IT Journal Research and Development (ITJRD)
Vol.7, No.2, March 2023, E-ISSN : 2528-4053 | P-ISSN : 2528-4061
DOI : 10.25299/itjrd.2022.10437

Comparison of Huffman Algorithm and Lempel Ziv Welch Algorithm in Text File Compression

Muhammad Alif[1], Mohamad Nurkamal Fauzan[2], Cahyo Prianto[3]
Department of Informatics Engineering, Polytechnic Pos Indonesia
alifmuhammad7210@gmail.com[1], m.nurkamal.f@ulbi.ac.id[2], cahyoprinato@ulbi.ac.id[3]

Article Info

Article history:

Received Sep 01, 2022
Revised Oct 02, 2022
Accepted Dec 26, 2022

Keyword:

Comparison
Compression
Lossless
Huffman
Lempel Ziv Welch

ABSTRACT

The development of data storage hardware has been very rapid over time. In line with the development of storage hardware, the amount of digital data shared on the internet is increasing every day. That way, no matter how big the size of the storage device we have, of course, it will only be a matter of time until that storage space is exhausted. Therefore, in terms of maximizing storage space, a technique called compression appeared. This study focuses on a comparative analysis of two lossless compression technique algorithms, namely the Huffman algorithm and Lempel Ziv Welch (LZW). A number of test files with different file types are applied to both algorithms that are compared. The performance of the algorithm is determined based on the comparison of space-saving and compression time. The test results showed that the Lempel Ziv Welch (LZW) algorithm was superior to Huffman's algorithm in.txt file type compression and.csv. The average space savings produced were 63.85% and 77.56%. The degree of compression speed that each algorithm produces is directly proportional to the file size.

Corresponding Author:

Muhammad Nurkamal Fauzan
Department of Informatics Engineering
Polytechnic Pos Indonesia
Sariasih Street No. 54, Sarijadi, Sukasari, Bandung City, West Java 40151, Indonesia
Email: m.nurkamal.f@ulbi.ac.id

1. INTRODUCTION

Technology is currently growing rapidly along with the changing era and holding important role in the way humans communicate [1]. Currently, advances in information technology changing human habits to exchange data and information, thereby increasing digital data requests. The growth of devices and storage media is very fast from time to time. Today's data storage hardware is very capable in terms of storing hundreds of gigabytes even up to terabytes per 1 unit of hardware storage [2]. Currently, the capacity of hardware storage space often found/circulated in the market has reached terabyte units or the equivalent of 1.000.000 megabytes [3]. If the analogy has a text file with a size of 10 megabytes to be stored in the storage hardware, then the hardware can only accommodate text files as many as 100.000 files. With fast-growing hardware storage, it makes the costs of storage hardware to increase. On the other hand, the need for storage hardware is urgently needed in save the resulting file at any time. Currently, there are many data compression applications that have spread on the internet, both web-based and mobile-based. However, the data security issues offered by the compression application need to be asked again considering that data

security is a very important aspect of an application. Data security is needed to prevent all forms of information from reaching other parties who are not interested [4].

In line with the growth of storage hardware, the amount of digital data that is shared on the internet is increasing day by day and is very easy to access[2]. That way no matter how big the size of the storage device we have, of course, it will only be a matter of time until the storage space is exhausted [2]. Using effective and efficient storage media is everyone's desire. The size of the data has a big impact on the storage space, and also affects the data transmission speed [5]. Therefore, in terms of maximizing storage space then appear a technique called compression.

Data compression is a study in computer science to reduce the file size before storing or moving data into storage media. There are two types of data compression techniques, namely lossy compression techniques, and Lossless compression techniques [6]. The lossy compression method reduces the file size by removing some of the original data of the file. The file result cannot be completely reconstructed. The general lossy compression method is used to compress file types where data loss is not visible, such as files video, audio, and images [7]. Lossless Compression reduces file size without loss of data (bits). In the Lossless compression method, the compressed data can be restored to its original form, this process is called decompression [8]. Lossless data compression algorithms can be categorized into two types, namely entropy-based encoding, and dictionary-based encoding. Examples of entropy-based compression algorithms include Huffman Encoding, Run Length Encoding, Arithmetic Coding, and Shannon-Fano Coding. The compression algorithm based on dictionaries includes LZ77, LZ78, and Lempel-Ziv-Welch (LZW) [9].

In the study "Comparison of Huffman Method and Run Length Encoding on Document Compression" by Pujianto, Mujito, Basuki Hari Prasetyo, and Anang Prabowo. In this study, a comparison of two compression methods was discussed, namely the Run Length Encoding method and the Huffman method. The data types used are document files with docx file types, pdf files, xlsx files, and pptx files. To obtain the results of the comparative analysis, several parameters were used in calculating the performance of the two compression algorithms, including the number of compression results, the number of decompression results, the compression ratio, and how long the compression time required. The results of this study show that Huffman's algorithm is superior compared to the Run Length Encoding algorithm. It is shown that there is 1 pdf file that has increased in file size after compression using the Run Length Encoding algorithm [13]. Another study was "A Review on Different Types of Lossless Data Compression Techniques" by Anshul Gupta and Prof. Sumit Nigam. A comparative analysis of various kinds of lossless compression techniques was carried out. The algorithms compared include Huffman Coding, Shannon-Fano Coding, RLE, LZW, LZ77, LZ78, and Lempel-Zev Welch. The results showed that data compression was distinguished by two, namely entropy-based compression and dictionary-based compression. The LZW algorithm shows greater efficiency in saving space than other algorithms. The results showed that the LZW algorithm could save space by 81.31% [9].

In this study, an analysis of two lossless compression algorithms will be carried out to obtain the maximum algorithm for compressing files, where the Huffman algorithm will be used as a representation of entropy-based algorithms and the Lempel Ziv Welch (LZW) algorithm as a dictionary-based algorithm representation. In this study, there are 3 types of input files that will be used, namely .txt extension files, .csv extension files, and .docx extension files. Each test file type consists of 12 test files that have different file sizes. Test files with txt extensions contain text data written in Indonesian. The csv extension test file contains text data written in Indonesian. The docx extension test file consists of a file that contains text only and a file containing text data along with images written in Indonesian. The results of this study will show a comparison of compression using the Huffman algorithm and the Lempel Ziv Welch algorithm, which is shown using several parameters, namely the initial file size before compressing, the file size after compression, the space saving value, and the length of time of the compression process.

