IJIRCCE

e-ISSN: 2320-9801 | p-ISSN: 2320-9798

INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN COMPUTER & COMMUNICATION ENGINEERING

Volume 10, Issue 7, July 2022

INTERNATIONAL STANDARD SERIAL NUMBER INDIA

Impact Factor: 8.165

9940 572 462 | 6381 907 438 | ijircce@gmail.com | www.ijircce.com

International Journal of Innovative Research in Computer and Communication Engineering

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 8.165 |

|| Volume 10, Issue 7, July 2022 ||

| DOI: 10.15680/IJIRCCE.2022.1007001|

Performance Evaluation of Lossless Data Compression Algorithms

Gideon Asante[1], Abdul-Barik Alhassan[2], Abdul-Mumin Salifu[3]

Department of Computer Science, University for Development Studies, Tamale, Ghana [1&2]

Department of Information Systems and Technology, C.K. Tedam University of Technology and Applied Sciences, Navrongo, Upper East Region-Ghana [3]

ABSTRACT:Our world today is increasing in demand for information storage and data transfer thereby increasing the significance of its compression (data compression). Data compression therefore is a method used to reduce the size of data on storage or transit. This is helpful and necessary when large file(s) are to be transmitted on a network or stored. There are mainly two categories of data compression (lossy and lossless), however in this paper, lossless method of data compression is considered. A review of different lossless compression algorithms are evaluated and tested on some files of different size and format. Run-Length Encoding, Huffman Algorithm and the Shannon-Fano Coding are used in this research. Their performance are evaluated based on size, ratio and speed of execution or compression time. The Huffman Algorithm performs better in terms of compression size while Run-Length Encoding performs better in terms of execution time.

KEYWORDS: Data Compression, Lossless, Huffman Coding, Run-Length Encoding, Shannon-Fano

I. INTRODUCTION

Data compression is one of the areas of interest in our world today as far as data storage and transmission is concern. It is the process by which a particular data is represented with lesser storage size by converting it to another form.Data compression has benefited our world for some decades. With the use of data compression techniques, much disk storage will be saved or transmission bandwidth, thus by reducing the consumption of resources [3, 11, 13].

Basically, two main types or categories of data compression exist; Lossy and Lossless datacompression. Lossy, as the name suggest, deals with removing some part of the data which takes much space to make the data small in size. In lossy data compression, the removed part or details cannot be retrieved (data cannot be restored to original state in Lossy Data Compression technique) [11, 12, 13].

Lossless data compression does not seek to remove part of data but seeks to reduce file size, such that, the same file can be restored or decompressed to the original state. The principle used by Lossless data compression is statistical modelling approach which check the probability of a phrase/character appearing. With the use of such technique, a string or characters of some size can be re-characterised with few, thus by removing a good sum of excess or extra characters [7].

There are different algorithms that are designed either with specific sort of data in mind or with assumption about what sorts of redundancy the uncompressed data are likely to contain. In this research, Run-Length Encoding (RLE) algorithm, Huffman algorithm and Shannon-Fano algorithm are examined based on their performance in terms of compression size, ratio and time.

II. RELATED WORK

The fundamental objective of data compression is to figure out and eliminate redundancy through different efficient methods; so that the compressed data can save space [10].To eliminate the redundancy, some code notation is used to represent the source file and this coded file is known as 'encrypted file'. With efficiency in mind, the compressed file must be less in size than the source file. Decompression technique is used to get back the original file [1, 6, 10].

Statistical Compression and Dictionary Compression Techniques are used on text data and in statistical compression techniques, arithmetic coding does better than other algorithms. Also, the Lempel-Ziv Banikazemi (LZB) performs best than other Lempel-Ziv (LZ77) family [5, 8, 10].

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 8.165 |

|| Volume 10, Issue 7, July 2022 ||

| DOI: 10.15680/IJIRCCE.2022.1007001|

III. MATERIALS AND METHODS

The following materials and methods were used in order to evaluate the performance of the algorithms under consideration.

