An Approach to Select Efficient Data Optimization Techniques in Connected Cars

Ayushi Jain[1] Durgesh Nandan[2]
Symbiosis institute of technology, Symbiosis International (Deemed University), Pune, India

durgeshnandano51@gmail.com

Abstract

To make data storage more successful, even using much storage, data can be compressed. By compressing data, information can be shared much more quickly. There are multiple techniques available for data optimization. Each approach also provides a different set of outcomes. This paper will discuss the optimization techniques using four different algorithms: Huffman encoding, Lempel Ziv Welch, Run Length Encoding, and Shannon Fano methods. This essay shows how a method performs compression and which method is more appropriate and effective when utilized to perform real-based data compression. The result of a technique can be determined by the compression file size, which is less than the original file.

Keywords: *Data compression, connected cars, compression techniques, Lossless compression, Real data-based compression*.

1. Introduction

The connected car, which is a vehicle with Internet connectivity, communication capabilities with other vehicles and road infrastructure, and the ability to gather real-time data from multiple sources, is predicted to be a key player in the future Internet of Things [1]. With the support of hardware and software that facilitate widespread knowledge transfer rapidly via the internet, around the world, technology is developing quickly. Information technology [2] professionals can effortlessly by sending information over the internet. Not all data can be sent easily, though. To decrease the quantity of data that needs to be kept and transferred while easing data transmission, a data set is converted into a code through the process of compression. Compression can help cut down on time and memory [3] requirements (storage). The Huffman, Lempel Ziv Welch, Run Length Encoding, and Shannon Fano methods are only a few examples of efficient compression algorithm techniques. The optimization method is shown in Figure 1.

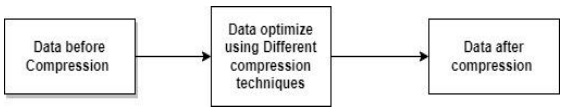

Figure 1: Block diagram of Data compression process

Figure 1 describes the general data compression process [4]. When data is not compressed, lossless compression is used to process the uncompressed tour to compress its size. The file will be smaller than it was before compression once the content has been reduced. A file's size is decreased through compression from a huge size to a smaller size.

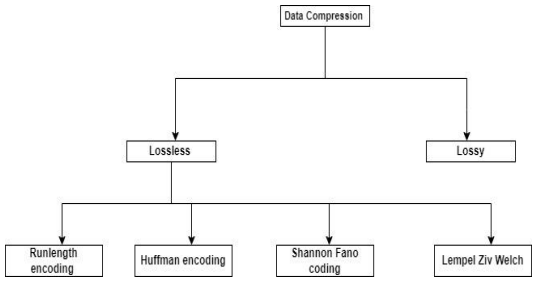

Figure 2: Types of data compression

Figure 2, there are two compression techniques named lossless and lossy compression techniques [5]. In the lossy compression technique, the data in a file is takeout and brought bring back to its original form after decompression. Lossy compression is generally used to compress multimedia data (audio, video, and images). In lossless compression techniques, it belongs to a kind of data compression that makes it possible to recover the main data exactly from the optimized data lack of losing any details. Since most real-world data exhibits statistical redundancy, lossless compression is practical. It maintains quality. In lossless compression, it uses different algorithms for compressed data like as, Huffman encoding, Shannon Fano coding, Run-length encoding, and the LZW method [6]. Here is an explanation of how data compression can be executed.

- Compression for audio
- Compression for text
- Compression for video
- Compression for image

There are different explanations for how data compression can be executed. As we are working on Real-based data so, it required text data for compression [7]. When text is optimized, the decompression procedure resets the compressed file to the beginning of the text. The results of decompression depend on whether Lossless Compression or Lossy Compression was applied. If a text has completed lossless compression, the main details

can be precisely retrieved from the uncompressed file. Lossy compression causes some information to be lost, and the text generated after decompression cannot fully equal the text from the original data. It is explicitly evaluated and compared to how well various approaches compress text data.

2. Literature Review

In this paper [8], A novel multilevel Huffman coding-based test database optimization method was proposed. The suggested method works with IP cores whose structural details are unknown. The technique encrypts three of the primary data kinds. The same Huffman codewords produce better compression outcomes. According to most of the relevant procedures from the literature, the efficiency of the space above the intended Decompressor is relatively low.

