I.J. Image, Graphics and Signal Processing, **2022, 6, 59-69**
Published Online on December 8, 2022 by MECS Press (http://www.mecs-press.org/)
DOI: 10.5815/ijigsp.2022.06.05

A Comparative Analysis of Lossless Compression Algorithms on Uniformly Quantized Audio Signals

Sankalp Shukla*
Indira Gandhi Engineering College, Sagar, Madhya Pradesh, 470021, India
Email: sankalp2shukla@yahoo.in
ORCID iD: https://orcid.org/0000-0001-8734-0478
*Corresponding Author

Ritu Gupta
Indira Gandhi Engineering College, Sagar, Madhya Pradesh, 470021, India
Email: guptaritu725@gmail.com
ORCID iD: https://orcid.org/0000-0001-6351-4875

Dheeraj Singh Rajput
Indira Gandhi Engineering College, Sagar, Madhya Pradesh, 470021, India
Email: dheerajrajput22@gmail.com
ORCID iD: https://orcid.org/0000-0002-5608-1511

Yashwant Goswami
Rewa Engineering College, Rewa, Madhya Pradesh, 486002, India
Email: yashwantgoswami111@gmail.com
ORCID iD: https://orcid.org/0000-0002-4922-1264

Vikash Sharma
Rewa Engineering College, Rewa, Madhya Pradesh, 486002, India
Email: vikash900220@gmail.com
ORCID iD: https://orcid.org/0000-0001-6097-0189

Received: 27 March, 2022; Revised: 16 May, 2022; Accepted: 30 June, 2022; Published: 08 December, 2022

Abstract: This paper analyses the performance of various lossless compression algorithms employed on uniformly quantized audio signals. The purpose of this study is to enlighten a new way of audio signal compression using lossless compression algorithms. The audio signal is first transformed into text by employing uniform quantization with different step sizes. This text is then compressed using lossless compression algorithms which include Run length encoding (RLE), Huffman coding, Arithmetic coding and Lempel-Ziv-Welch (LZW) coding. The performance of various lossless compression algorithms is analyzed based on mainly four parameters, viz., compression ratio, signal-to-noise ratio (SNR), compression time and decompression time. The analysis of the aforementioned parameters has been carried out after uniformly quantizing the audio files using different step sizes. The study exhibits that the LZW coding can be a potential alternative to the MP3 lossy audio compression algorithm to compress audio signals effectively.

Index Terms: Audio Compression, Lossless Compression, Lempel-Ziv-Welch (LZW), Huffman, Arithmetic, Run Length Encoding (RLE), Uniform Quantization, Compression Ratio, Signal-to-Noise Ratio (SNR).

1. Introduction

From the last two decades, the world has witnessed a revolutionary advancement in technology. This advancement has resulted in a massive rise in the usage of internet. With increasing internet usage, the amount of data has been rising

at an alarming rate. The necessity of using data compression escalates further due to the fact that the data that is being generated is in digital form which is stored in the form of bytes of data and the multimedia data requires a large number of bytes for storage as well as transmission. Various compression algorithms help in fulfilling the desire for good quality data overcoming the bandwidth constraints and getting rid of the redundancy present in original data, thereby reducing the volume of memory required for storage and the bandwidth required for transmission of data. Compressing data not only reduces the volume of memory required but also helps in reduction of load on the input and output channels of communication systems.

The various compression algorithms can be broadly classified into two categories –lossless compression algorithms and lossy compression algorithms. In lossless compression algorithms, there is no loss of information, i.e., it is possible to exactly recover the original data from compressed data [1,2]. LZW coding, Run Length Encoding (RLE), Arithmetic Coding and Huffman Coding are some of the frequently used lossless compression algorithms. In lossy compression algorithms, there is some loss of information, i.e., it is not possible to exactly recover the original data from compressed data.

This paper proposes an alternative audio compression algorithm to address the bandwidth constraints while transmitting an audio signal through a channel. In the current scenario, MP3 is the most widely used audio compression algorithm. The performance of any compression algorithm can be measured in many ways, viz., compression time, decompression time, compression ratio and SNR [3,4,5]. SNR basically measures the closeness between the reconstructed data and original data. The current work compares a number of different compression algorithms to explore the possibility of a potential alternative to MP3 compression algorithm.

2. Methodology

This paper involves the use of lossless compression algorithms, viz., Run Length Encoding (RLE), LZW Coding, Arithmetic Coding and Huffman Coding to compress uniformly quantized audio signal data. The audio signal that is compressed is originally in .wav format. This audio signal data is first read and then uniformly quantized with different step sizes in order to convert it into text [6,7]. Four different step sizes that have been used in uniform quantization are 256, 512, 1024 and 2048.Each of these steps corresponds to an ASCII character so that the audio data is transformed into text consisting of a series of ASCII characters. Uniform quantization has been applied to extract the inherent redundancy present in the original audio signal data and eliminate the floating-point numbers. The elimination of floating-point numbers reduces both the memory storage space and the transmission bandwidth requirements. Once the original audio signal data is converted to text, the lossless compression algorithms are employed to eliminate the inherent redundancy.

The aforementioned algorithm follows a completely new approach to compress an audio signal in order to get better compression ratio than the presently used MP3 audio compression algorithm.

2.1. Run Length Encoding (RLE)

It is the simplest manifestation of a lossless compression algorithm. It works on the principle of conversion of sequence of symbols in succession to a number and the symbol, where the number denotes the count of the symbol in picture [8]. In a simplified manner, RLE converts a sequence of symbols in succession into two parts with the first part denoting a number gives the count of the symbol and the second part denoting the symbol itself. The number of times a particular symbol is encountered in succession is called run count [9].

The uniformly quantized audio signal data may contain large runs of ASCII characters which can be compressed by RLE. For example:

If RLE lossless compression algorithm is applied to the above text, it will be converted to 11W1B13W3B23W1B15W. Assuming that 1 byte of memory storage space is required to store one character or number, the original text will require a memory of 67 bytes whereas after compressing it using RLE, it will require a memory of 14 bytes. In this way, RLE compresses the data consisting of large runs.