2. RESEARCH METHOD

2.1. Compression

Compression means reducing or compressing. Data compression is a method to compress data or files to a smaller size than the original file, thereby reducing storage space and transmission time for file transfer over the network. Most of the computer file types have the same data. With a file compression program, you can eliminate the redundancy of the data owned, then list that information once and then refer back to it whenever it appears in the original program. Compression works by scanning the entire file for identified similar or repetitive data and patterns, then replacing duplicates with a unique identifier. This identifier is typically much smaller than the original word and takes up less space. As a result, the compressed file is significantly smaller in size. Data compression is possible because many redundant bits are found in most real-world data. Data compression can be done on various types of files including text, audio, image, and video files. With compression, one can save more storage space [10].

2.2. Lossless Compression

In lossless compression, a compression process is carried out on a file to reduce the size of the file. With the lossless compression method, the original data from the file is maintained without any damage or data loss when the data is not compressed. With this advantage, the lossless compression method is very suitable in compressing with text file types, where the data or information contained in it is very important [11].

2.3. Huffman Algorithm

Huffman's algorithm was first discovered in 1952, by A named David Huffman. The Huffman Algorithm is a type of Lossless Algorithm entropy-based compression [9]. there is no data loss during the compression process [7]. The draft used by the Huffman algorithm comes from the binary tree used for performing the data compression process [12]. Huffman's algorithm uses coding based on variable length where all characters are coded variable length based on how often they appear in the text. Characters that appear most often receive the smallest code while the least frequent get the largest code [7]. Code Huffman used almost the same principle as Morse code where for each character encoded into a series of bits, each character's most frequent occurrences are encoded with shorter sequences of bits, and characters with the fewest occurrences are encoded with the longer bit set [13]. There are four phases in the Huffman algorithm for text compression [14]. The first phase is to group the character of the file to be compressed. The Second Phase is building Huffman. The third stage is coding. The last phase, the fourth phase is to perform bit code generation. The principle of the Huffman algorithm is that every character with multiple occurrences is encoded with short bit strings and characters that appear slightly bit-encoding with a longer series.

2.4. Lempel Ziv Welch Algorithm

The Lempel Ziv Welch (LZW) algorithm is an algorithm that was found and named after its inventors Abraham Lempel, Jakob Ziv, and Terry Welch [7]. The LZW algorithm is a type of dictionary-based lossless compression algorithm [15]. The simple LZW algorithm only replaces the character string with a single code. Data compression with algorithm LZW starts from reading the sequence of symbols, then grouping the symbols into strings, and in the end converting the string to code [16]. During the compression process, variables; CHAR and STR are used. CHAR holds a single character (that is, a single byte value between 0 and 255) while STR captures a group of one or more characters. Every character in STR is one byte. The LZW algorithm starts by taking the first byte of the file input and storing it in STR. After that, looping every additional byte of the input file started. The next byte read from the input file is stored in CHAR, thus making a data table. This table is scanned to confirm whether the code has been assigned to the circuit STR+CHAR. It only outputs the code for the STR when a match in the table is not visible. Otherwise, the STR+CHAR string is stored in STR, without further action [7].

2.5. Space Saving

Space saving is known as size reduction compared to uncompressed size [10]. Space saving can be calculated by the following equation,

$$Space\ Saving = 1 - \frac{Compressed\ file\ size}{Uncompressed\ file\ size}\ x\ 100\% \qquad (1)$$

2.5. Compression Time

Compression time is the length of time it takes to execute the data compression algorithm used [17].

2.6. Research Workflow

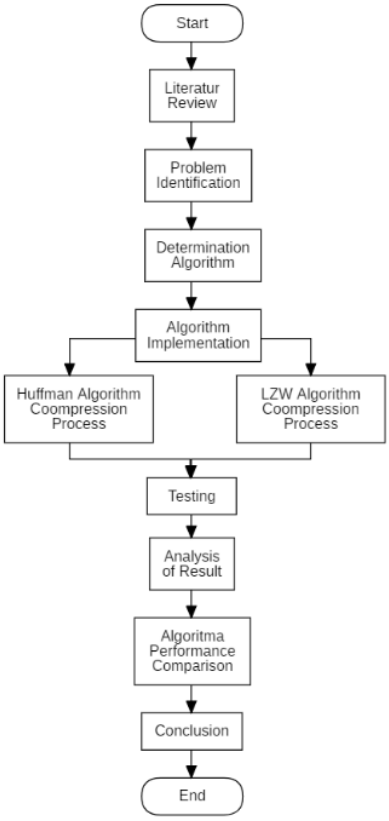

Figure 1. Research Workflow

This research starts by conducting a literature review to determine the algorithm to be compared, then proceeds to the problem identification stage, namely formulating the problems found in the form of questions. The next step is the determination of the algorithm to be compared, at this stage it is proposed the Huffman algorithm and the Lempel Ziv Welch algorithm. Then the Huffman algorithm and the Lempel Ziv Welch algorithm were implemented into an application using a programming language. This implementation process takes place in 2 processes, namely the process on the Huffman algorithm and the process on the Lempel Ziv Welch algorithm. The next step is to test 2 compared algorithms to see the performance of the 2 algorithms. algorithms are compared. The last step is to draw the conclusion of an algorithm that is superior to the 2 algorithms that have been compared.

3. RESULTS AND ANALYSIS

3.1. Algorithm Implementation

Furthermore, the proposed algorithms are the Huffman algorithm and the Lempel Ziv Welch algorithm to compress text files. Huffman's algorithm and Lempel Ziv Welch's algorithm

are translated into a programming language that was later created into a web-based application to carry out the compression process.

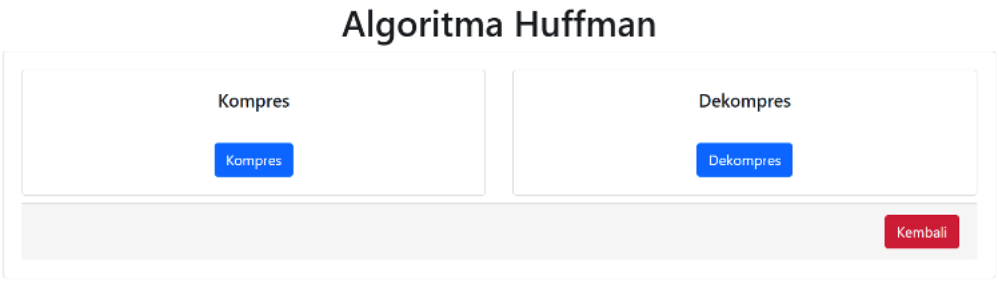

Figure 2. Huffman Algorithm Action Selection Page

Figure 2 is a page view of the selection of actions to be performed on the compression process with the Huffman algorithm. On this page there are two types of actions that can be selected, namely compressing data or decompressing data.

