ISBN: 978-1-948012-17-1

Ethics and Information Technology (ETIT)

DOI: <http://doi.org/10.26480/etit.02.2020.40.44>

CrossMark

EVALUATION OF LOSSLESS ALGORITHMS FOR DATA COMPRESSION

Neha Sharma^a, Usha Batra^b^aResearch Scholar, G D Goenka University, Gurugram, 122103, India^bAssistant Dean, G D Goenka University, Gurugram, 122103, India*Corresponding Author Email: nehasharma0110@gmail.com

This is an open access article distributed under the Creative Commons Attribution License CC BY 4.0, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

ARTICLE DETAILS

Article History:

Received 25 October 2020

Accepted 26 November 2020

Available online 03 December 2020

ABSTRACT

Nowadays the communication and exchange of information over the internet which includes sending e-mails, text messages via online apps e.g. messengers, has become the need of the hour. While transmitting the data, some of the critical aspects like size of message or file, need to deal with extreme precaution as these are very crucial. Furthermore, the transmitting time is directly proportional to the file size i.e. lesser file size always take less time. Compression techniques are used to decrease the size of the file, meanwhile not impacting the quality. This paper demonstrates that the use of two lossless compression techniques on images so that they become suitable for information security using the techniques like steganography, cryptography etc. Thus, the objective of the paper is to reduce the image size by using Huffman encoding and run-length encoding algorithms. The algorithms are implemented and the performances are analysed by evaluating the results on different parameters, such as compression ratio, compressed file size, and compression & decompression time. The paper is concluded with the analysis of the results obtained. © 2012 Published by Elsevier Ltd. Selection and/or peer-review under responsibility of Global Science and Technology Forum Pte Ltd.

KEYWORDS

Compressed File size, Compression Ratio, Compression Time, Decompression Time.

1. INTRODUCTION

Data compression is a procedure undertaken to convert the depicted data from one form to another form post compression, which contains the same information but with the reduced size (Patil, Kulat, 2017). The main benefits of compressing the data are the transmission bandwidth and the reduced storage capacity. This is very useful because storing and transmitting the files requires huge resources. One of the major area of application for the data compression techniques is on digital images. There are a number applications for image processing, such as medical imaging, satellite imaging, etc where the image size is large and it requires a more storage capacity or high bandwidth for transmitting it in its original form over a communication channel. After compression, when the size of data is reduced, it gives us the leverage to send more data. Therefore reducing the size to the half is equal to doubling its storage capacity.

After getting this extra storage space, we can store data hieratically at better and higher levels that also avoids extra loads on input/output devices of computer system.

2. LITERATURE REVIEW

RLE compression algorithm has been used widely because of its simplicity and less complexity. The algorithm is easy to implement and some of the researchers has made many modifications in it such as laying stress on the way the pixels are scanned like either the pixels are to scanned in row fashion, or column order (Karthikeyan, 2014). Some other authors have focused on calculating the bit depth of the runs of repeated pixels and tried to used enhance entropy coding to enhance (Suarjaya, 2012). The authors (Albahadily and Tsviatkou, 2016) have modified run length encoding

algorithm in order to achieve reduced encoded size and reduced encoding time. The authors (Canard et. al., 2017) have used the run length algorithm in order to make it fit within the constraints of FHE execution and then analyzing it to the achieve the optimized FHE execution efficiency. Hybrid DWT-DCT algorithm is applied (Rafea and Salman, 2018) in order to compress the medical image and an adaptive RLE algorithm is used to encode the runs of zero created by the hybrid algorithm in order to achieve better compression ratio results. The authors (Khasawneh and Alshorman, 2020) have used Frei-chen bases technique for compressing the large image data and then used a modified RLE algorithm in order to enhance the compression factor with out adding any distortion so as to receive the high-quality decompressed image. The use of Huffman encoding algorithm also brings the high-quality compression results. The authors (Erdal and Erguzen, 2019) have used Huffman encoding and arithmetic encoding in order to provide a solution to long bit sequences. The Huffman encoding in general gives better compression results (Ajala et al, 2018). The authors have combined to Huffman encoding with LZW algorithm in order to achieve cheap, reliable and efficient system.

