Proceedings of the Fifth International Conference on Inventive Computation Technologies (ICICT-2020)
IEEE Xplore Part Number:CFP20F70-ART; ISBN:978-1-7281-4685-0

Comparison of Lossless Data Compression Techniques

Athira Gopinath
Center for Wireless Networks & Applications (WNA)
Amrita Vishwa Vidyapeetham
Amritapuri
India
athirag@am.amrita.edu

Ravisankar. M
Center for Wireless Networks & Applications (WNA)
Amrita Vishwa Vidyapeetham
Amritapuri
India
ravisankar@am.amrita.edu

***Abstract*—The transmission of humongous data from the monitoring field to the central unit for further processing is a highly challenging task for the hardware, especially when the communication bandwidth is limited. The data overflow in the single-board computer will lead to the loss of valuable data. Since the sampling frequency of our application is fixed and the bandwidth of the communication channel cannot be increased, the only way by which the data overflow and loss can be avoided is by using data compression techniques. Data compression minimizes the size of the data to be stored and communicated, by reducing the size of data file either by using lossless or lossy compression techniques. This research focuses on a comprehensive study of different lossless compression methods by applying data files to each compression method. To estimate the best method of compression for storage and communication based on comparing the compression ratio, compression rate, space-saving, and execution time of existing methods of lossless compression.**

Keywords— Data compression, Lossless methods, Lossy methods, Compression ratio, Compression rate, Space Saving

INTRODUCTION

Real-time applications such as disaster management [14], smart buildings [15], smart health [16] etc. require a huge amount of information for its analysis. Often the rate of data generation is more than the rate of transmission of data to the monitoring station. This results in the overflow of data in the data collection station and subsequently, loss of precious data. To overcome these hurdles, data compression techniques are extensively cast-off to handle wide records for storing and transmission.

Data Compression, also known as compaction, is a method which reduces the amount of data needed to be saved and transmitted.Digitally,data compression can be achieved either by lossless(exact) and loss(inexact) compression. Lossless compression method, the integrity of the data is preserved so that decompression of the compressed data will give the original data. The redundant is removed during the compression and added during the decompression. Lossless methods are usually applicable only when there is no compromise in data loss [1]. On the other hand, lossy compression techniques have the benefits to provide a better compression ratio than lossless compression methods, with some data loss. Lossy algorithms are castoff mainly for immobile images, videos and audio files since the loss cannot be perceived by human beings or the brain fills up the loss. On the other hand, lossless algorithms are used to compress file data such as numeric data, text file etc., since programs that method such file data cannot endure faults in the data [2].
Different lossless and lossy compression techniques are explained along with the basic measurement parameters of a compressed file [3]. In [4], different lossless compression techniques have applied to the image and found that the lossy compression technique is used for JPEG. The hybrid approach method, which is the combination of different lossless compression, is used to reduce the time of execution [5]. The [6] compares Run length and Huffman using text data. Huffman algorithm [7] is one of the best lossless compression methods rather than other conventional methods. Delta encoding, class-based delta model to enhance the compression ratio and reduction in the complexity [8]
This research work aims to study different lossless compression techniques using accelerometer data, also find the compression ratio, compression rate, space-saving and execution time of each encoding technique in the lossless method. The analysis of each lossless compression technique is achieved using Matlab 2019a.
Section 1 elaborates the Methods and implementation of three lossless compression techniques, section 2 deals with the various measurement parameters, section 3 gives the analysis and discussion and section 4 conveys the conclusion and future work.

METHODOLOGIES AND DISCUSSION

Basic block diagram for data transmission and reception as shown in Fig1

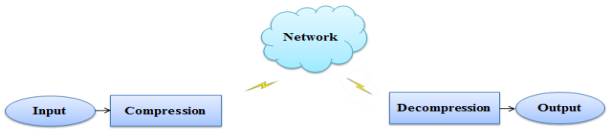


Fig.1 Basic Block diagram

The basic block diagram in Fig.1 involves the input data stream, compression, network layers, decompression and output. The input data can be either text, audio or video which is given to the compression block before it sends to the network. The compression can be lossless or lossy based on the requirements of the end-user. The compressed data is forwarded to the network layer to send from the transmitter to the receiver. When the data is received by the receiver, it is then passed to the decompression block to retrieve the original data.