A. Run-Length Encoding (RLE)

Run-Length Encoding (RLE) is one of the simplest when it comes to lossless data compression. RLE is mostly used for data with symbols of repetitions. The length of the text or string is known as the Run. The main point behind RLE is to symbolise redundant characters as pairs[10, 11, 13].

The algorithm for the Run-Length Encoding (RLE) is depicted below;

Start

Count characters of the source file (**Sf**)

While (|**Sf**| > 0)

Count occurrences of characters

Append current character and its count to the result

(Example; Encoding+=string(count)+**Sf**(index))

Stop

For example, consider a string "AABBCCCADDDDDD" of repeated sequence, when RLE is performed on the string, notice that, the character "A" appears two times (2A), "B" appears two times (2B), "C" appears three times (3C), "A" appears again (1A), and lastly, "D" appears six times (6D). So, the original string of 14 bytes (characters) can now be represented as 2A2B3C1A6D thus 10 bytes (characters) after the use of RLE. However, the biggest problem that comes with Run-Length Encoding is that, sometimes the compressed text or string can be bigger than the original file. For example, a string "AABCDEFG" of size 8 bytes (characters), when RLE is performed on it, it results in; 2A, 1B, 1C, 1D, 1E, 1F and 1G. Therefore, the compressed file will be "2A1B1C1D1E1F1G" which is 14bytes (characters), bigger than the original file of size 8 bytes[10].

B. Huffman Coding Algorithm

In 1950, in an information theory class at Massachusetts Institute of Technology (MIT), a student named DavidHuffman, developed an algorithm and was named after the inventor as Huffman Coding algorithm. When it comes to string or text compression, Huffman algorithm is known to be more successful or gives better results [3, 12].

The main idea behind Huffman algorithm is to substitute fixed-length codes by variable-length codes, meaning it deals with ASCII characters in compression. In Huffman coding, files are turned to binary trees. The leaves of the binary tree are represented by the characters, in the binary tree, the successful construction of the tree is a means of determining the Huffman code. Huffman coding reduces the total number of bits used without information loss [2, 11].

The algorithm for the Huffman Coding is depicted below;

Start

Count each character in source file

Sort to non-decreasing order

Create leaf node (character, frequency **f**, left child, right child) of the tree for each character and then put nodes into queues **Q**

While (|**Q**| >= 2) do

Start

Pop first two nodes$(n1, n2)$ with lowest **f** from sorted **Q**

Create a node with sum of the chosen units, successors are chosen (esp, f(n1)+f(n2)) units

Insert new node into queue **Q**

Stop

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 8.165 |

|| Volume 10, Issue 7, July 2022 ||

| DOI: 10.15680/IJIRCCE.2022.1007001|

Note, evaluate from root to leaf node (left child **1**, right child **0**)

Output results

Stop

C. Shannon-Fano Algorithm

Shannon-Fano Coding is one of the lossless compression techniques that was discovered by Claude Shannon and Robert Fano in 1949. This technique is for constructing the prefix code base on symbols and probability. In other words, it can be said to be an algorithm used to compress strings [9, 11, 13].

Shannon-Fano coding is similar to Huffman coding, but differs in the way the binary tree of symbol nodes are built.

The algorithm for Shannon-Fano is depicted below;

Start

Count characters of source file (*Sf*)

Sort *Sf* to non-decreasing order

Ag (Split (*Sf*))

Print (count of symbols, encoded tree)

Stop

Procedure Ag (Split (*Sf*))

Start

If (|*Sf*| > 1) then

Start

Divide *Sf* to *Sf1* and *Sf2*

Add 0 to codes in Sf2

Add 1 to codes in *Sf1*

Ag (Split (*Sf1*))

Ag (Split (*Sf2*))

Stop

Stop

D. Performance Measurement

Compression algorithm performance mostly lies in the redundancy in the source file. The following are factors or metrics used to assess the performance of algorithms.

1. *Compression Size (CS),* is the size of the new file after compression is completed. In other words, the compressed file size.