3. DATA COMPRESSION

There are two types of compression categories: Lossy Compression and Lossless Compression.

3.1 Lossy Compression

Lossy compression technique is one that ignores the less important data. The file compressed using lossy technique will not be exactly same as the original file. After decompression we get the closer approximation of the original file (Klein et. al., 2019). Lossy type compression technique shrinks the bits by finding and eliminating unnecessary information

Quick Response Code	This paper was presented at	Access this article online	
	International Conference on Contemporary Issues in Computing (ICIC-2020) - Virtual IETE Sector V, Salt Lake, Kolkata From 25th-26th July 2020	Website: www.intelcomp-design.com	DOI: 10.26480/etit.02.2020.40.44

(Uthayakumar et.al, 2018). This mechanism of shrinking the file size of the data required to send is termed as the data-compression. Its formal name is source-coding which means that coding is done prior sending (Thanki and Kothari, 2019)

3.2 Lossless Compression

As the name says about itself, Lossless compression techniques restructure the original data without losing any data. The file compressed using lossless technique will be same as the original file. After decompression we get the exact replica of the original file. Lossless compression is categorized in following three categories (Sitaridi, 2016):

Figure 1: Categories of Lossless compression Techniques

• **Entropy based encoding:** Here, the algorithm first calculates the number count of the unique symbol's occurrence in the document. Then the symbol is replaced by another symbol which is generated through an algorithm. The symbols which are generated by algorithms are static/fixed for some symbols of the original document. They are not content dependent. The dimension of symbols obtained through algorithms are not fixed, it varies according to the symbol's frequency in the document. The algorithms that fall under this category are (): Arithmetic coding, Huffman encoding, Shanon etc.

• **Dictionary based encoding:** Often addressed as substitution encoding. Here the data structure is maintained by the encoder as 'Dictionary' which is just a collection of string (Setyaningsih, and Wardoyo, 2017). The substring is chosen from the original source and is then compared to the strings in dictionary. If there is a positive match, then the dictionary selected substring replaces the original substring in the programmed file. The algorithms that fall under this category are Byte pair encoding, DEFLATE, Lempel-Ziv, etc.

• **Others:** All other algorithms fall here i.e., RLE, BWT, CTW, delta, MTF, PAQ, PPM, etc.

4. MEASUREMENT PARAMETERS

4.1 Compression Ratio

Defined as a ratio between the file size after compression & Pre-compression file size.

$$\text{Compression ratio} = \frac{\text{file size after compression}}{\text{pre-compression file size}} \quad (1)$$

4.2 Compression Time

It is the time taken by the algorithm to compress the file.

4.3 Decompression Time

It is the time taken by the algorithm to decompress the file and retrieve the original file from the compressed one.

4.4 Compressed File size

It is the size that is measured in bytes after the image is being compressed.

5. COMPRESSION ALGORITHMS

There are various kinds of compression algorithms. In this paper, the results of RLEs algorithm and Huffman Encoding Algorithm are analysed.

5.1 Run Length Encoding (RLE)

The algorithms identifies the symbols as runs and non runs. The runs is generally the consecutive sequence of symbols while rest are marked as non runs. Occurrences of redundancies are also dealt by this algorithm (Sharma, and Bollavarapu, 2015). Any sort of symbols which are being repeated, frequency of its repetition and their length is checked in the process. Consecutive recurrent symbols are identified and are marked as runs while remaining arrangements are marked as non-runs. E.g. the text

"XYXYYYYYZ" is acknowledged as compressing source, front 3 letters are acknowledged to be a non-run having 3 length, and succeeding 4 letters are acknowledged to be run with 4 length as Y is repeating. There are 2 main tasks performed by this algorithm (i) identifying runs of the source file (ii) recording symbol & length of every run. These algorithms uses those runs 1) to shrink original source file 2) Symbol & length recording for each run. These algorithms also uses these runs for the compression of the original file meanwhile separating all the non-runs from the compression process. The objective is to change consecutive repetitive occurrences of symbols' followed by the number of occurrences. The efficiency of the algorithm can be improved if the data uses only two symbols (e.g. 0 and 1) in its bit pattern and frequency of one symbol should be more than the other (Souley et al, 2014).