Lossless Methods can be classified as shown in Fig 2

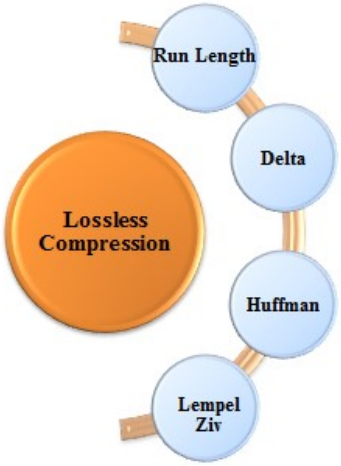


Fig.2 Classification of Lossless Data Compression

In this research work, comparing the frequently used lossless techniques to compress the accelerometer data and wish to find out the best method with respect to measurement parameters. Basic lossless compression methods are described in the subsection.

A. Run-length Coding

Run-length encoding (RLE) is a modest lossless compression method, suited for the repetitive data by replacing the consecutive occurrence of a symbol with its count of occurrence and the symbol itself [9]. Regardless of information content, RLE can be applicable for any kind of data, but the compression ratio of RLE depends on the content of the data. If the dataset is not having any repetition, then the RLE would essentially escalate the size of a file. The compression ratio of RLE is very less when compared to the advanced compression techniques. Although the ease of implementation and execution shaped the RLE a better substitute for the complex compression algorithms. The method can be more efficient if the data uses only two symbols in its bit pattern and one bit is more frequent than the other. It is sustained in most of the bitmap files like TIFF, BMP, and PCX.

RLE mainly works for decreasing the physical size of a repeated character string by replacing it with its count of occurrence and one of the repeated symbols itself. The rerun string is called the run. RLE will encode the repeated string two bytes, the first byte consists of the number of characters in that run noted as run count and the second byte represents the value of the character in a run called run value. From the practical point of view, the encoded run may contain 1to 128 or 256 characters, the run count will be the number of characters minus one and the run ranges from 0 to 255.

An uncompressed packet of data, the run is RRRRRRRRRRRRRRRRRR, then run count will be 20 and run value will be R. The encoded data will be in the form of 20R. The physical size of the string is reduced to 2-bytes. Another set of data is AAAAAAGGGGGAAAASSSMM, the encoded file in the form of 10A5G3S2M. This file is compressed to four 2-byte packets. Hence the 15-bytes of the data can be represented by 8 bytes of data after RLE with a compression ratio of 2:1.RLE finds its wide application in the text with more identical symbols, where it can replace recurring symbols with its length of symbols. RLE schemes are very simple fast,and a good alternative for the complex compression algorithms. But the compression efficiency of the RLE algorithm depends on the type of data being encoded.

B. **Delta Encoding**

Delta [10] compression is a kind of simplest compression technique used for storing and transmitting humongous data. This encoding method uses the difference between the sequential samples to store the differences rather than the original samples. Delta compressed file contains the first value as such in the original file and the difference of the samples. The algorithm is trying to make the value of the samples nearer to zero. The reason is that the original file contains the samples with higher digits, then each value consumes some space, if it reduced nearer to zero then it consumes less space than the original. At the same time, also ensured that there should be data loss after the compression. That's how the algorithm works to reduce the file size. This algorithm is highly applicable for the smooth data, ie is the small changes between adjacent samples. At the receiver the reverse process is taking place, the differences are getting added to the first sample and retrieve back the second sample and further the other samples. Whatever be the transmitted data, the same data can be fetched at the receiver side without any loss. It finds wide application in HTTP, online backup services, delta copying, Git.

Consider a temperature sensor, which is deployed in a room and collecting the temperature samples for a few hours, so it should be shown in Fig.3.