2.2. Huffman Coding

This lossless coding algorithm generates codes which are known as Huffman codes. These Huffman codes are basically prefix codes. For a given set of probabilities in a model, these codes are optimal. The following two points can be established with reference to the optimum prefix codes generated using Huffman coding:

1. In these codes, the codewords have smaller code length for symbols that occur more often, i.e., symbols which are occurring with a larger probability than the symbols which are occurring with a smaller probability.
2. The codes generated using Huffman coding have equal code length for the two least often occurring symbols.

The first point ensures that the average bits per symbol required to denote the symbols is minimum. If the length of codeword for a symbol which occurs more frequently is larger than the length of codeword for a symbol which occurs less frequently, the average bits per symbol required to denote the symbols is larger than if the conditions are reversed.

As a result, an algorithm that allocates codewords of longer code lengths to more often occurring symbols cannot be termed as an optimal code algorithm [10,11].

The correctness of the second point can be observed by assuming the existence of an optimum code such that the codewords for the two symbols having least probabilities of occurrence are of unequal lengths with the difference in code length between the longest and shortest codewords being k bits. In the case of prefix code, it is not possible for the shorter codeword to be a prefix of longer codeword, so, even if the last k bits of the longer codeword are dropped, the longer and shorter codewords will be different with equal length. These two codewords belong to the symbols having least probability, hence, there cannot be a codeword which is longer than these codewords and the shortened codeword cannot be a prefix code of any other codeword. In addition, the elimination of k bits results in a codeword having a smaller average code length that the assumed code, thereby, demonstrating that the assumed code is not an optimal code. This proves the correctness of the second point.

Furthermore, the procedure of Huffman coding is completed by a final requirement apart from the above two observations, i.e., the codewords for the symbols with least probabilities of occurrence must be different only in the last bit.

Suppose there are 5 symbols a_0, a_1, a_2, a_3, a_4 with probabilities of 0.4, 0.2, 0.2, 0.1, 0.1 respectively. Then, the following steps can be used to get the Huffman codes for the given 5 symbols:

1. The symbols are arranged in the decreasing order of their probabilities.
2. The two symbols having least probabilities of occurrence are assigned binary ‘0’ and ‘1’.
3. The two least probabilities in each phase are added to get a new probability for the combination of the two least frequently occurring symbols.
4. The new probability along with the remaining probabilities is again placed in the decreasing order in the next phase.
5. The steps 2 to 4 are repeated until only two symbol probabilities are left. Both these probabilities are assigned binary ‘0’ and ‘1’.
6. To get the Huffman codes for a symbol, the assigned binary bits are read in reverse order for the symbol for which Huffman code is to be formulated.

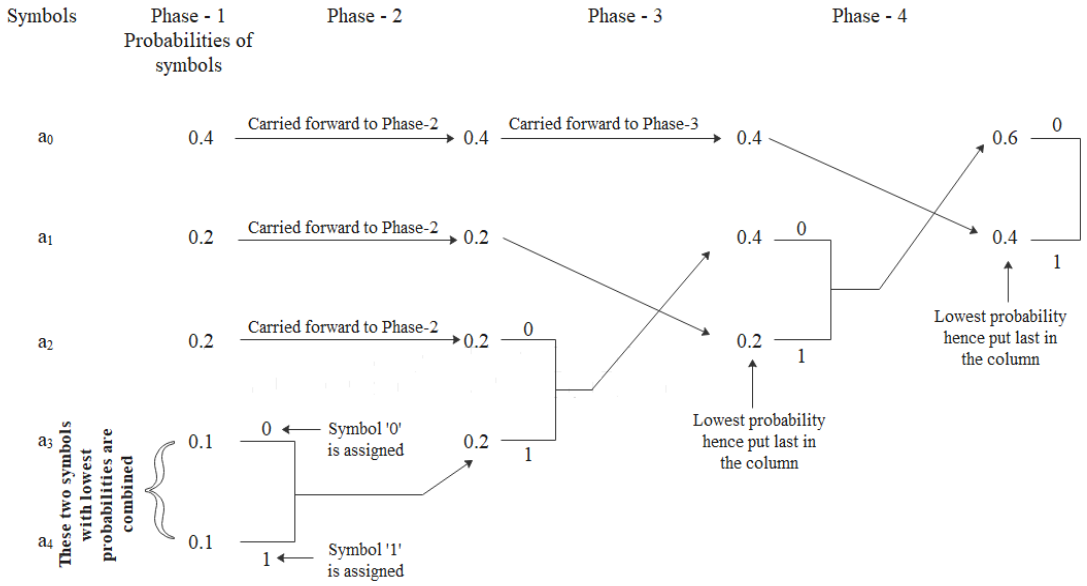

Fig. 1. Huffman code procedure.

Fig. 1 shows the steps to be followed to get the Huffman codes for corresponding symbols. Fig. 2 shows the procedure for formulation of Huffman code for a particular symbol. Table 1 shows the list of symbols along with their corresponding probabilities, their codewords and the lengths of codewords. The average codelength is always greater than or equal to the entropy for a particular set of symbols. The difference between the average codelength and entropy is a measure of the efficiency of Huffman coding [12].