2. *Compression Ratio (CR),* is the ratio of compressed file size and the original file. This is given by;

CR=$\frac{Compressed\ file\ size\ (Size\ after\ compression)}{Original\ file\ size\ (Size\ before\ compression)}$, in percentage wise, CR=$\frac{Compressed\ file\ size\ (Size\ after\ compression)}{Original\ file\ size\ (Size\ before\ compression)}$x 100.

3. *Compression Time (CT*), is the time taken for the compression to complete in milliseconds. The execution time in milliseconds is divided by the number of characters in the original file. The results will be the time taken to compress each character or bit in the original file. Compression Time can also be called, Speed of Execution

IV. RESULTS AND DISCUSSION

In this section, three lossless data compression algorithms are tested on ten (10) files of different types and sizes, as well as content. Among these files are seven (7) text files and the other three (3) are Visual Basic program files.

The tables below show the test performed on the files with the Run-Length Encoding (RLE), Huffman Coding, and the Shannon-Fano Coding.

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 8.165 |

|| Volume 10, Issue 7, July 2022 ||

| DOI: 10.15680/IJIRCCE.2022.1007001|

TABLE I
RUN-LENGTH ENCODING (RLE)

No.	File Name	File Size(bytes)	Compressed Size (bytes)	Compression Ratio (%)	Compression Time (ms)
1	Txt File 1	12883256	12127483	94.13368	530
2	Txt File 2	373874	371216	99.28906	15
3	Txt File 3	18757175	18704914	99.72138	1070
4	Txt File 4	611682	600720	98.20789	35
5	Txt File 5	28081207	28070493	99.96185	1615
6	Txt File 6	2689240	2637663	98.0821	156
7	Txt File 7	559	560	100.1789	1
8	Vb File 1	34627	28975	83.67747	1
9	Vb File 2	8993	7424	82.5531	1
10	Vb File 3	15940	12922	81.0665	1

TABLE II
HUFFMAN CODING ALGORITHM

No.	File Name	File Size(bytes)	Compressed Size (bytes)	Compression Ratio (%)	Compression Time (ms)
1	Txt File 1	12883256	6756616	52.44494	3135
2	Txt File 2	373874	222137	59.41494	100
3	Txt File 3	18757175	9514989	50.7272	4903
4	Txt File 4	611682	331355	54.17112	180
5	Txt File 5	28081207	14236923	50.69911	7296
6	Txt File 6	2689240	1473616	54.79675	688
7	Txt File 7	559	279	49.91055	5
8	Vb File 1	34627	20362	58.80383	16
9	Vb File 2	8993	5307	59.01257	5
10	Vb File 3	15940	9109	57.14555	8

TABLE III
SHANNON-FANO ALGORITHM

No.	File Name	File Size(bytes)	Compressed Size (bytes)	Compression Ratio (%)	Compression Time (ms)
1	Txt File 1	12883256	6794410	52.7383	3155
2	Txt File 2	373874	223534	59.78859	105
3	Txt File 3	18757175	9787621	52.18068	4741
4	Txt File 4	611682	335718	54.8844	165
5	Txt File 5	28081207	14666024	52.22718	7141
6	Txt File 6	2689240	1490946	55.44117	672
7	Txt File 7	559	287	51.34168	5
8	Vb File 1	34627	20601	59.49404	16
9	Vb File 2	8993	5416	60.22462	5
10	Vb File 3	15940	9232	57.91719	8

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 8.165 |

|| Volume 10, Issue 7, July 2022 ||

| DOI: 10.15680/IJIRCCE.2022.1007001|

A. *Results Analysis*

Three lossless data compression algorithms are tested on ten (10) different types of files (with different size and content). The outcome of the test is analysed comparatively based on three key metrics or factors including comparison based on compression size, ratio (CR), and time (CT).