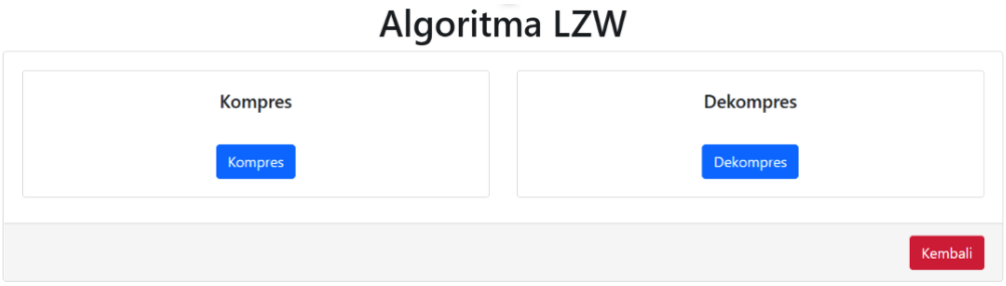

Figure 3. LZW Algorithm Action Selection Page

Figure 3 is a page view of the selection of actions to be performed on the compression process with the Lempel Ziv Welch algorithm. On this page there are two types of actions that can be selected, namely compressing data or decompressing data.

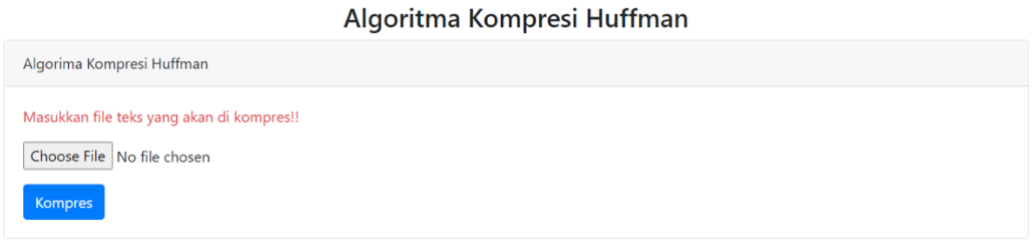

Figure 4. File Input Page on Huffman Algorithm

Figure 4 is a display image for input or inserting a file to be compressed using the Huffman algorithm. On this page, there is a sentence affirming that the accepted file type is a text file type.

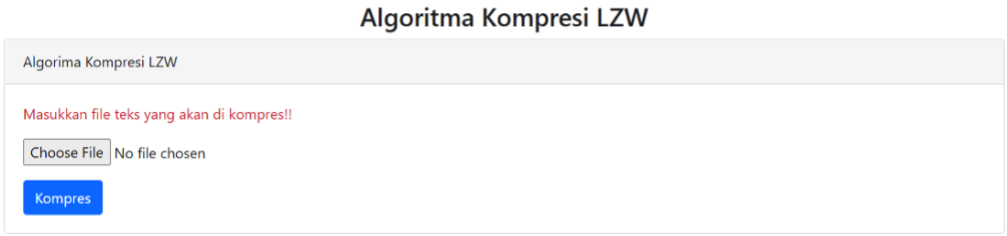

Figure 5. File Input Page on LZW Algorithm

Figure 5 is a display image for input or inserting a file to be compressed using the Lempel Ziv Welch algorithm. Similar to Figure 4, a statement on this page states that the allowed file type is a text file type.

Figure 6. Huffman Algorithm Compression Result Page

Figure 6 above is a display after compression using the Huffman algorithm. This page will display information in the form of file names, file sizes before and after compressing, compression ratio, and compression time.

Figure 7. LZW Algorithm Compression Result Page

Figure 7 above is a display after compression using the Lempel Ziv Welch algorithm. This page will display information in the form of file names, file sizes before and after compressing, compression ratio, and compression time.

3.2. TXT File Testing

The following is a summary table comparing the results of testing the compression process using the Huffman algorithm and the Lempel Ziv Welch (LZW) algorithm against 12 test files with extension .txt.

Table 1. TXT File test Result

No	File Name	Initial file size	File size after compression (bytes)	Space Saving (%)	Compression Time (second)

		(bytes)	*Huffman*	LZW	*Huffman*	LZW	*Huffman*	LZW
1	Test1.txt	104	132	95	-18.27	8.65	0.01	0.03
2	Test2.txt	523	372	359	28.87	31.36	0.08	0
3	Test3.txt	5086	2854	2574	43.89	49.39	0.34	0.02
4	Test4.txt	9661	5313	4409	45.01	54.36	0.65	0.06
5	Test5.txt	19654	10698	7701	45.57	60.82	1.3	0.05
6	Test6.txt	50636	27404	16180	45.88	68.05	3.21	0.09
7	Test7.txt	101413	54750	28766	46.01	71.63	6.39	0.17
8	Test8.txt	511338	275684	111337	46.09	78.23	32.27	0.9
9	Test9.txt	1025015	552510	193356	46.1	81.14	70.1	1.64
10	Test10.txt	2047233	1085909	339715	46.96	83.41	126.62	3.31
11	Test11.txt	5125077	2762017	601757	46.11	88.26	315.1	8.87
12	Test12.txt	9489772	5776526	936857	39.13	90.86	313.19	12.76
		Average			**38.45**	**63.85**	**72.44**	**2.33**