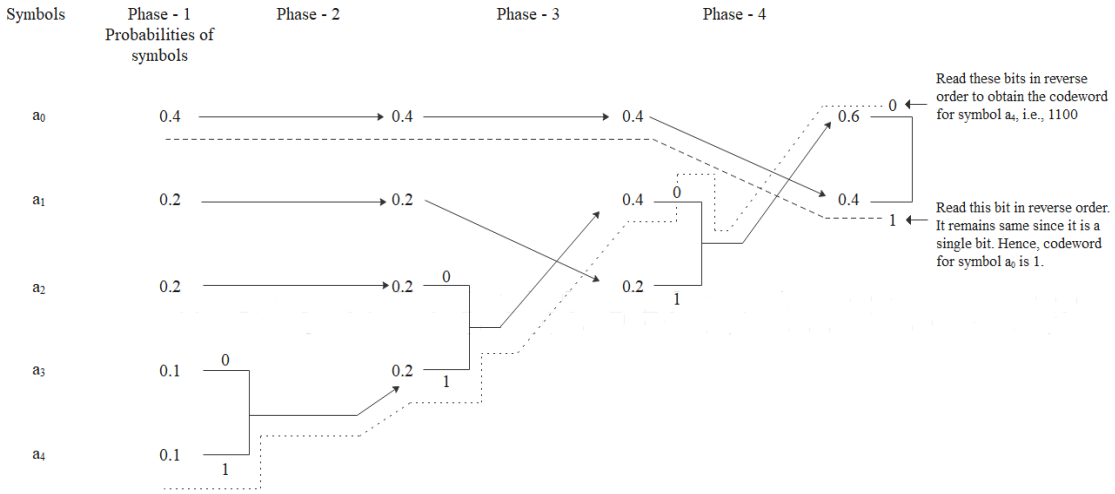

Fig. 2. Huffman code formulation for a particular symbol

Table 1. Huffman codes for the five symbols

Symbols	Probability	Codeword	Codelength
a_0	0.4	1	1
a_1	0.2	01	2
a_2	0.2	000	3
a_3	0.1	0010	4
a_4	0.1	0011	4

2.3. Arithmetic Coding

It is often considered inefficient to generate separate codewords for separate symbols present in a sequence. Instead, it is preferable to group a sequence of symbols and then generate codewords corresponding to each such group. But it is impractical to generate Huffman codes for such groups of symbols. In Huffman coding, it becomes necessary to generate codewords for all possible groups of symbols of a particular length being encoded resulting in a drastic increase in the size of codebook. The arithmetic coding generates codewords for a group of symbols without the need to assign codewords to all the possible groups of symbols of same length.

Arithmetic coding generates a tag or a unique identifier for the group of symbols being encoded [13]. The tag that is generated is assigned a unique binary code. In general, tag generation and binary code assignment correspond to the same process. However, this process is often divided into two phases to make it simpler. The first phase corresponds to tag generation for a specific group of symbols and the second phase involves the assignment of a unique binary code to the tag generated.

The interval [0,1) is chosen as one of the suitable collection of tags to represent groups of symbols. The reason behind the selection of this interval is that it contains infinite numbers which can be used as tags, thereby, allowing the assignment of a unique tag to each group of symbols. The cumulative distribution function (cdf) which is linked with the source is used for mapping groups of random variables into the unit interval.

The encoding procedure involves the following steps:

1. The initial interval [0,1) is divided into a number of parts. This number depends on the number of different symbols present in the group of symbols being encoded. The division of the initial interval is based on the probabilities of the symbols in the group.
2. The sub-interval corresponding to the symbol being encountered is sub divided into the same number of parts and based on the same criteria as the initial interval.
3. The step 2 is repeated until the last symbol in the group of symbols to be encoded is encountered.
4. The lower limit of the sub-interval for the last symbol constitutes the tag or unique identifier.

Fig. 3 shows the encoding procedure for a group of four symbols a_0, a_1, a_2, a_3 with probabilities of 0.4, 0.3, 0.1, 0.2 respectively. The procedure assumes that group of symbols being encoded are $a_0a_1a_3a_3a_2$.

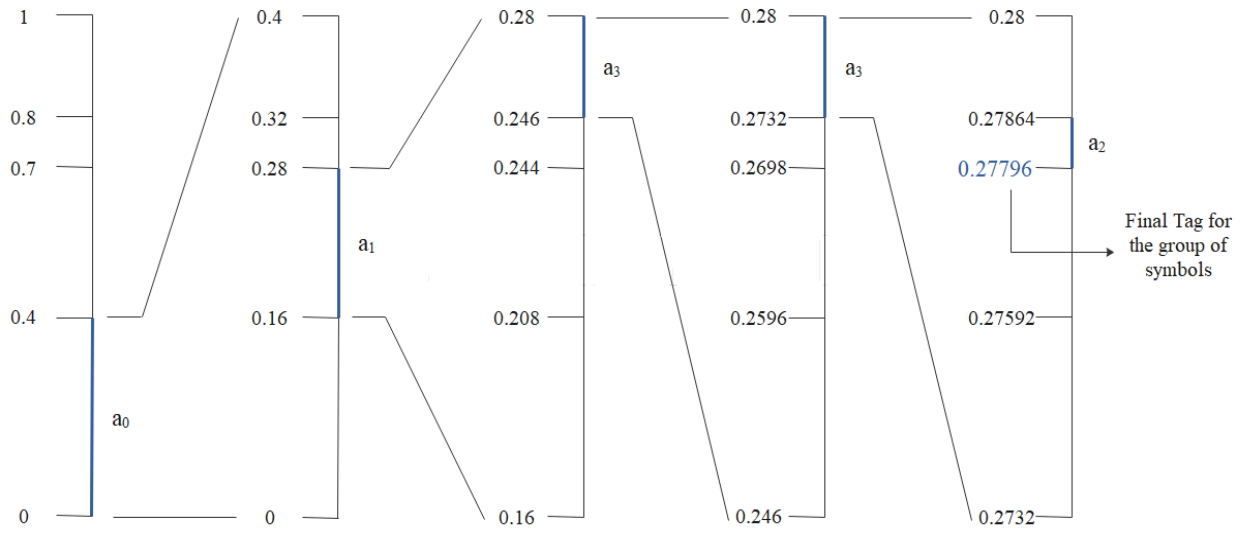

Fig. 3. Arithmetic encoding procedure

The decoding procedure involves the following steps:

1. The initial interval [0,1) is divided into a number of parts. This number depends on the number of different symbols present in the group of symbols being encoded. The division of the initial interval is based on the probabilities of the symbols in the group.
2. The sub-interval to which the tag belongs is determined.
3. The symbol corresponding to the above established sub-interval is noted down.
4. The established sub-interval is sub divided into the same number of parts and based on the same criteria as the initial interval.
5. The steps 2, 3 and 4 are repeated until the last symbol to be decoded is encountered.

