be extremely large. All data compression methods rely on a priori assumptions about the structure of the source data. Huffman coding, for example, assumes that the source data consists of a stream of characters that are independently and randomly selected according to some static probabilities,[4].The ultimate goal of compression is to reduce data size, which saves computing resources or reduces the transmission time. Sometimes it becomes even impossible to store the sequences without compression. Various techniques have been used by different researchers, which resulted into different algorithms. The technique in this research work will involve the use of Forward Difference Scheme to produce the first codes representation whose new probability will be recomputed and compressed again using Huffman's algorithm.

II. STATEMENT OF THE PROBLEM

A compression problem, according to [6], involves finding an efficient algorithm to remove various redundancies from a certain type of data and to take less storage space. He further suggested that the *solutions* to the compression problems would then be the compression algorithms that will derive distinct sequence of symbols which contains *less number* of bits in total, plus the decompression algorithms to recover the original string. As there is no 'one size fits all' solution for data compression problems, this gives rise to a variety of data model [10] and representation techniques, which are at the heart of compression techniques.[12]concluded that there is no such thing as a "universal" compression algorithm that is guaranteed to compress any input, or even any input above a certain size. This research work intends to come up with anew algorithm that optimizes data compression using Forward Difference Scheme on Huffman's algorithm to compress and decompress text messages.

III. AIM AND OBJECTIVES OF THE STUDY

The aim of this work is to compress and decompress data without loss of information.
The objectives are as follows:

[1] To provide algorithm for coding and decoding using Forward Difference on Huffman Coding Technique
[2] To develop a software framework for the implementation the proposed algorithm
[3] To measure the performance of Huffman algorithm against the Forward Difference on Huffman using Compression Ratio, Compression Factor and Saving Percentage of the algorithms.

IV. SIGNIFICANCE OF THE STUDY

Data compression is an important field in Information Theory. The two fundamental significance of compression is saving storage space and maximize speed of transmission of messages. This work will be relevant to all that would like to minimize storage space on the computers and have a faster transmission of files on a network. This study will be of great importance to organizations or institutions that deal with large volume of data for storage and transmission.

V. SCOPE OF THE STUDY

This research will cover Forward Difference (FD) Scheme and Huffman's technique. It will examine the application of FD on Huffman to produce new coding algorithm and to improve the efficiency with which data are stored and transmitted over a network.

VI. LITERATURE REVIEW

HISTORY OF DATA COMPRESSION

[18]Claimed that Morse code, invented in 1838 for use in telegraphy, is an early example of data compression. He further added that modern work on data lossless 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. An optimal method for doing this was then found by David Huffman in 1951. Early implementations were typically done in hardware, with specific choices of code-words being made as compromises between compression and error correction. [20] related that lossy compression begins in the 1960s where an analogue videophone system had been tried out in the 1960s, but it required a wide bandwidth and the postcard-size black-and-white pictures produced did not add appreciably to voice communication. In the 1970s, it was realized that visual speaker identification could substantially improve a multiparty discussion and videoconference services were considered. Interest increased with improvements in picture quality and digital coding.

[15]added that in the early 1990s, the Motion Picture Experts Group (MPEG) started investigating coding techniques for storage of video, such as Compact Disc. The aim was to develop a video codec capable of compressing highly active video such as movies, on hard disks. MPEG-1 standard, was capable of accomplishing this task at 1.5 Mbit/s. Since for the storage of video, encoding and decoding delays are not a major constraint, one can trade delay for compression efficiency. Broadcasters soon adopted a new generation of MPEG, called MPEG-2. A slightly improved version of MPEG-2, called MPEG-3, was to be used for coding of High Definition (HD) TV, but since MPEG-2 could itself achieve this, MPEG-3 standards were folded into MPEG-2. [20]stated that it was foreseen by 2014, the existing transmission of National Television System Committee (NTSC)format in North America will cease and instead HDTV with MPEG-2 compression will be used in terrestrial broadcasting.