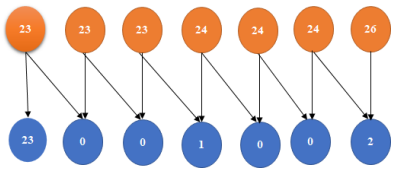


Fig.3 Delta Encoding

The temperature sensor senses the values as 24, 23, 25,.. ., the delta compression algorithm finds the delta value of the successive samples by maintaining the first sample as such in the original file. The delta values are received at the receiver side without any data loss and can be retrieved using the reverse algorithm.

C. *Huffman Encoding*

Huffman Encoding, is a type of greedy algorithm for lossless data compression. It utilizes a variable-length encoding where the variable lengths are allocated to each character in the string. It will give shorter codes to symbols that occur more frequently, whereas longer codes to symbols which occur less frequently. In order to avoid ambiguities during the time of decoding, Huffman code [11] follows the prefix rule ensuring that the code allotted to any character is not a prefix of the code allotted to any other character. The main stages for Huffman coding are described below,

- Construct a Huffman tree based on the input string
- Codes are assigned to characters by traversing the Huffman tree

Steps to build a Huffman tree

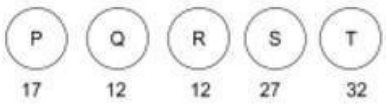
1. Started by creating a leaf node with all the characters in the string and assign the priority based on its probability of occurrence
2. Sort the nodes based on its increase in priority
3. Select the two nodes with a minimum frequency
4. Create a new internal node with the frequency equal to the sum of the frequency of the two nodes. Primary node is considered as the left child and another node as the right child of the newly created node
5. Repeat steps 3 and 4 till all the nodes form a single tree

Consider a text file that uses only five characters PQRST, prior to assigning the bit pattern to the characters, attribute weight to each character according to the probability of occurrence known as the frequency. The frequency of the character are shown in the table below

Table I: Frequency of the character

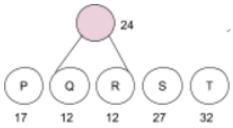
Character	**P**	**Q**	**R**	**S**	**T**
Frequency	**15**	**12**	**25**	**12**	**20**

Let PQRST be the string which wants to encode, the initial step is to create a Huffman tree. The characters and their frequency are shown in the above table.



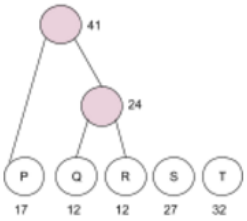
(a)

The building of a Huffman tree starts with sorting the list as per the frequency. Then select the nodes which have the minimum frequency called leaf nodes. Obtain the parent node by adding the frequencies of two leaf nodes. Now the leaf nodes are removed from the list and the parent will be in place of those two leaf nodes.

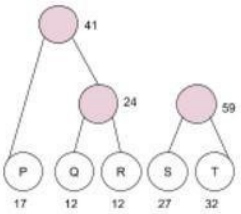


(b)

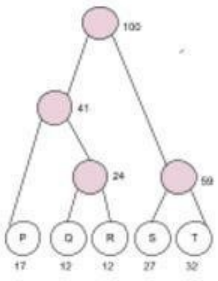
The process turns on repeating until only one character present in the list



(c)

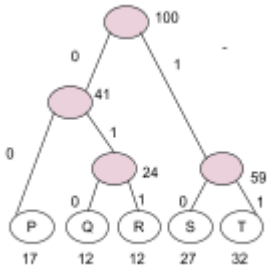


(d)



(e)

The node at the top position of the tree is the root node. After creating the Huffman tree, the next step is to assign the codes to each character in the string. Assigning the code requires navigating the tree from the root node to the bottom node. From the root node, giving 0 to the left side and 1 to the right side of the node.



(f)

Fig.3(a-f) Construction of Huffman Tree

Table II: Code

Symbol	Code
P	00
Q	010
R	011
S	10
T	11

Table II shows the code assigned to each of the symbols presented in the tree. During the Huffman encoding the symbols are replaced by the binary codes and transmit the codes instead of the symbols.