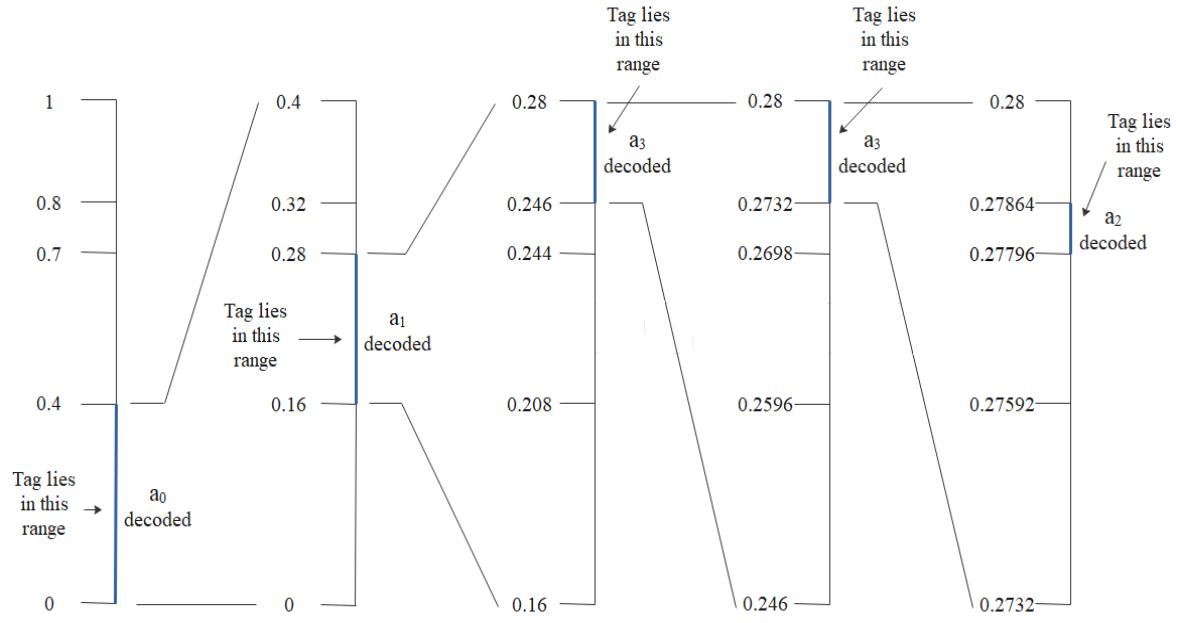

Fig. 4. Arithmetic decoding procedure

Fig. 4 shows the decoding procedure for the group of symbols encoded in the encoding example.

2.4. LZW Coding

LZW is a lossless dictionary compression technique. It is an enhanced version of LZ78 compression algorithm. This algorithm uses a dictionary for compression [14,15,16]. The dictionary being used initially consists of a certain number of ASCII symbols. In this algorithm, the compression is obtained by replacing a group of symbols with the indices of the dictionary. The dictionary is continuously modified based on the symbols to be encoded. The most

prominent feature of LZW coding is that apriori information about the probability of occurrence of the symbols to be encoded is not essential making it one of the most widely used compression algorithm in image compression [17]. The LZW encoding procedure involves the following steps:

1. The dictionary is first initialized by a particular number of ASCII symbols.
2. The first symbol is then read and its presence in the dictionary is checked.
3. If the symbol read is found in the dictionary, then it is grouped with the next symbol and presence of the group of symbols is checked in the dictionary. The symbols are grouped till the group of symbols are found in the dictionary.
4. If a group of symbols is not found in the dictionary, it is added to the next index of the dictionary and the index of previously found symbol or group of symbols is considered as a codeword for that symbol or group of symbols.
5. The last symbol whose grouping with the rest of the symbols which led to the group of symbols not being found in the dictionary is considered as the next symbol.
6. The steps 2 to 5 are repeated till all the symbols or group of symbols are encoded.

The LZW decoding procedure involves the following steps:

1. The dictionary is first initialized in a similar manner as in the encoding algorithm.
2. The first index is read. The symbol or group of symbols at that index in the dictionary is stored in the output.
3. The first symbol from above read symbol or group of symbols is stored separately.
4. The next index is read. If the length of present dictionary is less than the index read, then the previously read symbol or group of symbols is grouped with the symbol stored in step 3.
5. If the length of present dictionary is greater than the index read, the symbol or group of symbols is read from the dictionary at the recently read index and is stored separately.
6. Based on whether the length of dictionary is lesser or greater than the index read, the first symbol from the grouped symbols in step 4 or the separately stored symbol in step 5 is stored separately while combining the group of symbols obtained either in step 4 or step 5 with the output.
7. The separately stored symbol in step 6 is combined with the symbol or group of symbols present in the dictionary at the previously read index and the result is stored as a new entry in the dictionary.
8. The steps 4 to 7 are repeated until all the indexes are decoded to get the uncompressed output.

3. Result Analysis and Discussion

The analysis of various lossless compression algorithms on uniformly quantized audio signals was carried out in MATLAB R2019a. There are a number of parameters based on which the performance of various lossless compression algorithms can be analyzed depending on a particular application. Some of them that have been used to measure the performance of various lossless compression algorithms on uniformly quantized audio signals are – compression ratio, compression time, decompression time and signal-to-noise ratio (SNR).

Compression ratio is defined as the ratio of the size of audio file before compression to the size of audio file after compression, i.e.

$$\text{Compression Ratio} = \frac{\text{Size of audio file before compression}}{\text{Size of audio file after compression}} \tag{1}$$

Compression time is the time taken by the algorithm to compress the original audio file while the decompression time is the time taken by the algorithm to decompress the compressed audio file. In the present analysis, the compression time includes the time taken by the algorithm to quantize the original audio file.