Early work on lossless data compression, according to [13], began in the late 1940s with the development of Information Theory. Claude Shannon and Robert Fano, in 1949, devised a methodical way to assign codewords based on probabilities of blocks. An optimal method for doing this was then found by David Huffman in 1951. Early implementations were typically done in hardware, with specific choices of codewords being made as compromises between compression and error correction. In the mid-1970s, the idea emerged of dynamically updating codewords for Huffman encoding, based on the actual data encountered. And in the late

1970s, with online storage of text files becoming common, software compression programs began to be developed, almost all based on adaptive Huffman coding,[18].
Until 1980, [13]said, most general-compression schemes used statistical mode[10]. Later, in 1977, Abraham Lempel and Jacob Ziv published their groundbreaking LZ77 algorithm, the first algorithm to use a dictionary base to compress data. LZ77 used a dynamic dictionary oftentimes called a sliding window. The drawback to LZ77 is that it has a small window size.Around the middle 80s, subsequent work by Terry Welch, the so-called LZW algorithm rapidly became the method of choice for most general-purpose compression systems. It was used in programs such as PKZIP, as well as in hardware devices such as modems and UNIX machines. In 1988, a man named Phil Katz came up with PKARC after he has studied ARC's popularity and decided to improve it by writing the compression and decompression routines in assembly language. The format was again updated in 1993, when Katz released PKZIP 2.0.

VII. DATA COMPRESSION

[1]are of the view that data compression is the art or science of representing information in a compact form. Identifying and using structures that exist in the data create this compact representation. [22]defined data compression as the reduction of the volume of a data file without loss of information or message. [15]added that data compression is the art or science of representing information in a compact form.Data, when is compressed, is said to be encoded data or compacted data.
[15]associated two standard metrics often employed in data compression, which are efficiency and effectiveness. Efficiency is the measure of a resource requirement. Generally, it is the speed or throughput of an algorithm. It can be measured in CPU time (sec), symbols per second (sym/sec), or another similar hybrid measure. Effectiveness is the amount of redundancy removed. It is commonly expressed as a compression ratio (%), or in bits per symbol (bps).

DATA ENCODING

Encoding is a mapping of source messages (words from the source alphabet α) into codewords (words of the code alphabet β). The source messages are the basic units into which the string to be represented is partitioned. These basic units may be single symbols from the source alphabet, or they may be strings of symbols. For example,αcould be a set (a, b, c, d, e, f, g, space) and βa binary set (0, 1). The binary representation of the distinct characters is called codeword,[6].
Encoding [15], means the assignment of binary sequences to elements of an alphabet. The set of binary sequences is called a code, and the individual members of the set are called codewords. An alphabet is a collection of symbols called letters.
[18]stated that the code used by most computers for text files is known as ASCII (American Standard Code for Information Interchange). ASCII can depict uppercase and lowercase alphabetic characters, numerals, punctuation marks, and common symbols. Other commonly used codes include Unicode etc.

DATA DECODING

Decoding is the opposite process of encoding, that is, the conversion of an encoded format back into the original sequence of characters. Encoding and decoding are used in data communications, networking and storage [11].

APPLICATION OF DATA COMPRESSION

Data compression has a wide range of applications as stated by [7], especially in data storage and data transmission. Some of these applications include its use in many archiving systems such as ARC and, telecommunications such as voice mail and teleconferencing. Audio, video, graphical, streaming media, Internet telephony applications and textual information can all benefit from data compression [17].[15]held that Data Compression is the enabling [10] technology behind Information and Telecommunication, and multimedia revolution. He added that without Data Compression, it would not have been possible to:
[1] Put images, audio and video onto a web page.
[2] Provide communication with clarity on cellular phones
[3] Have digital TV
[4] Make a long distance call
[5] Listen to music on mp3 player or watch video on a DVD player.

CURRENT COMPRESSION SOFTWARE

[14]released his archiver known as WinRAR. The latest version of Roshal ARchiver (RAR) uses a combination of the Prediction by Partial Matching (PPM) and Lempel-Ziv-Storer-Szymanski (LZSS) algorithms, but not much is known about earlier implementations. In 1999, he again released another open-source compression

program was released which was known as the 7-Zip or .7z format. 7-Zip could be the first format to challenge the dominance of ZIP and RAR due to its generally high compression ratio and the format's modularity and openness. This format is not limited to using one compression algorithm, but can instead choose between bzip2, LZMA, LZMA2, and PPM algorithms among others. Finally, on the bleeding edge of archival software are the PAQ formats. The first PAQ format, [12] said was released by Matt Mahoney in 2002, called PAQ1. PAQ substantially improves on the Prediction by Partial Matching (PPM) algorithm by using a technique known as context mixing which combines two or more statistical models to generate a better prediction of the next symbol than either of the models on their own.