Table 1 shows the comparison of the results of the compression process 12 test files with the TXT extension. The information displayed includes the file name; the size of 12 files before compression; a comparison of the sizes of 12 files compressed using the Huffman algorithm and the Lempel Ziv Welch algorithm; a comparison of the space-saving value of 12 files compressed using the Huffman algorithm and the Lempel Ziv Welch algorithm; and a comparison of the compression time of 12 files compressed using the Huffman algorithm and the Lempel Ziv Welch algorithm. The test was carried out in 2 stages. The first stage testing 12 test files using the Huffman algorithm. The second stage is testing of 12 test files using the Lempel Ziv Welch algorithm. Testing using the Huffman algorithm begins by inputting each test file (starting from the test1 file.txt to test12.txt) alternately into the application, then selecting the type of Huffman algorithm for the compression process, then the application performs the compression process using the Huffman algorithm. Next, the application displays the information resulting from the compression process, which includes the file name, file size before compressing, file size after compressing, space saving value, and compression time. Testing using the Lempel Ziv Welch algorithm starts by inputting each test file (starting from the test1 file.txt to test12.txt) alternately into the application, then chooses the type of Lempel Ziv Welch algorithm for the compression process, and then the application carries out the compression process using the Lempel Ziv Welch algorithm. Next, the application displays the information resulting from the compression process, which includes the file name, file size before compressing, file size after compressing, space saving value, and compression time. Information on the compression results of 12 test files using the Huffman algorithm and information on the results of the compression of 12 test files using the Lempel Ziv Welch algorithm are summarized in a table that can be seen in table 1 above. From table 1 above, the comparison results were obtained, namely the average space saving by the Huffman algorithm was 38.45, the average space saving by the Lempel Ziv Welch algorithm was 63.85, the average compression time of the Huffman algorithm was 72.44, and the compression time of the Lempel Ziv Welch algorithm was 2.33.

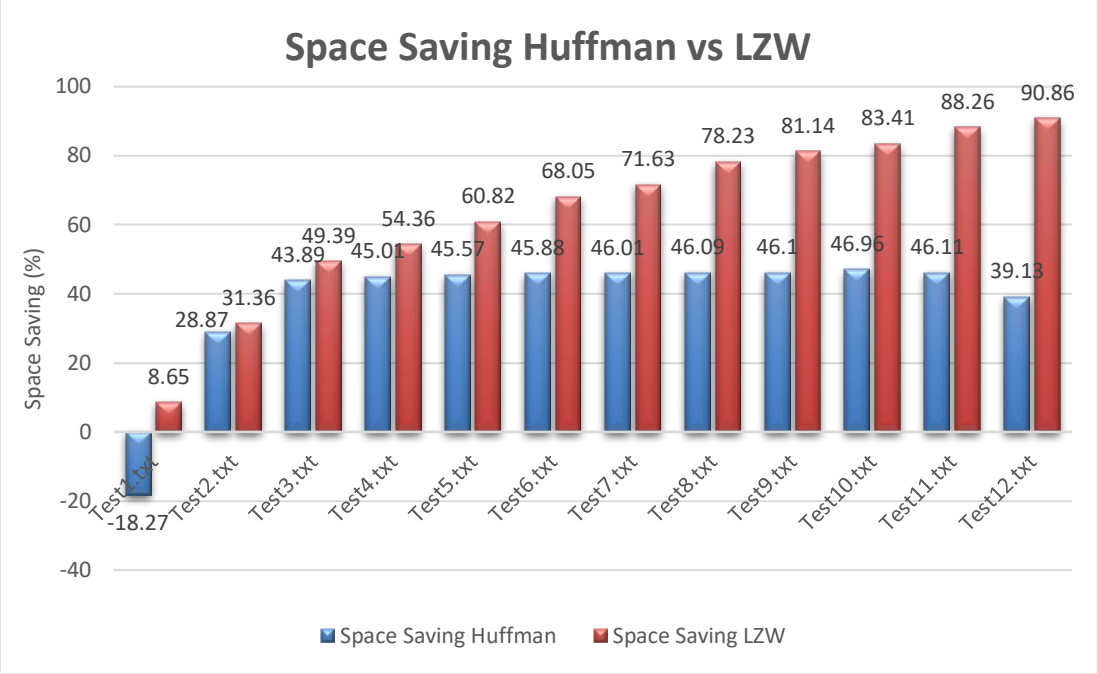

Figure 8. Space Saving Comparison of Huffman and LZW Algorithms on TXT Files

Figure 8 is a comparison diagram of the space savings produced by the Huffman algorithm and the LZW algorithm. Huffman algorithm space saving is indicated by blue, LZW algorithm space saving is indicated by red. Huffman's algorithm shows the highest space saving value of 46.96 and the lowest space saving value of 28.87. LZW algorithm shows the highest space saving value of 90.86 and the lowest space saving value of 8.65.

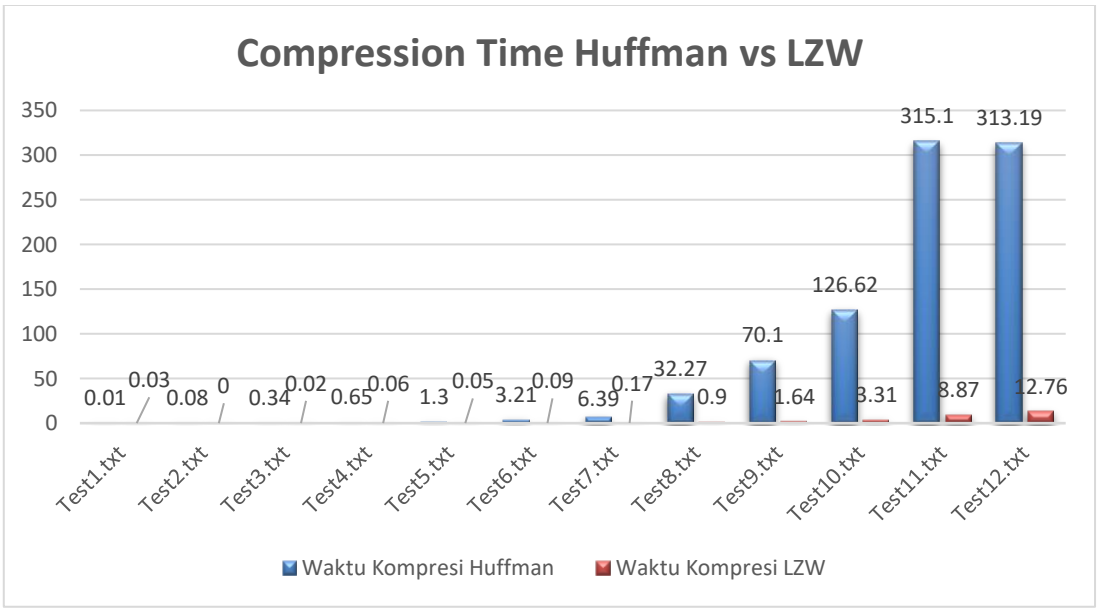

Figure 9. Comparison of Compression Time of Huffman Algorithm and LZW Algorithm Against TXT File

Figure 9 is a comparison diagram of the compression time generated by the Huffman algorithm and the LZW algorithm. Huffman's algorithm compression time is indicated by blue, LZW algorithm compression time is indicated by red. Based on the diagram, it can be concluded that the LZW algorithm is faster in compressing TXT files than Huffman algorithms.