Figure 2: Run-length encoding for two symbols (Souley et. al, 2014)

In this algorithm, repeated pixel can be substituted by a number that indicate which pixel value is repeated and how many times it is repeated (Singh and Devgon, 2019). The occurrence of 0 and 1 in sequence counts. The image file format like TIFF, BMP, etc. mostly use this algorithm for image compression (Hussaina et.al, 2018).

Algorithm:

Step1: Reading the character from input string.

Step2: Storing of the repeated character.

Step3: If the character isn't repetitive, mark the current character.

Step4: If current character match with previous character fails, mark previous character as current character and go to step 1.

Step5: Those characters are encoded as 2 bytes.

Step 6: If the character is null then exit.

5.2 Huffman encoding

This technique uses varying length code for the encoding of source symbol (i.e. characters). The varying-length code table generation is built on the projected probability of how many times each value of source symbol can possibly occur. Using this method, table is formed by merging the occurrences of symbols individually (Pal et.al, 2018). This table is also known as frequency table.

After a certain ordering of this table a tree is created. In this tree higher frequency symbols are assigned with lesser bit codes and less frequent symbols are assigned higher bit codes. After such arrangements of codes a code table is generated (Büyükoçak, 2019).

Let us consider the following example, there is a text file which uses 5 characters (A, B, C, D, and E). Before assigning bit patterns, each character is assigned some weight based on its frequency of use. Assume frequency of characters as shown in Fig 3.

Figure 3: Huffman Coding (Souley et. al, 2014)

Commencing from root as shown in Figure 4, a character code is found by pursuing the branches leading to character which is the bit value of each branch taken sequentially (Souley et. al, 2014).

Figure 4: Reduced Huffman Coding (Souley et. al, 2014)

Huffman encoding algorithms use the source alphabet's probability distribution for code word symbol development. To calculate this, initially probability distribution of all source character's frequency distribution is calculated. According to these calculated probabilities, the code words are then assigned (Khaitu and Panda, 2018). Generally the pattern of assigning is that the shorter code word is assigned for higher probabilities and a longer code word is assigned for a smaller probabilities. Now Binary tree is constructed with symbols as leaves (according to the respective probabilities) and paths as code words (Thanki et.al, 2019). There are two subs of Huffman encoding (i) static Huffman algorithms (ii) Adaptive

Huffman Algorithms. In static Huffman algorithms the frequencies are calculated initially while common tree for compression and decompression is postliminary created (Narmatha et.al, 2017). It is very important to transfer these tree details or save these tree details along the compressed file. In adaptive algorithm (Dath and Panicker, 2017), the tree is developed while the process of calculating the frequencies is going on. That's why two trees are spawned subsequently during processes. Using this, a flag symbolized tree is generated initially and later on in the process, updated to next symbol.

Algorithm:

Step1: Analysis of the inputs and frequency of each symbol.

Step 2: Calculate each symbolic occurrence probability with symbol count.

Step3: Sequential sorting of symbol based on occurrence probability.

Step4: Leaf nodes generation for each symbol & sequential sorting.

Step5: Select two least frequent characters to combine them logically to obtain combined frequency leading to the binary tree structure formation.

6. ANALYSIS

As per implementation of the algorithm, the following results have been analysed. Table 1 evaluates Huffman encoding algorithm based on certain parameters such as compressed file size, compression ratio, and compression and decompression time. The minimum compression ratio is achieved by image 1 which is 0.40.

Table 1: Parametric comparison of Huffman encoding

Sno	Original Image	Original File Size (bytes)	Compressed File Size (bytes)	Compression Ratio	Compression (secs)	Time	Decompression (secs)	Time
1	Image 1	472424	169571	0.40	0.45		0.29	
2	Image 2	473831	244038	0.52	0.49		0.56	
3	Image 3	786486	247482	0.38	0.51		0.45	

Run length encoding algorithm is evaluated on the performance metrics taken into consideration as shown in Table 2 where the least time taken to compress the image is 0.60 secs by Image 1 and the amount of time required for decompression is 1.05 secs.