TYPES OF COMPRESSION TECHNIQUES

Two types of data Compression techniques were identified by [6], namely:

[1] Lossless Compression: [15]asserts that as their name implies, involve no loss of information. Lossless compression, he added, is generally used for applications that cannot tolerate any difference between the original and reconstructed data, and a good example is in the text compression.

[2] Lossy Compression: [6] stated that compression is lossy if it is not possible to reconstruct the original message exactly from the compressed version. He added that Lossy compression is called irreversible compression since it is impossible to recover the original data exactly by decompression. This means that there are some insignificant details that may get lost during the process of compression. The word insignificant here [6] said implies certain requirements to the quality of the reconstructed data, for instance, some certain qualities of an image might be chopped off but that does not affect the image after reconstruction, as the human eye does not mostly notice this. Examples of lossy file formats are MP3, AAC, MPEG and JPEG.

ENTROPY OF A MESSAGE

[13]relates that the term entropy is well used in Information Theory as a measure of how much information is encoded in a message. The word entropy [6] said, was borrowed from thermodynamics, meaning measure of the level of disorder in a system or amount of unavailable energy in a system. In our case the system here is the message. The higher the entropy of a message, the more information it contains. *The entropy of a symbol* is defined as the negative logarithm of its probability. To determine the information content of a message in bits, mathematically, the number of bits is given by negative logarithm base 2 of the probability, that is: $Number\ of\ bits = -\log_2(Probability)$ (1)

The entropy of an entire message is simply the sum of the entropy of all individual symbols. Entropy fits with data compression in its determination of how many bits of information are actually present in a message. If the probability of the character ‘b’ appearing in this manuscript is 1/16, for example, the information content of the character is four bits. So the character string “bbbbb” has a total content of 20 bits. If the standard 8-bit ASCII characters were to be used to encode this message, 40 bits will be used against 20. The difference between the 20 bits of entropy and the 40 bits used to encode the message is where the potential for data compression arises. But [8]added that this method could be used only if the compression algorithm is based on statistical information of the source file.

DATA COMPRESSION ALGORITHMS

Different algorithms have been used for compressing data. Most compression algorithms cannot stand alone; but must be combined together to form a compression algorithm. Those that can stand-alone are often more effective when joined together with other compression techniques. Most of these techniques fall under the category of entropy encoders (such as Huffman and Arithmetic), but there are others such as Run-Length Encoding and the Burrows-Wheeler Transform that are also commonly used. The following lossless are discussed as illustrations on how they work.

Run-Length Encoding

Run-Length Encoding, as described by [6], is a very simple lossless compression algorithm that replaces the consecutive recurrent symbols in a message, with record of the length of each run and the symbol in the run. Run-length algorithms are very effective if the source contains many runs of consecutive symbols. For instance, given an input string below PPPPPPPPP, the output will be: 9P and an input of ABABBBC gives the output: ABA3BC

Shannon-Fano Algorithm

This is one of the earliest lossless compression techniques, invented in 1949 by Claude Shannon and Robert Fano. In the Shannon-Fano approach, [6]said that a binary tree is constructed in a 'top-down' manner.

This technique, according to [13],is to build a Shannon-Fano tree according to a specification designed to define an effective code table. The actual algorithm is simple:
The algorithm to generate Shannon-Fano codes said [13] is fairly simple
[1] Parse the input, counting the occurrence of each symbol.
[2] Determine the probability of each symbol using the symbol count.
[3] Sort the symbols by probability, with the most probable first.
[4] 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 and 1 to the left and right nodes' codes, respectively.
[7] Recursively apply steps 5 and 6 to the left and right subtrees until each node is a leaf in the tree.