TABLE IV

COMPARISON BASED ON COMPRESSED FILE SIZE

Original File Size			**Compressed File Size**		
No.	File Name	File Size (bytes)	Run-Length Encoding(bytes)	Huffman Coding(bytes)	Shannon-Fano (bytes)
1	Txt File 1	12883256	12127483	6756616	6794410
2	Txt File 2	373874	371216	222137	223534
3	Txt File 3	18757175	18704914	9514989	9787621
4	Txt File 4	611682	600720	331355	335718
5	Txt File 5	28081207	28070493	14236923	14666024
6	Txt File 6	2689240	2637663	1473616	1490946
7	Txt File 7	559	560	279	287
8	Vb File 1	34627	28975	20362	20601
9	Vb File 2	8993	7424	5307	5416
10	Vb File 3	15940	12922	9109	9232

TABLE V

COMPARISON BASED ON COMPRESSION RATIO

Original File Size			**Compression Ratio**		
No.	File Name	File Size (bytes)	Run-Length Encoding (%)	Huffman Coding (%)	Shannon-Fano (%)
1	Txt File 1	12883256	94.13368	52.44494	52.7383
2	Txt File 2	373874	99.28906	59.41494	59.78859
3	Txt File 3	18757175	99.72138	50.7272	52.18068
4	Txt File 4	611682	98.20789	54.17112	54.8844
5	Txt File 5	28081207	99.96185	50.69911	52.22718
6	Txt File 6	2689240	98.0821	54.79675	55.44117
7	Txt File 7	559	100.1789	49.91055	51.34168
8	Vb File 1	34627	83.67747	58.80383	59.49404
9	Vb File 2	8993	82.5531	59.01257	60.22462
10	Vb File 3	15940	81.0665	57.14555	57.91719

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 8.165 |

|| Volume 10, Issue 7, July 2022 ||

| DOI: 10.15680/IJIRCCE.2022.1007001|

TABLE VI

COMPARISON BASED ON COMPRESSED TIME

Original File Size			Compression Time(ms)		
No.	File Name	File Size (bytes)	Run-Length Encoding (ms)	Huffman Coding (ms)	Shannon-Fano (ms)
1	Txt File 1	12883256	530	3135	3155
2	Txt File 2	373874	15	100	105
3	Txt File 3	18757175	1070	4903	4741
4	Txt File 4	611682	35	180	165
5	Txt File 5	28081207	1615	7296	7141
6	Txt File 6	2689240	156	688	672
7	Txt File 7	559	1	5	5
8	Vb File 1	34627	1	16	16
9	Vb File 2	8993	1	5	5
10	Vb File 3	15940	1	8	8