Table 3 depicts the output of the input images using Huffman encoding and Run-length encoding. In this paper, 3 images are taken into consideration.

Figure 5 represents the performance analysis of Huffman encoding and

Run-length encoding based on compressed file size. As can be seen, for image 1, the original image size is 472424 bytes, the size of the image is reduced to 169571 bytes by Huffman encoding and through RLE, and it is reduced to 179607 bytes. Considering image 2, the input image size is 473831 bytes, and the compressed image size received is 244038 bytes and 265114 by Huffman encoding and RLE, respectively. For image 3, the input image size is 786486 bytes, RLE has compressed it to 248554 bytes and after applying Huffman encoding the size is reduced to 247482.

Table 2: Parametric comparison of Run-Length encoding

Sno	Original Image	Original File Size	Compressed File Size (bytes)	Compression Ratio	Compression (secs)	Time	Decompression Time (secs)
1	Image 1	472424	179607	0.38	0.6	1.05	
2	Image 2	473831	265114	0.48	1.08	1.9	
3	Image 3	786486	248554	0.32	0.97	1.39	

Table 3: Qualitative Analysis of images

S No	Original Image	Huffman Encoding	Run Encoding	Length
1				
2				
3				

Figure 6 depicts the performance analysis of RLE and Huffman encoding through compression time. For image 1, the least time taken to compress is 0.45 secs by Huffman encoding and on the contrary RLE compresses the image in 0.60 secs. Image 2 is compressed in 0.49 secs through Huffman

encoding whereas RLE compresses it in 1.08 secs. Image 3 is compressed in 0.51 secs through Huffman encoding and on the other hand RLE compresses it in 0.97 secs.

Figure 5: Performance Analysis based on Compressed File Size

Figure 7 represents the performance analysis of Huffman encoding and RLE through decompression time. Image 1 using RLE takes 1.05 secs to decompress whereas using Huffman encoding it takes 0.29 secs. For image 2, the time required to retrieve the original image in case of Huffman encoding is 0.56, on the contrary for RLE it is 1.9 secs. Similarly, for image 3 using RLE decompression time is 1.39 secs and through Huffman encoding it is 0.45 secs.

Figure 6: Performance Analysis based on Compression Time

Figure 7: Performance Analysis based on Decompression Time

Figure 8 depicts the performance analysis based on compression ratio. Considering image 1, the compression ratio achieved by RLE is 0.38 whereas that of Huffman encoding, it is 0.40. For image 2, the ratio achieved by Huffman encoding is 0.52 and that for RLE it is 0.48. The image compression ratio for image 3 using RLE is 0.32 and that with Huffman encoding is 0.38.

Figure 8: Performance Analysis based on Compression Ratio

7. CONCLUSION

From the perspective of compression & decompression time, Huffman encoding is less time consuming than RLE. In case of compressed file size Huffman encoding gives better results. Considering the parameter compression ratio, it is evident that RLE has achieved better results.

Nevertheless, we can also say the performance of the compression technique is influenced by original file content. In this paper, two lossless data compression algorithm are compared on image data. However, more parameters can be taken into consideration in order to analyze the performance of the two algorithms.

Moreover, both lossy and lossless algorithms will be considered over other different parameters in order to evaluate the performance of other compression algorithms. As the paper focuses on image data, other types of data can also be taken into consideration like audio, video and text.

REFERENCES

Ajala F. A., Adigun A.A, Oke A.O. (2018). Development of Hybrid Compression Algorithm for medical images using Lempel-Ziv-Welch and Huffman Encoding. *International Journal of Recent Technology and engineering*, 7(4), 1-5.

Büyükoğlu, O. (2019). Simultaneous Image Encryption and Compression Using Modified Huffman Tables. *International Conference on Digital*

Information Processing and Communications (ICDIPC), Trabzon, Turkey, 47-51.