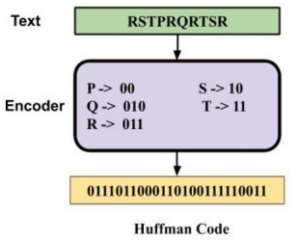


Fig.4 Huffman Encoding

Decoding at the reception side is simple, it requires the Huffman tree. The method is to iterate the binary encoded data, start from the root of the Huffman tree and reach until it finds a leaf. The decoding method starts with reading the encoded data stream, if it finds a zero, move to the left side of the Huffman tree. Similarly, if the input stream is one, then move to the right side of the Huffman tree. During the traversal, if it finds any leaf node, note the character of that leaf node and continue the iteration until it covers all the leaf nodes.

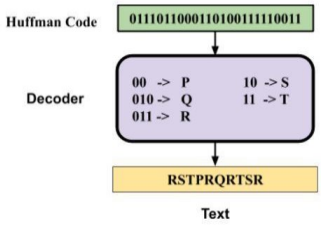


Fig.5 Huffman Decoding

The compression ratio of the Huffman algorithm is much better than RLE. However, it is necessary to convey the Huffman table at the commencement of the compressed file. Otherwise, the decompressor will not be able to decode it, which leads to cause overhead. In Huffman coding all the codes in the compressed data are of varying size. Therefore at the decoder side, it is very difficult for the decoder to understand that the code has reached the last bit and only the solution to understanding this by following the paths of the top to bottom of the tree and reaching an end of it. Although if the compressed data is distorted with bits missing or additional bits added, then whatever code is decoded at the decoder will be garbage

D. LEMPEL- ZIV-WELCH

Lempel Ziv coding [12] is a simple dictionary-based lossless encoding, typically used in GIF, optionally in TIFF and PDF. The implementation of the Lempel Ziv algorithm is very easy and has a high throughput in hardware implementation. The algorithm depends on the recurring patterns to redeem the data space. The algorithm finds its wide application in the Unix file, GIF image format. The simplicity and versatility model the LZW algorithm the foremost technique for general purpose data compression. The LZW algorithm simply replaces the strings of character with a single code. The flow of the LZW algorithm is started by reading the sequence of the symbol, then grouping the symbols into strings, and at the end converting the strings to codes. The codes are because the codes consume less space than the strings. The algorithm poses a code table with 4096 as a plain choice for the number of table entries. In that 4096 entries, codes 0-255 in the code table are regularly attributed to serving a single byte from the input file. At the initial time of encoding, the code table encloses with the first 256 entries and the remnant of the table are being an abyss. The LZW compression is accomplished by employing codes 256 through4095 to symbolize the sequences of bytes. The LZW algorithm detects the recurring sequences in the data and adds them to the code table while the encoding proceeds. Decoding can be simply done by retrieving each node from the compressed file and decode it through the code table to obtain the character it substitutes.

During the compression phase, two subsequent processes will take place

- Constructing an indexed dictionary
- Compressing a string of symbols

The LZW algorithm primarily extracts the smallest substring which is not present in the dictionary from the remnant

uncompressed string. A replica of the substring is added to the dictionary as a new entry and allotted with an index value. Compression is performed by replacing the substring with the index in the dictionary excluding the last character. Now the process inserts the last character index of the substring into the compressed string.

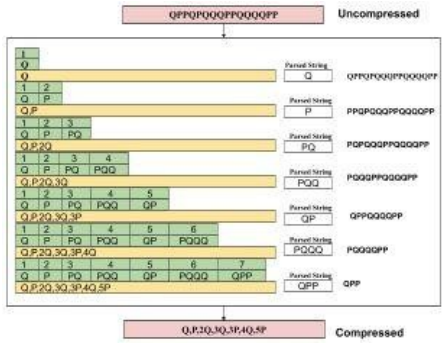


Fig.6 Lempel-Ziv Encoding

In the case of decompression, the inverse process of compression will take place. The decompression algorithm will take out the compressed substring and replace the indices with corresponding entries in the dictionary. The idea behind that is, when an index is encountered, there is already an entry for the encountered index.