Huffman Coding

Huffman Coding as lossless coding technique is another variant of entropy coding that works in a very similar manner to Shannon-Fano Coding, but the binary tree is built in a “top-down” manner in order to generate an optimal result, [5].The algorithm to generate Huffman codes:
[1] Parse the input, counting the occurrence of each symbol.
[2] Determine the probability of each symbol using the symbol count.
[3] Sort the symbols by probability, with the most probable first.
[4] Generate leaf nodes for each symbol, including P, and add them to a queue.
[5] While (Nodes in Queue is greater than 1)
[6] Remove the two lowest probability nodes from the queue.
[7] Prepend 0 and 1 to the left and right nodes' codes, respectively.
[8] Create a new node with value equal to the sum of the nodes’ probability.
[9] Assign the first node to the left branch and the second node to the right branch.
[10] Add the node to the queue
[11] The last node remaining in the queue is the root of the Huffman tree.

Arithmetic Coding

Arithmetic coding, which is a member of the lossless coding family, transforms the input data into a single fractional number between 0 and 1 by changing the base and assigning a single value to each unique symbol from 0 up to the base, [6]. Arithmetic coding, said [18], overcomes the problem of assigning integer codes to the individual symbols by assigning one code to the entire input file. The method starts with a certain interval, it reads the input file symbol by symbol, and employs the probability of each symbol to narrow the interval. The output of arithmetic coding is interpreted as a number in the range [0, 1).
The main problem of Arithmetic Coding is that it is quite complicated compared to the other coding techniques.
[6]states a general algorithm to compute the arithmetic code as follows:
[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, replace the symbols in the input with their codes Convert the result from step from base b to a sufficiently long fixed-point binary number to preserve precision.
[4] Record the length of the input string somewhere in the result as it is needed for decoding

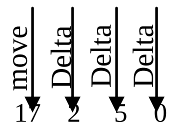
FORWARD DIFFERENCE

The forward difference is a finite difference scheme defined by:

$$\Delta a_n \equiv a_{n+1} - a_n \tag{2}$$

Where n = 0, 1, 2…
[19]said that in science, engineering, and mathematics, the Greek letter *delta* (Δ) is used to denote the *change* in a variable. The term delta encoding refers to several techniques that store data as the difference between successive samples (as residuals), rather than directly storing the samples themselves (as source symbols). The following is an example of Delta Encoding.

Original data stream: 17 19 24 24 24 21 15 10 89 95 96 96 95 96 93 90 87 86 … …



Delta encoded: 17 2 5 0 0 -3 -6 -5 79 6 1 0 -1 1 -3 -3 -3 -1 …

Delta encoding can be used for data compression when the values in the original data are *smooth*, that is, there is typically only a small change between adjacent values. [19]further explained that delta encoding has increased the probability that each sample's value will be near zero, and decreased the probability that it will be far from zero. This uneven probability is just the thing that Huffman encoding needs to operate.

Algorithm's Performance Evaluation

Following are some measurements used to evaluate the performances of compression algorithms[8].

- Compression Ratio: is the ratio between the size of the compressed file and the size of the source file.

$$CompressionRatio = \frac{Size\ after\ compression}{Size\ before\ compression} \tag{3}$$

- Compression Factor: is the inverse of the compression ratio. That is the ratio between the size of the source file and the size of the compressed file.

$$Compression\ Factor = \frac{Size\ before\ compression}{Size\ after\ compression} \tag{4}$$

- Saving Percentage calculates the shrinkage of the source file as a percentage.

$$= \frac{Sizebeforecompression - Sizeaftercompression}{Sizebeforecompression} X100\% \tag{5}$$

UNIQUE DECODABILITY

The ability to decode a coded symbol is referred to as the measure of decodability. This is computed using the Kraft's Inequality Criterion, which states that:

Let i be the number of symbols and N be the length of the corresponding code, i.e., N_i is the length of i^{th} code, then k is such that

$$K = \sum_{i=0}^{m} 2^{N_i} \quad , \tag{6}$$

Where *m* is the number of individual characters. However when k < 1, the code is said to be uniquely decodable, [5]&[15].