In this paper [9], It can be concluded that the Shannon-Fano algorithm for data compression has been very well achieved using Modalism SE 6.4 simulator and VHDL coding and that data is compressed using these techniques. The Shannon-Fano algorithm equation creates a very effective compression strategy for determining how much data is compressed.

In this paper [10], Arithmetic coding provides the best compression, but its slow execution can be a drawback. Arithmetic coding's efficiency and built-in separation of coding and modelling are its key benefit for statistical data compression. The drawbacks of arithmetic coding are its slow performance, implementation complexity, and shortage of prefix codes.

In this paper [11], For most of the text files, the combination of the RLE and LZW compressors will result in somewhat better compression than either RLE or LZW alone. Since both the RLE and the LZW algorithms take benefit of common redundancy in text-based files (i.e., repetitiveness or multiple examples of phrases), combining this beneficial Aspect of the two algorithms into one algorithm can lead to better results than the individual performance, but in the end, they find that individual performance gives more compression ratio.

3. Data Optimization

3.1 Different algorithms

Run length encoding

RLE, also known as run-length encoding, is the most straightforward data compression approach. This algorithm distinguishes between runs and non-runs by identifying successive symbol sequences as runs. This algorithm handles a certain amount of redundancy. Based on their redundancies and their lengths, it evaluates whether there are any repetitive symbols. All other sequences are regarded as non-runs, while consecutive recurrent symbols are labelled as runs. For instance, if the file "XYXZZZZX" is chosen to optimize, the first three letters are regarded as a non-run with a length of 3, while the following four characters are regarded as a run with a length of 4 because the symbol Z is repeated. This algorithm's primary priority is to locate the runs in the source file and to mark each run's symbol and length. While storing all the non-runs and not using any of

those runs for the compression process, the RLE algorithm [12] uses those runs to optimize the main research file.

Huffman Encoding

ASCII character data compression is the focus of Huffman coding. Numerous types of data, including text, audio, video, and images, are compressed using it. This method is based on developing a complete binary tree for each symbol in the original file after figuring out the probability of each symbol and sorting the symbols by reducing probability**.** Lossless compression techniques are included in the Huffman compression algorithm. A compression technique known as lossless compression does not alter the underlying data information to make it smaller**.** Huffman's approach is that each ASCII character is typically represented by 8 bits. As an illustration, if a file has the character "UUVWX" in a row, it has 40 bits, 5 bytes, or 5 bits. We only require a file that is 10 bits in size (0010111110) if each character is assigned a code, such as U = 0, V = 10, W = 111, or X = 110. that specifies that codes must be identical, or that a code cannot be generated from some other code [13].

Shannon Fano Encoding

Shannon methods, which replace each symbol with a binary code whose length is calculated based on the likelihood of the symbol, were at the time the best method, but after the Huffman encoding, they were hardly ever utilized. Shannon Fano is a method for generating a prefix code [14] based on a combination of symbols and probabilities in the area of data optimization. Huffman encoding, on the other hand, is more capable of producing the code.

Lempel Zev Welch Algorithm

In general, the LZW method uses a dictionary and is a lossless compression algorithm. Dictionary-based techniques do this instead of having a statistical model as their core. A dictionary is a collection of all words that can be used in a language. Larger and more frequent dictionary words are represented by the entries' indexes, which are preserved in a table-like format. The most popular method is known as the LZW algorithm [15]. This methodology stores and indexes the previously observed string patterns in a dictionary. Instead of repeating string patterns, these index values are used during compression. Instead of using repetitive string patterns, these index values are used during compression. The dictionary is generated dynamically during the compression process; thus, it is not necessary to send it along with the encoded message for decompression. During decompression, the same dictionary is dynamically created.

3.2 Measuring compression performance

Depending on the application, a compression algorithm's performance can be evaluated using a variety of factors. When assessing performance, space efficiency would be the main factor to take into consideration. The efficiency of using time is another factor. Because the behaviour of the compression is based on the symbol repetition in the main file, evaluating a compression technique's overall performance may be difficult. As a result,

calculating effectiveness is challenging, and many measurements should be used to analyze the performance of those compression categories. The measurements used to evaluate how well lossless algorithms work are listed below.