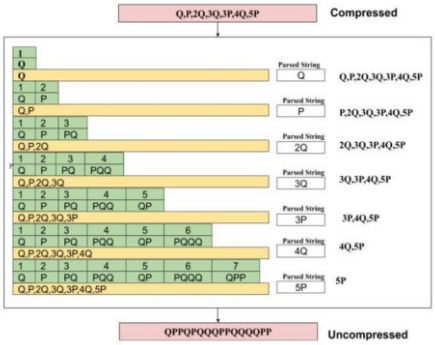


Fig.7 Lempel-Ziv Decoding

The main advantage of LZW algorithm is its simplicity, fast execution, and does not require any prior information regarding the input data stream. The compression will take place in a single pass. However the dictionary should be thrown out when the dictionary reaches its maximum size limit. This method is more efficient for a large amount of text data when the redundancy is high.

II. MEASUREMENT PARAMETERS

Compression ratios are the frequently used term in data compression algorithms, it is the calculation of the relative reduction in the size of the data represented by the different compression algorithm. Compression ratio of any compression algorithm is obtained by taking the ratio of uncompressed file size to the compressed file size. According to[13],the compression ratio can be calculated by,

$$Compression\ Ratio = \frac{Uncompressed\ file\ size}{Compressed\ file\ size}$$

Space-saving determines the truncation in size proportionate to the uncompressed size and it can be calculated using the following equation.

$$Space\ Saving = 1 - \frac{Compressed\ File\ Size}{Uncompressed\ File\ Size}$$

Compression Factor is the contrary of compression ratio, it can be calculated by taking the ratio of compressed file size to the uncompressed file size.

$$Compression\ Factor = \frac{Compressed\ File\ Size}{Uncompressed\ File\ Size}$$

Compression gain is the natural logarithm of the ratio of the size of input stream to the size of the compressed file obtained using any compression algorithm. 'CG' be the compression gain

$$Compression\ Gain = \log_e\left(\frac{Uncompressed\ file\ size}{Compressed\ file\ size}\right)$$

Compression Time is the time taken to execute the compression algorithm to compress the input file .

ANALYSIS AND DISCUSSION

TABLE III presents the performance evaluation of the four lossless compression algorithms. The accelerometer data is used as an input to the four lossless algorithms such as Delta encoding, Run Length, Huffman and Lempel-Ziv. Also various measurement parameters like compression ratio, space saving, compression factor, compression gain and elapsed time are determined using the accelerometer data. Based on the performance evaluation, it is found that Lempel Ziv compression provides a better compression ratio around 4:1, which will curtails the unwanted repetitive data by eliminating the statistical redundancy. Hence the efficiency of Lempel ziv method is enhanced by reducing the redundancy of the data. Also, 76.9% of space can be saved after doing the Lempel ziv compression method. But the Lempel ziv algorithm is a complex algorithm, it takes much compression time for the execution of the code.However, the Delta encoding also gives a comparably better measurement parameters rather than Run Length and Huffman.

TABLE III: Performance Evaluation of text data sequence

Parameters	Delta Encoding	Run length	Huffman	Lempel-Ziv
Original file size	5.85 KB	5.85 KB	5.85 KB	5.85 KB
Compressed file size	2.08 kB	3.46 kB	3.65kB	1.34 kB
Compression Ratio	2.803	1.68	1.602	4.33
Space Saving(%)	64.3	40.5	37.6	76.9

Compression Factor	0.356	0.595	0.624	0.230
Compression Gain	1.217	0.7296	0.695	1.880
Compression Time(Seconds)	0.00123	0.00542	0.05949	0.25954

CONCLUSION

The paper discussed the analysis on different lossless compression methods by evaluating the measurement parameters .From the analysis, Lempel ziv method shows better performance than the other lossless methods described. It has got a good compression ratio around 4:1 compression with 76.9% space saving. Delta encoding can combat the Lempel ziv algorithm when it accounts the compression time. At the same delta encoding shows a good performances when compared to the Run-length and Huffman encoding. Based on performance perspective, the algorithms to be used will depend upon the type of data. The analysis demonstrates, Lempel ziv is relevant to accelerometer data in terms of compression ratio and space saving .So it won't take extensive bandwidth to send the humongous data from the transmitter to the receiver.