3.3. CSV File Testing

The following is a summary table comparing the results of testing the compression process using the Huffman algorithm and the Lempel Ziv Welch (LZW) algorithm against 12 test files with extension .csv.

Table 2. CSV File test Result

No	File Name	Initial File Size	File size after compression		Space Saving (%)		Compression Time	
		(bytes)	(bytes)				(Second)	
			Huffman	LZW	*Huffman*	LZW	*Huffman*	LZW
1	Test1.csv	2350	1377	1044	41.4	55.57	0.06	0.04
2	Test2.csv	5149	3237	2832	37.13	45	0.22	0.02
3	Test3.csv	34997	20943	13180	40.16	62.34	1.22	0.07
4	Test4.csv	51589	32238	9756	37.63	81.13	1.89	0.08
5	Test5.csv	66539	39200	9799	41.08	85.27	2.79	0.14
6	Test6.csv	113479	70415	19018	37.95	83.24	4.69	0.19
7	Test7.csv	234466	149727	64225	36.14	72.61	9.49	0.45
8	Test8.csv	609462	23347	102861	61.73	83.12	14.3	0.98
9	Test9.csv	1182501	770623	114255	34.83	90.34	46.93	1.66
10	Test10.csv	2762130	1710073	293707	38.09	89.37	109.13	4.22
11	Test11.csv	6866709	4445867	419709	35.25	93.89	543.83	6.77
12	Test12.csv	9489772	5776526	1060770	39.13	88.82	700.87	7.28
Average					**40.04**	**77.56**	**119.62**	**1.83**

Table 2 shows the comparison of the results of the compression process 12 test files with the CSV extension. The information displayed includes the file name, the size of 12 files before compression, a comparison of the sizes of 12 files compressed using the Huffman algorithm and the Lempel Ziv Welch algorithm, a comparison of the space-saving value of 12 files compressed using the Huffman algorithm and the Lempel Ziv Welch algorithm, and a comparison of the compression time of 12 files compressed using the Huffman algorithm and the Lempel Ziv Welch algorithm. The test was carried out in 2 stages. The first stage testing 12 test files using the Huffman algorithm. The second stage is testing of 12 test files using the Lempel Ziv Welch algorithm. Testing using the Huffman algorithm begins by inputting each test file (starting from the test1 file.csv to test12.csv) alternately into the application, then selecting the type of Huffman algorithm for the compression process, then the application performs the compression process using the Huffman algorithm. Next, the application displays the information resulting from the compression process, which includes the file name, file size before compressing, file size after compressing, space saving value, and compression time. Testing using the Lempel Ziv Welch algorithm starts by inputting each test file (starting from the test1 file.csv to test12.csv) alternately into the application, then chooses the type of Lempel Ziv Welch algorithm for the compression process, and then the application carries out the compression process using the Lempel Ziv Welch algorithm. Next, the application displays the information resulting from the compression process, which includes the file name, file size before compressing, file size after compressing, space saving value, and compression time. Information on the compression results of 12 test files using the Huffman algorithm and information on the results of the compression of 12 test files using the Lempel Ziv Welch algorithm are summarized in a table that can be seen in table 2 above. From table 2 above, the comparison results were obtained, namely the average space saving by the Huffman algorithm was 40.04, the average space saving by the Lempel Ziv Welch algorithm was 77.56, the average compression time of the Huffman algorithm was 119.62, and the compression time of the Lempel Ziv Welch algorithm was 1.83.

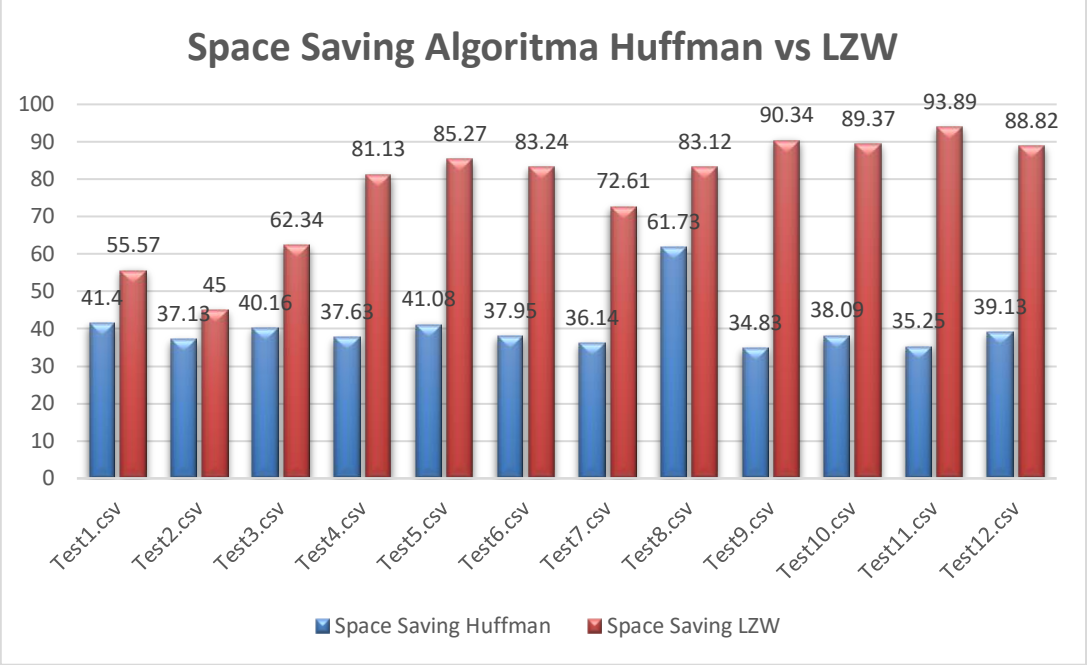

Figure 10. Space Saving Comparison of Huffman and LZW Algorithms on CSV Files

Figure 10 is a comparison diagram of the space savings produced by the Huffman algorithm and the LZW algorithm. Huffman algorithm space saving is indicated by blue, LZW algorithm space saving is indicated by red. Huffman's algorithm shows the highest space saving value of 61.73 and the lowest space saving value of 34.83. LZW algorithm shows the highest space saving value of 93.89 and the lowest space saving value of 45.