ADVANTAGES OF DATA COMPRESSION

The advantage of data compression as viewed by [9] is that it shrinks down a file so that it takes up less space; this invariably increases space for data storage and speeds up data transmission. Storage space on disks is expensive and as applications becomes more complex and data sizes to be transmitted keep on increasing; there is need for data file economy. However, a file that occupies less disk space (compressed file) is cheaper to maintain than one that occupies larger space (uncompressed file). Smaller files are also more desirable for data communication, because the smaller a file the faster it can be transferred. [22]observed that a compressed file appears to increase the speed of data transfer over an uncompressed file, and maintained that Compressed data reduces storage and communication costs. When the amount of data to be transmitted is reduced, the effect is that of increasing the capacity of the communication channel. They said that compressing a file to half of its original size is equivalent to doubling the capacity of the storage medium[10]. Similarly, [2]states that compressed data is useful because it helps reduce the consumption of expensive resources, such as storage device or bandwidth.

V. METHODOLOGY

The performance of the Data Compression using Forward Difference Technique on Huffman was measured using Compression Ratio, Compression Factor and Saving Percentage.

PROPOSED ALGORITHM

The proposed algorithm consists of the following:

Encoding Procedure

[1] Read the input file
[2] Serialize each character
[3] Determine the probability of each character
[4] Determine the positions of each character
[5] Compute the Forward Difference of the character positions using (4)
[6] Transform the difference to twos complement code
[7] Compute the new probability using the twos complement code using (3)
[8] Build a Huffman's Binary Tree using the new probability and the distinct characters.
[9] Determine the codeword for each distinct character by traversing the tree from root node to leaf node.
[10] Determine the binary symbols to be transmitted for the whole message.

Decoding Procedure

[1] Read the whole coded message bit-by-bit
[2] Determine a codeword from the coded message
[3] Traverse a Huffman Tree from the root to the leaf node
[4] Determine its symbol the code represents and its new probability
[5] Use the new probability to reconstruct the twos complement code
[6] Convert the twos complement code to its decimal equivalents
[7] Determine which of the decimal numbers describes a delta difference
[8] Use the delta differences and Backward Summation to regenerate the positions of each character
[9] Use the character positions to allocate each character its positions
[10] Write the whole message to a file.

VI. RESULTS AND DISCUSSION

Table 1: Compression using Huffman algorithm

FileNames	Original File Size (bytes)	Compressed File Size (bytes)	Compression Ratio	Compression Factor	Saving Percentage
File1	1024	631	61.62	1.62	38.38
File2	2048	1205	58.84	1.70	41.16
File3	4096	2567	62.67	1.60	37.33
File4	8192	4905	59.88	1.67	40.12
File5	11954	7215	60.36	1.66	39.64
File6	16384	10321	62.99	1.59	37.01
File7	32768	20433	62.36	1.60	37.64
File8	38105	23908	62.74	1.59	37.26
File9	46526	26337	56.61	1.77	43.39
File10	53161	31556	59.36	1.68	40.64
File11	65536	35879	54.75	1.83	45.25
File12	82199	44567	54.22	1.84	45.78
File13	93695	52373	55.90	1.79	44.10
File14	111261	55847	50.19	1.99	49.81
File15	131072	71234	54.35	1.84	45.65

Table 2: Compression using Forward Difference on Huffman (FDHuffman)

FileNames	Original File Size (bytes)	Compressed File Size (bytes)	Compression Ratio	Compression Factor	Saving Percentage
File1	1024	631	61.62	1.62	38.38
File2	2048	1205	58.84	1.70	41.16
File3	4096	2567	62.67	1.60	37.33
File4	8192	4905	59.88	1.67	40.12
File5	11954	7215	60.36	1.66	39.64
File6	16384	10321	62.99	1.59	37.01
File7	32768	20433	62.36	1.60	37.64
File8	38105	23908	62.74	1.59	37.26
File9	46526	26337	56.61	1.77	43.39
File10	53161	31556	59.36	1.68	40.64
File11	65536	35879	54.75	1.83	45.25
File12	82199	44567	54.22	1.84	45.78
File13	93695	52373	55.90	1.79	44.10
File14	111261	55847	50.19	1.99	49.81
File15	131072	71234	54.35	1.84	45.65

Compression Factor and Saving Percentage for Huffman and FDHuffman from Table 4.1 and Table 4.2 are pictorially represented below.