Compression Ratio: It describes the proportion of the source file's size to the compressed file's size.

$$CR = \frac{Size\ after\ compression}{size\ before\ compression}$$

Saving percentage:

$$Saving\ \% = \frac{size\ before\ compression - size\ after\ compression}{size\ before\ compression}\%$$

Compression factor: It is the absolute opposite of compression ratio.

$$CF = \frac{Size\ before\ compression}{size\ after\ compression}$$

4. Methodology

The Run Length Encoding Algorithm, Huffman Encoding Algorithm, Shannon Fano Algorithm, and Lempel Zev Welch Method are implemented and analysed in a set of data files to evaluate the performance of lossless compression methods. The mentioned factors are evaluated to measure results.

LZW Algorithm Performance Analysis

Entropy and code efficiency are not planned for this technique because it is not based on a statistical model. Calculations are performed regarding the compression and decompression processes, file sizes, compression ratios, and saving percentages [16].

Run length encoding Algorithm Performance Analysis

The Run Length Encoding Algorithm produces the File Sizes, Compression Ratio, and Saving Percentage since it does not use any statistical techniques to perform compression. For calculating, a variety of files with multiple source patterns and file sizes are used.

Huffman and Shannon Fano encoding Algorithm Performance Analysis

Implemented and carried out separately are Shannon Fano and Huffman's Encoding methods. Calculations are performed regarding the File sizes, compression ratio, and saving percentage [17].

Comparing the performance

The selected methods function differently depending on the calculations; while one approach offers a maximum saving percentage, it can take a longer processing time. Therefore, to pick the best option, all these aspects are examined. The best algorithm is one that produces a respectable saving percentage in a reasonable period.

5. Results and Comparison

Two text files retrieved from GitHub Real-word file [18] (open source) with various file sizes and distinct contents, i.e., actual-base data, are tested using four lossless compression algorithms. The original text files are 4096 bytes and 16384 bytes in size.

Evaluate the performance of the different Algorithms:

Table 1: shows the compression Ratio

File size		**Run-length encoding**	**Shannon Fano coding**	**Huffman encoding**	**LZW method**
4096	Compressed file	1919	1877	1650	1450
	Compression ratio	0.531	0.541	0.591	0.645
16384	Compressed file	7707	7100	6707	5707
	Compression ratio	52.96%	56.66%	59.06%	65.16%

Table 2: shows the comparison between saving%, compression factor, and ratio

Parameter	**Run-length encoding**	**Shannon Fano encoding**	**Huffman encoding**	**LZW method**
Compression Ratio	0.531	0.541	0.591	0.645
Compression factor	1.88	1.84	1.69	1.55
Saving percentage	53.14%	54.17%	59.71%	64.59%

As can be observed in tables 1, and 2, the relative compression ratios, compression factor, and saving percentage are displayed in each compression strategy. The run length encoding has the moderate compression ratio of all data sets. Nowadays, lossless data compression rarely uses RLE [19]. Based on the information that is currently known regarding compression ratio, the Huffman encoding strategy is determined to be the best alternative because it focuses only on reducing input data redundancy [20]. Although Huffman encoding, which has a moderate compression factor and compression percentage, seems to produce the outcomes that Shannon Fano encoding most nearly fits.

Dictionary size is an important factor in Lempel-Ziv-Welch encoding's success in achieving greater compression ratios. Therefore, when compared to other compression techniques, the results of lower dictionary sizes are reduced.

6. Conclusion and Future Scope

Our text bed had a limited amount of text data, thus we compared four lossless data compression algorithms in this research. In the future, a larger test bed including audio, video, and image data may be used to create more compression methods (both lossless and lossy). After that, a system that can find out the file and then choose the best compression methodology for that file can be put into place. It was performed as an experimental comparison of various lossless text data compression algorithms. The effectiveness of various known lossless compression techniques is evaluated. Although they are evaluated on various file types, the focus is primarily on multiple test patterns. The Lempel Zev Welch algorithm can be regarded as an effective technique among the shortlisted ones by considering the compression ratio, file size, compression factor, and saving percentages of all the algorithms. These algorithmic parameters are within a reasonable variety, and it produces effective outcome for big data.

References

[1] Coppola, R. and Morisio, M., 2016. Connected car: technologies, issues, future trends. *ACM Computing Surveys (CSUR)*, *49*(3), pp.1-36.

[2] Solanki, V.K. and Dhall, R., 2017. An IoT-based predictive connected car maintenance approach.

[3] Ranjan, A., Raha, A., Raghunathan, V. and Raghunathan, A., 2020. Approximate memory compression. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, *28*(4), pp.980-991.