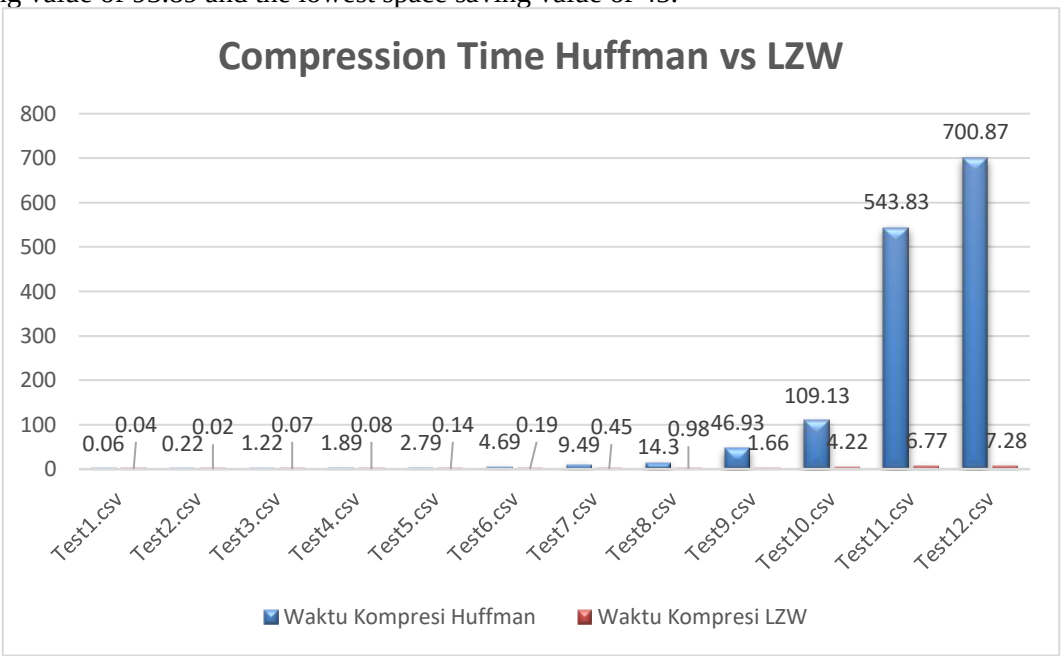

Figure 11. Comparison of Compression Time of Huffman Algorithm and LZW Algorithm Against CSV File

Figure 9 is a comparison diagram of the compression time generated by the Huffman algorithm and the LZW algorithm. Huffman's algorithm compression time is indicated by blue, LZW algorithm compression time is indicated by red. Based on the diagram, it can be concluded that the LZW algorithm is faster in compressing CSV files than Huffman algorithms.

3.4. DOCX File Testing

The following is a summary table comparing the results of testing the compression process using the Huffman algorithm and the Lempel Ziv Welch (LZW) algorithm against 12 test files with extension .docx.

Table 3. DOCX File Test Result

No	File Name	Initial File size (bytes)	File size after compression (bytes)		Space Saving (%)		Compression time (Second)	
			Huffman	LZW	*Huffman*	LZW	*Huffman*	LZW
1	Test1.docx	16202	15841	19905	2.23	-22.86	1.95	0.11
2	Test2.docx	34375	33920	45022	1.32	-30.97	4.14	0.14
3	Test3.docx	59341	59327	79977	0.07	-34.78	8.08	0.49
4	Test4.docx	75685	75933	101557	-0.33	-34.18	10.04	0.33
5	Test5.docx	111303	112094	149681	-0.71	-34.48	13.7	0.49
6	Test6.docx	291041	291287	335451	-0.08	-15.26	35.58	1.17
7	Test7.docx	448352	440213	481231	1.82	-7.33	54.61	1.72
8	Test8.docx	551820	552833	664338	-0.18	-20.39	69.24	2.36
9	Test9.docx	1026736	1027598	1225358	-0.12	-19.34	122.78	4.43
10	Test10.docx	2673142	2658357	3065425	0.55	-14.27	360.75	11.34
11	Test11.docx	3909414	3906816	4663422	0.07	-19.29	527.11	18.01
12	Test12.docx	4665822	4622234	4837630	0.93	-3.68	601.67	19.16
		Average			**0.46**	**-21.40**	**150.80**	**4.98**

Table 3 shows the comparison of the results of the compression process 12 test files with the DOCX extension. The information displayed includes the file name, the size of 12 files before compression, a comparison of the sizes of 12 files compressed using the Huffman algorithm and the Lempel Ziv Welch algorithm, a comparison of the space-saving value of 12 files compressed using the Huffman algorithm and the Lempel Ziv Welch algorithm, and a comparison of the compression time of 12 files compressed using the Huffman algorithm and the Lempel Ziv Welch algorithm. The test was carried out in 2 stages. The first stage testing 12 test files using the Huffman algorithm. The second stage is testing of 12 test files using the Lempel Ziv Welch algorithm. Testing using the Huffman algorithm begins by inputting each test file (starting from the test1 file.docx to test12.docx) alternately into the application, then selecting the type of Huffman algorithm for the compression process, then the application performs the compression process using the Huffman algorithm. Next, the application displays the information resulting from the compression process, which includes the file name, file size before compressing, file size after compressing, space saving value, and compression time. Testing using the Lempel Ziv Welch algorithm starts by inputting each test file (starting from the test1 file.docx to test12.docx) alternately into the application, then chooses the type of Lempel Ziv Welch algorithm for the compression process, and then the application carries out the compression process using the Lempel Ziv Welch algorithm. Next, the application displays the information resulting from the compression process, which includes the file name, file size before compressing, file size after compressing, space saving value, and compression time. Information on the compression results of 12 test files using the Huffman algorithm and information on the results of the compression of 12 test files using the Lempel Ziv Welch algorithm are summarized in a table that can be seen in table 3 above. From table 3 above, the comparison results were obtained, namely the average space saving by the Huffman algorithm was 0.46, the average space saving by the Lempel Ziv Welch algorithm was -21.40, the average compression time of the Huffman algorithm was 150.80, and the compression time of the Lempel Ziv Welch algorithm was 4.98.