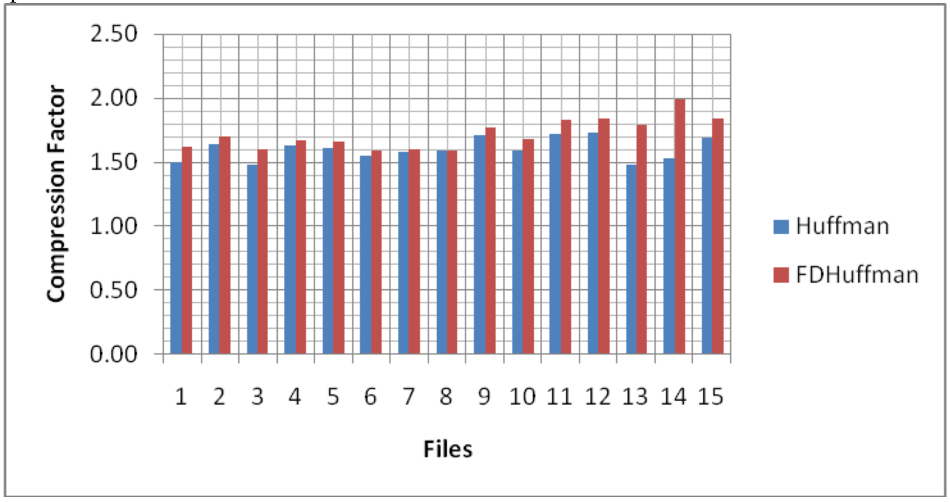


Figure 1: Compression Factors for FDHuffman and Huffman

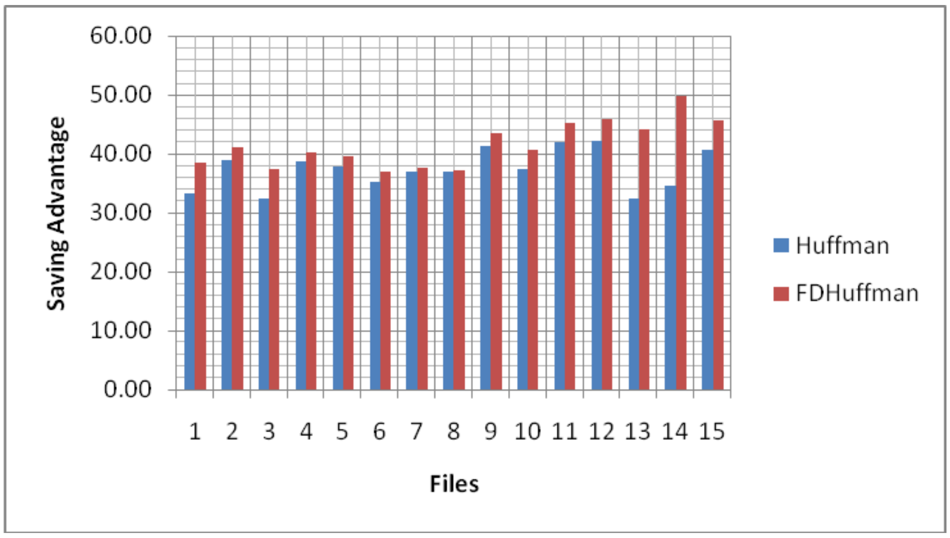


Figure 2: Saving Percentages for FDHuffman and Huffman

Figure 1 showed the compression factors for files File1, File2 … File15. That showed that the system achieved good compression as the compression factor for all the files are greater than 1, Salomon (2007). FDHuffman has higher compression factor than Huffman alone. Kodituwakku and Amarasinghe, (2012) affirmed in their work that high compression factor is a requirement for good compression.

Figure 2 showed the Saving Percentage (that is shrinkage of the source file as a percentage) for both Huffman and FDHuffman algorithms. It could be deduced from the figure that all files shrink faster in FDHuffman than in Huffman.

REFERENCES

[1] Adeniyi, M. &Oluwade, D. (2006), An Enhanced LZW text Compression Algorithm.pdf. Retrieved October 27, 2012 from http://www.ajocict.net/uploads/

[2] Ankush, K., Vikas, G., Sukant, V., &Sakshi, A. (2012), Implementing Watermarking Concept along with Data Compression in Multimedia Transmission, *International Journal of Electronics and Computer Science Engineering*. Retrieved December 5, 2012 from http://www.ijecse.org/wp-content/uploads/2012/08/Volume-1Number-2PP-527-532.pdf

[3] Choure, T. (2004), *Information Technology Industry in India*. Retrieved November 22, 2012 from http://books.google.com.ng/books

[4] Cormack, G.V. &Horspool, R. N. (1986), Data Compression Using Dynamic Markov Modeling. *The Computer Journal*, 30(6), 541-550.