[4] Zhaoping, L., 2006. Theoretical understanding of the early visual processes by data compression and data selection. *Network: computation in neural systems*, *17*(4), pp.301-334.

[5] Kavitha, P., 2016. A survey on lossless and lossy data compression methods. *International Journal of Computer Science & Engineering Technology*, *7*(03), pp.110-114.

[6] Shanmugasundaram, S. and Lourdusamy, R., 2011. A comparative study of text compression algorithms. *International Journal of Wisdom Based Computing*, *1*(3), pp.68-76.

[7] Bhattacharjee, A.K., Bej, T. and Agarwal, S., 2013. Comparison study of lossless data compression algorithms for text data. *IOSR Journal of Computer Engineering (IOSR-JCE)*, *11*(6), pp.15-19.

[8] Kavousianos, X., Kalligeros, E. and Nikolos, D., 2007. Multilevel Huffman coding: An efficient test-data compression method for IP cores. IEEE transactions on computer-aided design of integrated circuits and systems, 26(6), pp.1070-1083.

[9] Vaidya, M., Walia, E.S. and Gupta, A., 2014, August. Data compression using the Shannon-fano algorithm implemented by VHDL. In 2014 International Conference on Advances in Engineering & Technology Research (ICAETR-2014) (pp. 1-5). IEEE.

[10] Howard, P.G. and Vitter, J.S., 1994. Arithmetic coding for data compression. Proceedings of the IEEE, 82(6), pp.857-865.

[11] Moronfolu, D.O., 2009. An enhanced LZW text compression algorithm. Afr. J. Comp. & ICT, 2(2), pp.13-20.
[12] Bradley, S.D., 1969. Optimizing a scheme for run length encoding. *Proceedings of the IEEE*, *57*(1), pp.108-109.
[13] Fitriya, L.A., Purboyo, T.W. and Prasasti, A.L., 2017. A review of data compression techniques. *International Journal of Applied Engineering Research*, *12*(19), pp.8956-8963.
[14] Rochesteresti, D.A., Purboyo, T.W. and Prasasti, A.L., 2017. Comparison of Data Compression in Text Using Huffman, Shannon-Fano, Run Length Encoding, and Tunstall Method. *International Journal of Applied Engineering Research*, *12*(23), pp.13618-13622.
[15] Nandi, U. and Mandal, J.K., 2012, November. A compression technique based on the optimality of LZW code (OLZW). In *2012 Third International Conference on Computer and Communication Technology* (pp. 166-170). IEEE.
[16] Semunigus, W. and Pattanaik, B., 2021, July. Analysis for Lossless Data Compression Algorithms for Low Bandwidth Networks. In *Journal of Physics: Conference Series* (Vol. 1964, No. 4, p. 042046). IOP Publishing.
[17] Kodituwakku, S.R. and Amarasinghe, U.S., 2010. Comparison of lossless data compression algorithms for text data. *Indian journal of computer science and engineering*, *1*(4), pp.416-425.
[18] Findings from GitHub: Methods, Datasets, and Limitations.
[19] Porwal, S., Chaudhary, Y., Joshi, J. and Jain, M., 2013. Data compression methodologies for lossless data and comparison between algorithms. *International Journal of Engineering Science and Innovative Technology (IJESIT) Volume*, *2*, pp.142-147.
[20] ChenGhen, Y., Qu, Z., Zhang, Z. and Yeo, B.L., 2004, April. Data redundancy and compression methods for a disk-based network backup system. In *International Conference on Information Technology: Coding and Computing, 2004. Proceedings. ITCC 2004.* (Vol. 1, pp. 778-785). IEEE.

Biographies

Ayushi Jain received a bachelor's degree in electronics and communication engineering from Radharaman engineering college, RGPV University, Bhopal Madhya Pradesh. Currently pursuing master's degree in Embedded systems from Symbiosis Institute of Technology, Symbiosis International (Deemed University), Pune Maharashtra. She is in her second year of her master's. She is currently working as an Automotive System Engineer Intern at Company.

Dr. Durgesh Nandan did his Doctor of Philosophy (Ph.D.) from Department of Electronics & Communication Engineering, Jaypee University of Engineering and Technology, Guna, Madhya Pradesh, India in year 2018 with the specialization in VLSI. Currently, he is working as Assistant Professor (SG-8000 AGP), E&TC, Symbiosis Institute of Technology, Symbiosis International (Deemed University), Pune Maharashtra.