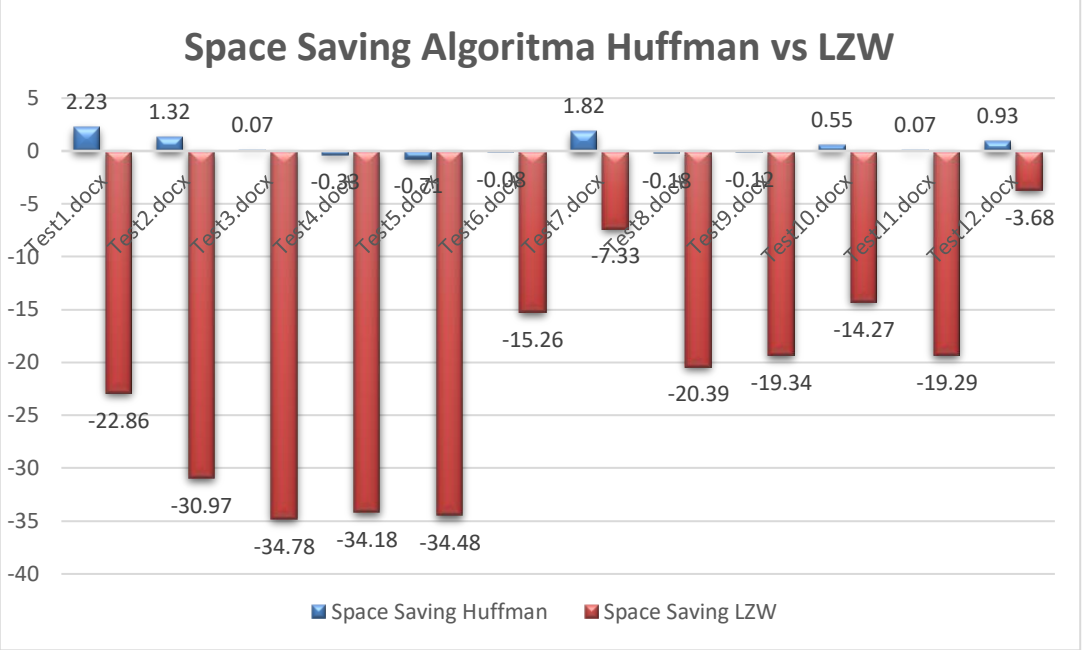

Figure 12. Space Saving Comparison of Huffman and LZW Algorithms on DOCX Files

Figure 12 shows the percentage of space saving from 2 algorithms compared, namely the Huffman algorithm and the Lempel Ziv Welch (LZW) algorithm. The Lempel Ziv Welch (LZW) algorithm shows the results of compression of 12 files failed. In figure 12, it is shown that in the space saving of 12 files using the Huffman algorithm 7 test files were successfully compressed and got space saving results below 3% and 5 test files failed and got minus space saving results.

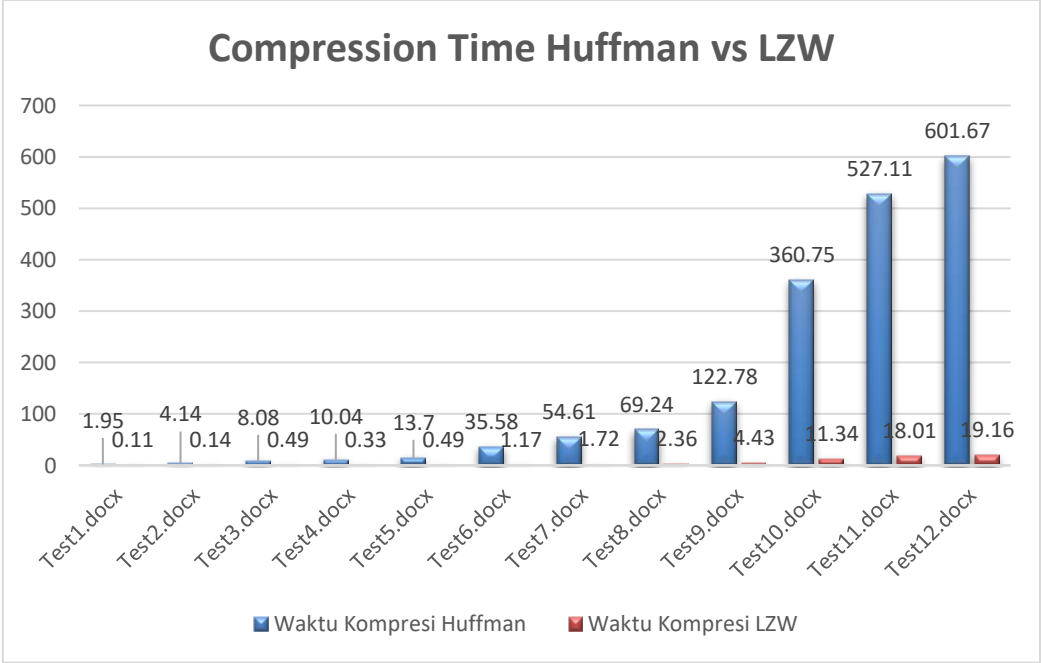

Figure 13. Comparison of Compression Time of Huffman Algorithm and LZW Algorithm Against DOCX File

Figure 13 is a comparison diagram of the compression time generated by the Huffman algorithm and the LZW algorithm. Huffman's algorithm compression time is indicated by blue,

LZW algorithm compression time is indicated by red. Based on the diagram, it can be concluded that the LZW algorithm is faster in compressing CSV files than Huffman algorithms

3.4. Discussion

Data compression is a study in computer science to reduce the size of a file before storing or moving data to a storage medium. There are two types of data compression techniques, namely lossy compression techniques and lossless compression techniques. An analysis of two lossless compression algorithms will be carried out to obtain the maximum algorithm for compressing files, of which the Huffman algorithm will be used as a representation of the entropy-based algorithm and the Lempel-Ziv-Welch (LZW) algorithm will be used as a representative of the dictionary-based algorithm. Based on tests conducted on TXT, CSV, and DOCX files, there are a few things to be concerned about. In the compression of TXT files using the Huffman algorithm, there is one file that fails. This is shown from the test results in the first test file where the size of the file after compressing increased from 104 bytes to 123 bytes. Because the file size increases, it causes the file compression value to be minus with a value of -18.27%. Based on these results, Huffman's algorithm will be effective in carrying out the compression process on files larger than 100 bytes in size. In CSV file compression, the Huffman algorithm and the LZW algorithm successfully compress the 12 test file that has been provided. In DOCX file compression, the LZW algorithm shows the result as 12 files failing to be compressed. The diagram shows the level of space saving generated by the Lempel Ziv Welch (LZW) algorithm is below the value of 0 or produces a minus value. With Huffman's algorithm, 7 test files were successfully compressed and got space-saving results below 3%, and 5 test files failed to be compressed and got minus space-saving results. Based on these results, it is considered that the Huffman algorithm is still ineffective because the resulting compression results still have some test files that fail to compress. Based on testing of TXT, CSV, and DOCX files, the compression time generated by the LZW algorithm is faster than the Huffman algorithm.