Canard, S., Carpo, S., Kuate, D., Sirdey R. (2017). Running compression algorithms in the encrypted domain: a case study on the homomorphic execution of RLE. *IEEE*.

Dath D., Panicker, J. (2017). Enhancing adaptive Huffman coding through word by word compression for textual data. *International Conference on Communication and Signal processing (ICCSP)*, Chennai, 1048-1051.

Erdal, E., Erguzen, A. (2019). An efficient encoding algorithm using local path on Huffman encoding algorithm for compression. *Applied Sciences*, MDPI, 1-19.

Albahadily, H., Tsviatkou, V. (2016). New modified RLE algorithm to compress gray scale image with lossy and lossless compression. *International journal of advanced computer science and applications*, 7(7), 250-255.

Hussaina, A., Al-Fayadh, A., Radi N., (2018). Image compression techniques: A survey in lossless and lossy algorithms. *Neurocomputing*, Volume 300, Pages 44-69.

Made, I., Suarjaya, A. (2012). A new algorithm for Data compression optimization. *International journal of advanced computer science and applications*, 3(8).

Karthikeyan, B. (2014). A Performance Analysis of Different Scanning Paths on Lossless Image Compression for Radiographic Welding Images, India, *Journal of Scientific & Industrial Research* Vol.73.

Khaitu, S., Panday, S. (2018). Canonical Huffman Coding for Image Compression. *IEEE International Conference on Computing, Communication and Security (ICCCS)*, Kathmandu 184-190.

Khassaweneh, M., Alshorman, O. (2020). Frei-Chen bases lossy digital image compression technique. *Applied computing and informatics*, <https://doi.org/10.1016/j.aci.2019.12.004>

Klein, S., Saadia, S., Shapira, D. (2019). Better Than Optimal Huffman Coding?. *Data Compression Conference (DCC)*, Snowbird, UT, USA, 582-582.

Narmatha, C.; Manimegalai, P.; Manimurugan, S. (2017). A Lossless Compression Scheme for Grayscale Medical Images Using a P2-Bit Short Technique. *Journal of Medical Imaging and Health Informatics*, 7(6), 1196-1204.

Pal, C., Pankaj, S., Akram, W., Acharyya, A., Biswas, D. (2018). Modified Huffman based compression methodology for Deep Neural Network Implementation on Resource Constrained Mobile Platforms. *IEEE International Symposium on Circuits and Systems (ISCAS)*, Florence, 1-5.

Patil, R., Kulat, K. (2017). Audio compression using dynamic Huffman and RLE coding. *International Conference on Communication and Electronics Systems (ICCES)*, Coimbatore, 160-162.

Rafea, S., Salman, N. (2018). Hybrid DWT-DCT compression algorithm and a new flipping block with an adaptive RLE method for high medical image compression ratio. *International Journal of Engineering and Technology*, 7(4), 4602-4606.

Setyaningsih, E., Wardoyo, R., (2017). Review of image compression and encryption techniques. *International Journal of Advanced Computer Science and Applications*, 8(2).

Sharma, R., Bollavarapu, S., (2015). Data Security using Compression and Cryptography Techniques. *International Journal of Computer Applications*, 117(14), 15-18.

Singh, S., Devgon, R., (2019). Analysis of Encryption and Lossless Compression Techniques for Secure Data Transmission *International Conference on Computer and Communication Systems (ICCCS)*, DOI: 10.1109/CCOMS.2019.8821637, 120-124.

Sitaridi, E. (2016). Massively-Parallel Lossless Data Decompression, *International Conference on Parallel Processing (ICPP)*.

Souley, B., Dass, P., Tanko, S. (2014). A comparative analysis of data compression techniques. International Journal Applied Sciences, 63-82.

Thanki, Rohit M., Kothari, Ashish. (2019). Data Compression and Its Application in Medical Imaging. Hybrid and Advanced Compression Techniques for Medical Images, 1-15.

Thanki, R., & Kothari, A. (2019). Color Medical Image Compression Techniques. 83-92

Uthayakumar, J., Vengattaraman, T., & Dhavachelvan, P. (2018). 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. doi: 10.1016/j.jksuci.2018.05.006.