Signal-to-noise ratio is the ratio of signal power to noise power. The difference between various samples of original audio file and decompressed audio file corresponds to noise in the present analysis, i.e., the noise is basically due to quantization error. The signal-to-noise ratio is calculated in dB (decibel) by using the formula given below:

$$\text{SNR (in dB)} = 10\log_{10}(\text{SNR}) \tag{2}$$

Table 2. Analysis results based on various parameters for file with size of 99.2 KB

Compression Technique	Step Size	SNR (in dB)	Compression Ratio	Compression Time (in sec)	Decompression Time (in sec)
RLE	256	84.85	0.21	0.33	0.28
	512	98.69	0.17	0.36	0.29
	1024	112.53	0.15	0.38	0.30
	2048	126.38	0.14	0.42	0.30
Arithmetic Coding	256	84.85	0.05	1.62	2.21
	512	98.69	0.04	1.90	2.20
	1024	112.53	0.04	2.15	2.52
	2048	126.38	0.03	2.54	2.97
Huffman Coding	256	84.85	0.05	0.57	1.49
	512	98.69	0.04	0.89	1.89
	1024	112.53	0.04	1.10	2.85
	2048	126.38	0.03	2.15	3.75
LZW Coding	256	84.85	0.75	3.53	0.37
	512	98.69	0.57	4.30	0.42
	1024	112.53	0.46	5.35	0.45
	2048	126.38	0.40	7.15	0.50

Table 3. Analysis results based on various parameters for file with size of 519 KB

Compression Technique	Step Size	SNR (in dB)	Compression Ratio	Compression Time (in sec)	Decompression Time (in sec)
RLE	256	85.00	0.43	0.55	0.47
	512	99.01	0.35	0.61	0.51
	1024	112.80	0.31	0.69	0.52
	2048	126.92	0.29	0.85	0.53
Arithmetic Coding	256	85.00	0.10	4.01	4.96
	512	99.01	0.08	4.56	5.54
	1024	112.80	0.07	5.22	6.30
	2048	126.92	0.06	5.84	7.24
Huffman Coding	256	85.00	0.10	0.89	3.53
	512	99.01	0.08	1.30	5.02
	1024	112.80	0.07	2.04	6.17
	2048	126.92	0.06	4.15	9.49
LZW Coding	256	85.00	2.04	15.27	0.72
	512	99.01	1.52	22.62	0.85
	1024.	112.80	1.16	34.36	1.00
	2048	126.92	0.93	43.64	1.19

Table 4. Analysis results based on various parameters for file with size of 919 KB

Compression Technique	Step Size	SNR (in dB)	Compression Ratio	Compression Time (in sec)	Decompression Time (in sec)
RLE	256	98.13	0.36	0.90	0.79
	512	111.24	0.31	1.03	0.83
	1024	126.03	0.29	1.26	0.89
	2048	139.75	0.27	1.71	0.92
Arithmetic Coding	256	98.13	0.08	7.79	9.36
	512	111.24	0.07	9.23	10.84
	1024	126.03	0.06	9.92	12.89
	2048	139.75	0.06	11.75	18.26
Huffman Coding	256	98.13	0.08	1.45	7.43
	512	111.24	0.07	1.80	8.55
	1024	126.03	0.06	3.64	17.37
	2048	139.75	0.06	6.22	18.18
LZW Coding	256	98.13	1.69	57.78	1.66
	512	111.24	1.28	88.65	2.01
	1024	126.03	1.01	133.84	2.33
	2048	139.75	0.83	179.01	2.68

The above analysis was done on three .wav type audio files of sizes 99.2 KB, 519 KB and 919 KB respectively on Intel(R) Core(TM) i7-8565U CPU @ 1.80GHz - 1.99 GHz processor with 8 GB RAM. Table 2, Table 3 and Table 4 show a comparison of the results based on various parameters, viz, SNR, compression ratio, compression time and decompression time obtained for the three files. The tables compare the parameters for different step sizes used in the uniform quantization of audio files. It is clearly visible that the SNR does not change with the compression algorithm being used. The values in the above 3 tables have been used to plot the graphs shown in Fig. 6. It is evident from the

values in these tables that LZW algorithm provides the best compression ratio which is a key to deal with the bandwidth constraints involved while transmitting an audio signal through a channel.