4. CONCLUSION

Based on the research and tests that have been carried out on the comparison of the Huffman algorithm and the LZW algorithm on the compression of text files, the following conclusions can be drawn:

1. The Lempel Ziv Welch (LZW) algorithm is superior to Huffman’s algorithm in compressing .txt files and .csv files. This is based on the average value of space saving generated by the Lempel Ziv Welch (LZW) algorithm against .txt files and .csv files, namely 63.85% and 77.56%. The average space savings generated by the Huffman algorithm on .txt files and .csv files are 38.45% and 40.04%.
2. In the compression test of the file .docx the Lempel Ziv Welch (LZW) algorithm failed, and the size of the 12 test files increased. Algorithms Huffman is considered still ineffective because the resulting compression results still have some test files that fail to compress and files that have been successfully compressed, the space saving rate is below the value of 3%.
3. The speed of the compression process using the Huffman algorithm and the Lempel Ziv Welch (LZW) algorithm does not depend on the data being processed but is directly proportional to the size of the file to be compressed, which means that the larger the file size that will be compressed, the longer it will take to perform compression. Based on testing of 3 types of test files, namely .txt files, .csv, and .docx files, the compression speed produced by the Lempel Ziv Welch (LZW) algorithm is superior to the Huffman algorithm.

REFERENCES

[1] M. R. Ashila, N. Atikah, D. R. I. M. Setiadi, E. H. Rachmawanto, and C. A. Sari, "Hybrid AES-Huffman Coding for Secure Lossless Transmission."

[2] I. K. Jaya and R. Perangin-angin, "Analisa Perbandingan Rasio Kecepatan Kompresi Algoritma Dynamic Markov Compression dan Huffman," *Publikasi Jurnal dan Penelitian Teknik Informatika*, vol. 2, pp. 78–85, 2018.

[3] Y. Murdianingsih and I. Isbahatunnisa, "IMPLEMENTASI METODE FUZZY TAHANI DALAM MENENTUKAN REKOMENDASI PEMBELIAN LAPTOP (Studi kasus di Toko Mega Alvindo Kalijati Subang)," *Jurnal Teknologi Informasi dan Komunikasi STMIK Subang*, vol. 13, no. 1, pp. 41–51, 2020.

[4] N. I. Putri, R. Komalasari, and Z. Munawar, "PENTINGNYA KEAMANAN DATA DALAM INTELIJEN BISNIS," *Jurnal Sistem Informasi*, vol. 1, no. 2, pp. 41–49, 2020.

[5] Pujianto, Mujito, B. H. Prasetyo, and D. Prabowo, "Perbandingan Metode Huffman dan Run Length Encoding Pada Kompresi Document," *InfoTekjar: Jurnal Nasional Informatika dan Teknologi Jaringan*, vol. 5, no. 1, pp. 216–223, 2020, doi: 10.30743/infotekjar.v5i1.2892.

[6] M. Ignatoski, J. Lerga, L. Stanković, and M. Daković, "Comparison of entropy and dictionary based text compression in English, German, French, Italian, Czech, Hungarian, Finnish, and Croatian," *Mathematics*, vol. 8, no. 7, Jul. 2020, doi: 10.3390/MATH8071059.

[7] K. B. Adedeji, "Performance Evaluation of Data Compression Algorithms for IoT-Based Smart Water Network Management Applications," *Journal of Applied Science & Process Engineering*, vol. 7, no. 2, pp. 554–563, 2020.

[8] A. Gupta, A. Bansal, and V. Khanduja, *Modern Lossless Compression Techniques: Review, Comparison and Analysis*. 2017.

[9] A. Gupta and S. Nigam, "A Review on Different Types of Lossless Data Compression Techniques," *International Journal of Scientific Research in Computer Science, Engineering and Information Technology*, vol. 7, no. 1, pp. 50–56, Jan. 2021, doi: 10.32628/cseit217113.

[10] A. Gopinath and M. Ravisankar, "Comparison of Lossless Data Compression Techniques," in *Proceedings of the 5th International Conference on Inventive Computation Technologies, ICICT 2020*, Feb. 2020, pp. 628–633. doi: 10.1109/ICICT48043.2020.9112516.

[11] W. Semunigus and B. Pattanaik, "Analysis for Lossless Data Compression Algorithms for Low Bandwidth Networks," *J Phys Conf Ser*, vol. 1964, no. 4, pp. 1–5, Jul. 2021, doi: 10.1088/1742-6596/1964/4/042046.

[12] B. A. Krishna, N. Madhuri, and M. Malleswari, "Comparison and Implementation of Compression Algorithms in WSNs," *IJERT Journal International Journal of Engineering Research and Technology*, vol. 8, no. 7, pp. 1039–1042, 2019, [Online]. Available: www.ijert.org

[13] E. Prayoga and K. M. Suryaningrum, "IMPLEMENTASI ALGORITMA HUFFMAN DAN RUN LENGTH ENCODING PADA APLIKASI KOMPRESI BERBASIS WEB," 2018.

[14] A. P. U. Siahan, "IMPLEMENTASI TEKNIK KOMPRESI TEKS HUFFMAN," *Jurnal Informatika*, vol. 10, no. 2, pp. 1251–1261, 2016.

[15] G. Shrividhiya, K. S. Srujana, S. N. Kashyap, and C. Gururaj, "Robust data compression algorithm utilizing LZW framework based on huffman technique," in *2021 International Conference on Emerging Smart Computing and Informatics, ESCI 2021*, Mar. 2021, pp. 234–237. doi: 10.1109/ESCI50559.2021.9396785.

[16] H. N. Saad, F. mushtaq Jafar, and H. A. Salman, "A new compression technique in MANET: Compressed-LZW algorithm," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 16, no. 2, pp. 890–896, 2019, doi: 10.11591/ijeecs.v16.i2.pp890-896.

[17] R. Radescu, *Comparative Study of Performances in Lossless Data Compression for English and Romanian Text Files Using the Q-Coder*. 2018.