[5] Guy, B. (2010), Introduction to Data Compression. Retrieved October 3, 2012 from http://www.cs.cmu.edu/afs/cs/project/pscico-guyb/realworld/www/compression.pdf

[6] Ida, M.P. (2006), *Fundamental Data Compression*. Elsevier Publishers: New York. Retrieved April 23, 2013 from http://www.dbebooks.com

[7] Kattan, A. J. (2006), Universal Lossless Compression Technique with built in Encryption. Retrieved October 24, 2012 from www.ahmedkattan.com/Master_Dissertation.pdf

[8] Kodituwakku, S.R. &Amarasinghe, U.S. (2000). Comparison of Lossless Data Compression Algorithms for text data. Retrieved September 16, 2012 from http://www.mattmahoney.net/dc/dce.htm

[9] Ladino P. (1996), Data Compression Reference. Retrieved April 23, 2013 from http://www.cs.toronto.edu/~mackay/itprnn/ps/65.86.pdf

[10] Ling, S. (2004), *Coding Theory-A First Course*, Cambridge University Press: New York.

[11] Margaret, R. (2005), Encoding and Decoding. Retrieved January 23, 2013 from http://www.searchnetworking.techtarget.com/definition/encoding-and-decoding

[12] Matt, M. (2012), Data Compression Explained. Retrieved October 19, 2012 from http://mattmahoney.net/dc/dce.html.

[13] Nelson, M. &Gailly, J. (1996). *The Data Compression* Book 2nd Ed. Penguin Publishers: New York.

[14] Roshal, E. (1999), RAR Compression. Retrieved November 1, 2012 from www.winrar.com

[15] Sayood, K. (2006), Introduction to Data Compression . Retrieved April 23, 2013 from http://www.csd.uoc.gr/~hy438/lectures/Sayood-DataCompression.pdf

[16] Sebastian, D. (2003), Universal lossless data compression algorithms. Retrieved August 31, 2012 from http://www.f5.com/pdf/white-papers/adv-compression-wp.pdf

[17] Shane, J.C. (2007), Efficient Data Representations for Information Retrieval. Retrieved January 27, 2013 from http://www.cs.rmit.edu.au/publications/sc07-thesis_final.pdf

[18] Wolfram, S. (2002), History of Data Compression. Retrieved December 6, 2012 from http://www.wolframscience.com/compression.html

[19] Steven, S.W (2011) The Scientist and Engineer's Guide to Digital Signal Processing. Retrieved April 23, 2013 from www.dspguide.com/ch27/4.htm

[20] Sundar, S. (2002), A Brief History of Compression. Retrieved October 19, 2012 from http://www.pha.jhu.edu/~sundar/intermediate/history.html

[21] Twari, P. (2007), *Information Technology and Library Evolution*. Retrieved November 11, 2012 from http://books.google.com.ng/books

[22] Zirra, P.B. &Wajiga, G.M. (2011) Radical Data Compression Algorithm Using Factorization, *International Journal of Computer Science and Security* (IJCSS), 5(2), 221-226: 2011. Retrieved October 24, 2012 from http://www.cscjournals.org/csc/manuscript/Journals/IJCSS/volume5/Issue2/IJCSS-454.pdf

AUTHORS:

Adamu Garba Mubi is a postgraduate student from Adamawa State University, Mubi Nigeria. He obtained hisB.Tech in Computer science from AbubakarTafawaBalewa University Bauchi, Nigeria in 2004. He is currently an M.Sc student at Adamawa University. His main area of research is Data Compression. He is a Registered Member of Nigeria Computer Society.

Dr. P. B. Zirra is a lecturer with Adamawa state University, Mubi Nigeria. He obtained his Doctorate Degree in Computer Science from Modibbo Adamawa University Yola in 2012, M.sc in Computer science from A.T.B.U Bauchi in 2006, MBA (Finance) from University of Maiduguri in 2000 and had his B.Tech. Computer 1994 from ATBU. His Area of interest include Computer Network and Security, he is happily married with two Children.