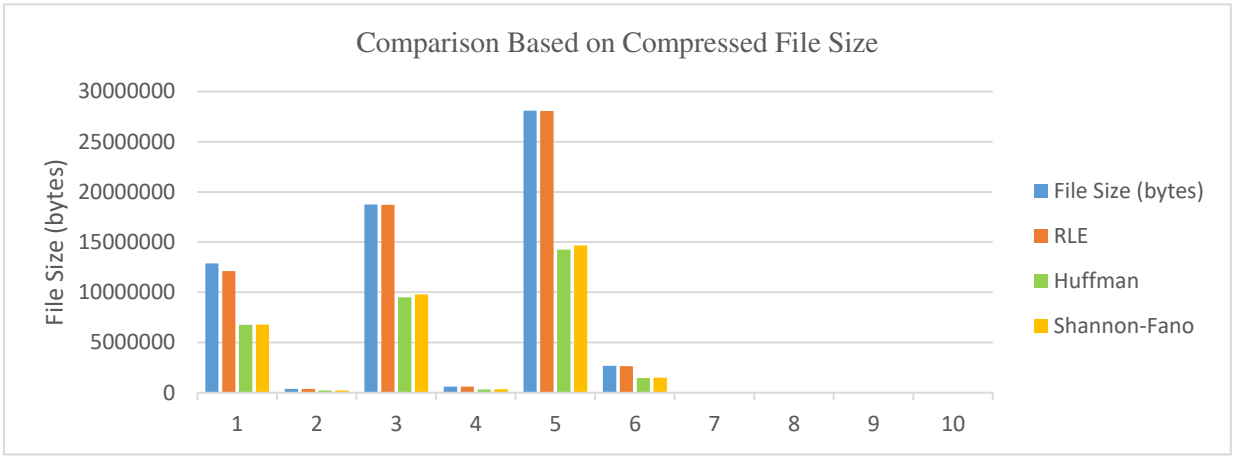

Fig. 1 Compressed File Size

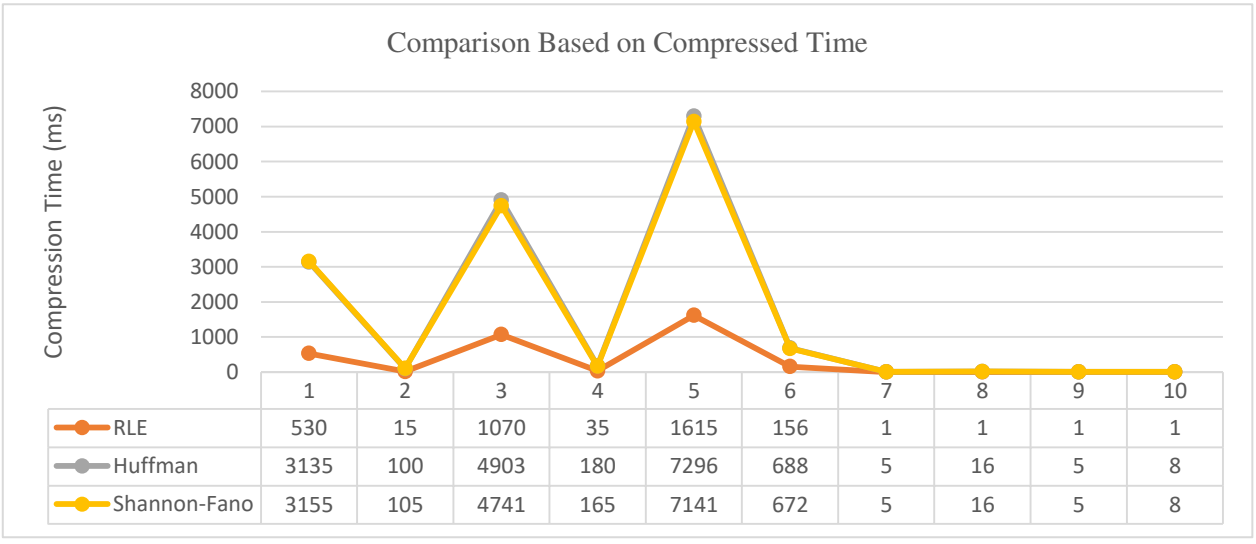

Fig. 2 Compression Time

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 8.165 |

|| Volume 10, Issue 7, July 2022 ||

| DOI: 10.15680/IJIRCCE.2022.1007001|

B. *Discussion*

The performance of the algorithms was analysed base on the compression size, the compression ratio and lastly, the compression time (speed of execution).

In Table IV; Run-Length Encoding does not perform better in terms of compression size. The compressed size of file number 7 is even bigger than that of the original size. In terms of compression size, Huffman Coding does better, followed by Shannon-Fano algorithm and lastly, the Run-Length Encoding (RLE) algorithm. In terms of Compression ratio percentage, Huffman Coding performs better, in the range of 49% to 60%, and the least performing is the Run-Length Encoding of 81% to 101%.

Finally, the Huffman Coding again performs better than the rest in terms of Compression or execution time.

V. CONCLUSION AND FUTURE WORK

A. Conclusion

The lossless data compression schemes evaluated all performs better in their own terms based on the metrics of compression size, ratio, and time of execution used, but in specific terms, the Huffman's compression scheme outperformed all the algorithms.

Notice that, the content of the document and type of document also influences the metrics aside the robustness or efficiency of the algorithms and hence the variations in some of the metrics.

Finally, the Huffman algorithm performs better than the Shannon-Fano Algorithm, followed by the Run-Length Encoding (RLE) algorithm.

B. Extension for Future Work/Research

In this research, the performance of three (3) lossless compression algorithms including the Run-Length Encoding (RLE), Huffman Coding and Shannon-Fano Coding were evaluated. They were tested on some selected text files of varying size and content.