ACKNOWLEDGMENT

We express our deep gratitude to our Chancellor and world renowned humanitarian leader Sri. (Dr.) Mata Amritanandamayi Devi (Amma) for her inspiration and support towards working on interdisciplinary research that has direct societal benefit.

REFERENCES

[1] Uthayakumar, J., T. Vengattaraman, and P. Dhavachelvan. "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 (2018).

[2] Prof. Dipti Mathpal and Prof. Mittal Darji, " A Research Paper on Lossless Data CompressionTechniques" IJIRST –International Journal for Innovative Research in Science & Technology| Volume 4 | Issue 1 | June 2017.

[3] Gupta, Ruchi, Mukesh Kumar, and Rohit Bathla. "Data Compression-Lossless and Lossy Techniques." International Journal of Application or Innovation in Engineering & Management 5.7 (2016): 120-125.

[4] Avudaiappan. T, Ilam Parithi.T, Balasubramanian.R And Sujatha.K, "Performance Analysis On Lossless Image Compression Techniques For General Images," International Journal of Pure and Applied Mathematics,Volume 117 No. 10 2017, 1-5

[5] Sidhu, Amandeep Singh, and Meenakshi Garg. "Research Paper on Text Data Compression Algorithm using Hybrid Approach." International Journal of Computer Science and Mobile Computing 3.12 (2014): 01-10.

[6] Ibrahim, A. M. A., & Mustafa, M. E. (2015). Comparison between (rle and huffman) algorithms for lossless data compression. IJITR, 3(1), 1808-1812.

[7] Vikash Kumar, Sanjay Sharma, "Lossless Image Compression through Huffman Coding Technique and Its Application in Image Processing using MATLAB," International Journal of Soft Computing and Engineering (IJSCE) ISSN: 2231-2307, Volume-7 Issue-1, March 2017

[8] Lin, Y., Wang, P., Lin, J., Ma, M., Liu, L., & Ma, L. (2015, December). Class-based delta-encoding for high-speed train data stream. In 2015 IEEE 34th International Performance Computing and Communications Conference (IPCCC) (pp. 1-8). IEEE.

[9] Akhter, Shahin, and M. A. Haque. "ECG compression using run length encoding." In 2010 18th European Signal Processing Conference, pp. 1645-1649. IEEE, 2010.

[10] Samteladze, N., & Christensen, K. (2012, October). DELTA: Delta encoding for less traffic for apps. In 37th Annual IEEE Conference on Local Computer Networks (pp. 212-215). IEEE..

[11] Sharma, Mamta. "Compression using Huffman coding." IJCSNS International Journal of Computer Science and Network Security 10, no. 5 (2010): 133-141.

[12] Bedruz, R. A., & Quiros, A. R. F. (2015, December). Comparison of Huffman Algorithm and Lempel-Ziv Algorithm for audio, image and text compression. In 2015 International Conference on Humanoid, Nanotechnology, Information Technology, Communication and Control, Environment and Management (HNICEM) (pp. 1-6). IEEE.

[13] https://en.wikipedia.org/wiki/Data_compression_ratio

[14] Ramesh, M. V. (2014). Design, development, and deployment of a wireless sensor network for detection of landslides. *Ad Hoc Networks*, *13*, 2-18.

[15] Shibu, N. S., Hanumanthiah, A., Rohith, S. S., Yaswanth, C. H., Krishna, P. H., & Pavan, J. V. S. (2018, December). Development of IoT Enabled Smart Energy Meter with Remote Load Management. In *2018 IEEE International Conference on Computational Intelligence and Computing Research (ICCIC)* (pp. 1-4). IEEE.

[16] Pathinarupothi, Rahul Krishnan, P. Durga, and Ekanath Srihari Rangan. "Iot-based smart edge for global health: Remote monitoring with severity detection and alerts transmission." *IEEE Internet of Things Journal* 6, no. 2 (2018): 2449-2462