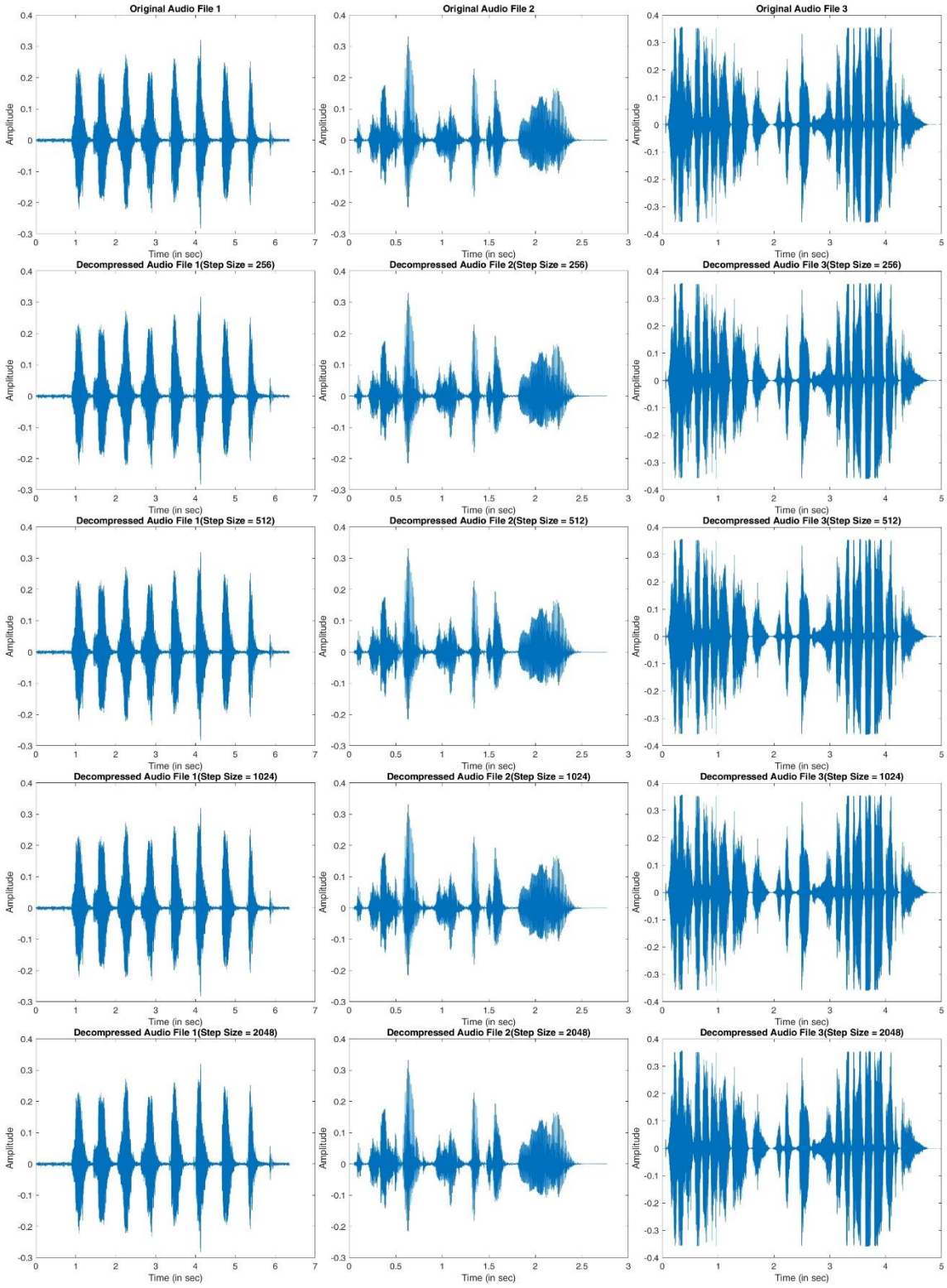

Fig. 5. Comparison between original and decompressed audio file waveforms for different step sizes.

Fig. 5 shows the comparison between original and decompressed audio file waveforms for different step sizes used in quantizing audio signals. The file sizes of original audio file 1, original audio 2, original audio file 3 are 99.2 KB, 519 KB, 919 KB respectively. It can be observed from the above figure that the difference between the waveforms of original audio files and their decompressed versions is negligible. This is due to the fact that lossless compression algorithms have been used for compressing audio files. This is a major advantage of the proposed compression algorithm.