In future, large number of algorithms can be implemented for larger data types including audio, video and image data types. An application can also be developed to detect the type of file and compress with the efficient appropriate compression method.

Furthermore, Redundant Residue Number System (RRSN) as an error detection and correction mechanism can be applied to these algorithms to deal with errors that may occur during the compression process.

REFERENCES

[1] Anshul, A. R., Ravi, A. R., and Pooja, R. (2019). Comparative Study of Data Compression Techniques. *International Journal of Computer Applications (0975 – 8887) Volume 178 – No. 28.*

[2] Barath, C. K., Varun, M. K. and Gayathri, T. (2013). Technique of Data Analysis and File Compression using Huffman Algorithm. *International Journal of Advanced Research in Computer Science and Software Engineering,* Vol.3 Issue 11, Pg346-348.

[3] Engineering and Technology History Wiki (2014). History of Lossless Data Compression Algorithms. *Retrieved from www.ethw.org/History_of_Lossless_Data_Compression_Algorithms.*

[4] Apoorv, V. S. and Garima, S. (2012).A Survey on Different Compression Techniques. *International Journal of Science and Research (IJSR)*

[5] Pannirselvam, S. andSelvanayagi, D. (2015). A Comparative Analysis on Different Techniques in Text Compression, *International Journal of Innovative Technology and Creative Engineering,* Vol.5 No.8, Pg283-287.

[6] Kodituwakku, S. R. and Amarasinghe,U. S. (2010). Comparison of Lossless Data Compression Algorithms for Text Data.*Indian Journal of Computer Science and Engineering*, Vol. 1 (4), Pg416 – 425.

[7] Mohammad, H. (2012). A Survey of Data Compression Algorithms and their Applications. *Network Systems Lab, School of Computing Science, Simon Fraser University, BC, Canada.*

[8] Mohammed, A. and Ibrahim, M. M. E. (2007). Comparative Study between Various Algorithms of Data Compression Techniques. *International Journal of Computer Science and Network Security*, Vol.7 (4), Pg281-291.

[9] Seema, A., and Priyanka, A. (2017). Comparative Study on Lossless Data Compression Techniques.*International Journal of Scientific Research and Management (IJSRM),* Vol.5, Issue 8, Pg6630-6637.

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 8.165 |

|| Volume 10, Issue 7, July 2022 ||

| DOI: 10.15680/IJIRCCE.2022.1007001|

[10] Senthil, S. and Robert, L. (2011). A Comparative Study of Text Compression Algorithms. *International Journal of Wisdom Based Computing,* Vol. 1, No.3, Pg68-76.

[11] Alhassan, A-B.,Gbolagade, K. A., and Bankas, E. K. (2015). A Novel and Efficient LZW-RNS Scheme for Enhanced Information Compression and Security. *International Journal of Advanced Research in Computer Engineering and Technology*. Volume 4(11). Pg1450-4019.

[12] Alhassan, A-B.,Ibrahim, S., and Agbedemnab, P.(2015). The Huffman's Method of Secured Data Encoding and Error Correction using Residue Number System (RNS). *Communications on Applied Electronics*. Volume 2(9). Pg14-18.

[13] Alhassan, A-B.,Bankas, E. K., and Agbedemnab, P. A. (2017). Computer Arithmetic Aided Lempel-Ziv-Welch's Algorithm using the Moduli Set $\{2^n - 1, 2^n, 2^n + 1, 2^{n+1} - 1\}$ for Fast and Secured Transmission of Data via Network Communication Channels. *American International Journal of Research in Science, Technology, Engineering & Mathematics*. Volume 19(1). Pg62-68.

INNO SPACE
SJIF Scientific Journal Impact Factor

Impact Factor: 8.165

doi cross ref

ISSN INTERNATIONAL STANDARD SERIAL NUMBER INDIA

निस्केयर NISCAIR

INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN COMPUTER & COMMUNICATION ENGINEERING

9940 572 462 6381 907 438 ijircce@gmail.com

www.ijircce.com

Scan to save the contact details