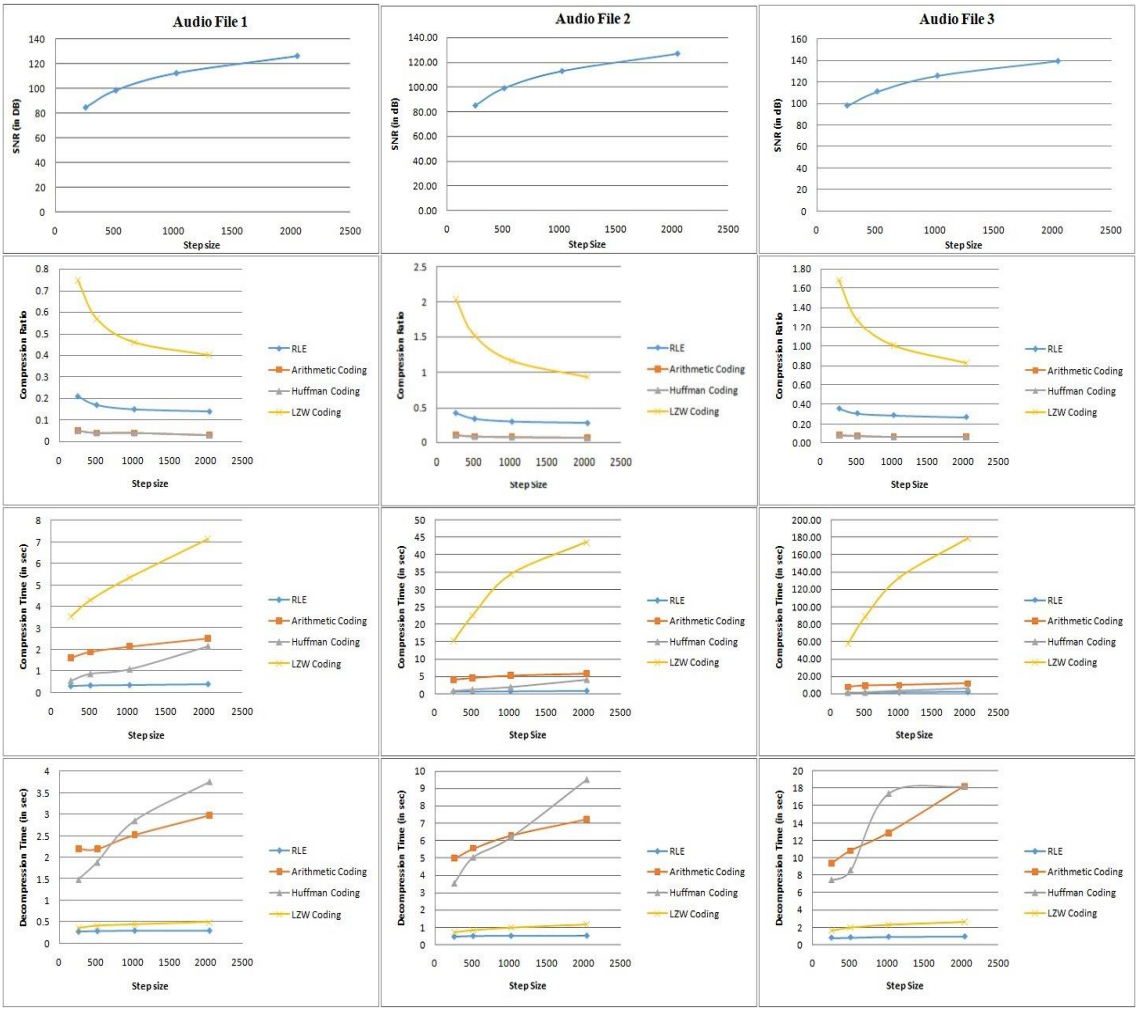

Fig. 6. Graphical comparison of various parameters on application of different lossless compression algorithms on quantized audio files of different sizes

Fig. 6 shows graphical comparison of various parameters when different lossless compression algorithms are applied to quantized audio files of three different sizes. The analysis can be done based on two different factors, viz, size of the original file to be compressed and step size used for quantization.

The graphs in Fig. 6 have been plotted with SNR/Compression ratio/Compression time/Decompression time on the y-axis and step size used in uniform quantization of audio signals on the x-axis based on the values in Table 2, Table 3 and Table 4.

The graphs in the first row have been plotted based on the values in the third column of each table, i.e., SNR (in dB). It can be observed that the SNR is independent of the compression technique being because all the techniques are lossless in nature. Due to the aforementioned fact, the graphs in first row of Fig. 6 contain only single curves indicating the SNR due to all the compression techniques.

The graphs in the second row have been plotted based on the values in the fourth column of each table, i.e., Compression Ratio. Similarly, the graphs in the third and fourth rows have been plotted based on the values in the fifth and sixth columns of each table respectively.

3.1. Analysis based on size of the original file to be compressed

Analysis based on size of the original file to be compressed can be done by comparing the graphs in Fig. 6 in a horizontal manner. As we move from left to right analyzing graphs in each row of Fig. 6, the size of the files is increasing.

Analysis of the graphs in the first row of Fig. 6 shows that SNR is independent of the compression algorithm used on a particular file as there is no loss of information due to compression but SNR depends on the size of the file being compressed. As the size of the file to be compressed increases, the SNR due to all the compression algorithms increases.

Analysis of the graphs in the second row of Fig. 6 shows that the LZW coding algorithm when applied to the quantized audio files results in the best compression ratio followed by RLE algorithm. Huffman coding and arithmetic coding result in almost equal and worst compression ratios for the three files. Apart from LZW coding algorithm, the other three algorithms result in a file which requires more storage space than the original file. As the size of the file increases, the compression ratio first increases and then decreases a little. This is true for all the compression algorithms.

Analysis of the graphs in the third row of Fig. 6 shows that the LZW algorithm requires the highest amount of time for compressing the quantized audio files followed by arithmetic coding and Huffman coding. The RLE coding requires least compression time. The compression time increases with increase in the file size for all the compression algorithms.

Analysis of the graphs in the fourth row of Fig. 6 shows that the RLE coding also requires least decompression time followed by LZW coding. The decompression time required in case of arithmetic coding and Huffman coding is higher. The decompression time increases with increase in the file size for all the compression algorithms.

3.2. Analysis based on step size used for quantization

Analysis shows that SNR increases with increase in step size for all the compression algorithms.

The LZW coding algorithm when applied to the quantized audio files results in the best compression ratio followed by RLE algorithm, Huffman coding and arithmetic coding. Only the LZW coding results in compression of a file with size more than 250 KB. For a file with size less than 250 KB, even LZW coding results in expansion rather than compression. As the step size increases, the compression ratio decreases, i.e., the compression ratio is maximum for a step size of 256 and minimum for a step size of 2048. This is true for all the compression algorithms.

The LZW algorithm requires the highest amount of time for compressing the quantized audio files followed by arithmetic coding and Huffman coding. The RLE coding requires least compression time. The compression time increases with increase in step size for all the compression algorithms.

The RLE coding also requires least decompression time followed by LZW coding. The decompression time required in case of arithmetic coding and Huffman coding is higher. The decompression time increases with increase in step size for all the compression algorithms. As the step size increases, the decompression time required for arithmetic coding becomes lesser than the decompression time required for Huffman coding.

4. Conclusions

LZW coding gives the best compression among the four lossless compression algorithms but the time required for compression is higher than RLE coding. When the file is to be used by a user, the decompression time required by LZW coding is almost same as that required by RLE coding. Considering the above facts, it can be concluded that LZW coding is the best lossless compression algorithm for compressing audio files of large sizes. LZW coding can also be combined with DWT (Discrete Wavelet Transform) or DCT (Discrete Cosine Transform) or MDCT (Modified DCT) to get better compression ratios, thus becoming a potential alternative to MP3 audio compression algorithm. This alternate algorithm can be used to compress audio signals with higher compression ratio than the MP3 compression algorithm, thereby, overcoming the bandwidth constraints encountered while transmitting such audio signals through a communication channel.

References

[1] Anup, Anshul & Ashok, Ravi &Raundale, Pooja. (2019). Comparative Study of Data Compression Techniques. International Journal of Computer Applications. 178. 15-19. 10.5120/ijca2019919104.

[2] A. Gopinath and M. Ravisankar, "Comparison of Lossless Data Compression Techniques," 2020 International Conference on Inventive Computation Technologies (ICICT), 2020, pp. 628-633, doi: 10.1109/ICICT48043.2020.9112516.

[3] T. Hidayat, M. H. Zakaria and A. N. C. Pee, "Survey of Performance Measurement Indicators for Lossless Compression Technique based on the Objectives," 2020 3rd International Conference on Information and Communications Technology (ICOIACT), 2020, pp. 170-175, doi: 10.1109/ICOIACT50329.2020.9332044.

[4] Sheraj, Mohammad & Chopra, Ashish. (2020). Data Compression Algorithm for Audio and Image using Feature Extraction. 1-6. 10.1109/ICCCSP49186.2020.9315248.

[5] Uthayakumar Jayasankar, Vengattaraman Thirumal, Dhavachelvan Ponnurangam, "A survey on data compression techniques: From the perspective of data quality, coding schemes, data type and applications," Journal of King Saud University - Computer and Information Sciences, Volume 33, Issue 2, 2021, Pages 119-140, ISSN 1319-1578, https://doi.org/10.1016/j.jksuci.2018.05.006.

[6] Ashida, S. &Kakemizu, H. & Nagahara, Masaaki & Yamamoto, Yutaka. (2004). Sampled-data audio signal compression with Huffman coding. 972 - 976 vol. 2.

[7] S. Shukla, M. Ahirwar, R. Gupta, S. Jain and D. S. Rajput, "Audio Compression Algorithm using Discrete Cosine Transform (DCT) and Lempel-Ziv-Welch (LZW) Encoding Method," 2019 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COMITCon), 2019, pp. 476-480, doi: 10.1109/COMITCon.2019.8862228.

[8] Dudhagara, Chetan & Patel, Hasamukh. (2017). Performance Analysis of Data Compression Using Lossless Run Length Encoding. Oriental journal of computer science and technology. 10. 703-707. 10.13005/ojcst/10.03.22.

[9] Khalid Sayood. *Introduction to Data Compression*. Elsevier Inc., San Francisco, CA, 2006.

[10] N. Dhawale, "Implementation of Huffman algorithm and study for optimization," 2014 International Conference on Advances in Communication and Computing Technologies (ICACACT 2014), 2014, pp. 1-6, doi: 10.1109/EIC.2015.7230711.

[11] Gaurav Kumar, Er. Sukhreet Singh Brar, Rajeev Kumar, Ashok Kumar, "A Review: DWT-DCT Technique and Arithmetic-Huffman Coding based Image Compression", IJEM, vol.5, no.3, pp.20- 33, 2015. DOI: 10.5815/ijem.2015.03.03.

[12] Gajendra Sharma, "Analysis of Huffman Coding and Lempel–Ziv–Welch (LZW) Coding as Data Compression Techniques," International Journal of Scientific Research in Computer Science and Engineering, Vol.8, Issue.1, pp.37-44, 2020.

[13] Witten, Ian & H, Ian & Neal, & M, Radford & Cleary, & G, John. (1987). Arithmetic Coding for Data Compression. Communications of the ACM. 30. 520-. 10.1007/978-0-387-30162-4_34.
[14] J. Ziv and A. Lempel, "A universal algorithm for sequential data compression," in IEEE Transactions on Information Theory, vol. 23, no. 3, pp. 337-343, May 1977, doi: 10.1109/TIT.1977.1055714.
[15] J. Ziv and A. Lempel, "Compression of individual sequences via variable-rate coding," in IEEE Transactions on Information Theory, vol. 24, no. 5, pp. 530-536, September 1978, doi: 10.1109/TIT.1978.1055934.
[16] Welch, "A Technique for High-Performance Data Compression," in Computer, vol. 17, no. 6, pp. 8-19, June 1984, doi: 10.1109/MC.1984.1659158.
[17] M. Alsenwi, T. Ismail and H. Mostafa, "Performance analysis of hybrid lossy/lossless compression techniques for EEG data," *2016 28th International Conference on Microelectronics (ICM)*, 2016, pp. 1-4, doi: 10.1109/ICM.2016.7847849.

Authors' Profiles

Sankalp Shukla completed his Bachelors of Engineering in Electronics & Communication Engineering from GGITS, Jabalpur and his Masters in Communication Engineering from NITK Surathkal. He is currently pursuing working as an Assistant Professor in Department of Electronics & Communication Engineering at Indira Gandhi Government Engineering College, Sagar (M.P.). His research interests include Audio and Image Signal Processing.

Ritu Gupta completed her Bachelors of Engineering in Electronics & Communication Engineering from LNCT, Bhopal and her Masters in VLSI & Embedded Systems from MANIT Bhopal. She is currently pursuing working as an Assistant Professor in Department of Electronics & Communication Engineering at Indira Gandhi Government Engineering College, Sagar (M.P.). Her research interests include Audio and Image Signal Processing, Low Power VLSI Design.

Dheeraj Singh Rajput completed his Bachelors of Engineering in Electronics & Communication Engineering from MITS, Gwalior and his Masters in VLSI & Embedded Systems from DTU, Delhi. He is currently pursuing working as an Assistant Professor in Department of Electronics & Communication Engineering at Indira Gandhi Government Engineering College, Sagar (M.P.). His research interests include Low Power VLSI Design.

Yashwant Goswami completed his Bachelors of Engineering in Electronics & Communication Engineering from GGCT, Jabalpur and his Masters in Nanotechnology from MANIT, Bhopal. He is currently pursuing working as an Assistant Professor in Department of Electronics & Communication Engineering at Rewa Government Engineering College, Rewa (M.P.). His research interests include Nanotechnology, Low Power VLSI Design.

Vikash Sharma completed his Bachelors of Engineering in Electronics & Communication Engineering from MITS, Gwalior and his Masters in VLSI & Embedded Systems from IIIT, Gwalior. He is currently working as an Assistant Professor in Department of Electronics & Communication Engineering at Rewa Government Engineering College, Rewa (M.P.). His research interests include Low Power VLSI Design.

How to cite this paper: Sankalp Shukla, Ritu Gupta, Dheeraj Singh Rajput, Yashwant Goswami, Vikash Sharma, "A Comparative Analysis of Lossless Compression Algorithms on Uniformly Quantized Audio Signals", International Journal of Image, Graphics and Signal Processing(IJIGSP), Vol.14, No.6, pp. 59-69, 2022. DOI:10.5815/ijigsp.2022.06.05