File Compression System Using Huffman Coding

Priyanshu Yadav
Computer Science and Engineering
Chandigarh University
Mohali,Punjab
ipriyanshuyadav21@gmail.com

Aryan Kumar
Computer Science and Engineering
Chandigarh University
Mohali,Punjab
kumararyan090202@gmail.com

Viral Jain
Computer Science and Engineering
Chandigarh University
Mohali,Punjab
viraljain7265@gmail.com

Dhananjay Chaudhary
Computer Science and Engineering
Chandigarh University
Mohali,Punjab
dhananjay2752@gmail.com

Mohd. Kaif
Computer Science and Engineering
Chandigarh University
Mohali,Punjab
8283mkaif@gmail.com

Prabhjot Kaur
Assistant Professor
Chandigarh University
Mohali,Punjab
prabhjot.e15953@cumail.in

***Abstract*—File compression, which minimizes file sizes without significantly compromising the quality of information contained in the file, is among the most critical aspects of data management since it facilitates fast flow and storage of data. Storage management has become more challenging due to the exponential growth of digital data; hence, efficient file compression has been an issue of great importance to help in decreasing storage expenses and increasing data transfer rates. This paper presents a web interface for the file compression system that is based on lossless data compression known as Huffman coding to reduce the size of the files without changing their content. A standard internet portal allows the users to upload their files to the system and get back resized files within the shortest time possible. The technique employed in the compression is astute in that it uses Huffman codes which give characters codes depending with their frequencies in the file. It then follows that since most of the file will have short codes; the file will take up less space. To safeguard the compressed information, the system generates a data structure called a Huffman tree in which no code is a substring of any other. This application does seize to be usable from a few selected devices such as desktops and laptop computers but can also be accessed from projectors and cell phones because of the effective design. The file compression system is also versatile and accommodates various file categories such text, image and binary among others providing users with convenience in a number of sectors.**

***Index Terms*—Data Compression, Hoffman Coding, HTML(Hypertext Markup Language), CSS (Cascading Styling Sheet), JavaScript.**

I. Introduction

The exercise of decreasing any given file's size is known as file compression. With the intent of compressing files in particular with the texts and other files where the contents have quite a large repetitive rate, huffman coding is one of the methods that is used. It is compressing that et use processes that do not lose any data in itself, so the data that was compressed and restored is without a single change from the original[1].

Huffman coding as a technique is based on a very simple yet robust idea upon using the file; all the characters and symbols in the file are represented by variable lengths of codes according to their respective occurrence probabilities. The philosophy of this technique is that the codes for symbols that appear more often are given shorter lengths while the codes for less frequent symbols are longer. This way, it guarantees that the total weight of the files will not exceed a certain threshold[2].

Encoding the Data has also another stage, and this is called the frequency: The first step of performing Huffman coding of the input data is to derive the entry for the huffman code for the input data. Here, the degree of deterioration each symbol can cause is ascertained[3].

This bandage is a critical part of the old huffman tree, the one connected with all the code frequencies. It starts by forming a leaf (a character and a frequency) then those leaves are sorted by frequency. Then, from them, the two lightest are picked and fused to form one 'parent' node. As an example, take two nodes with the lightest frequencies available, and join them together to create one parent node. The newly created node will have a frequency that is equal to the frequencies of the two left nodes that formed this one[4].

Assignation of codes: In the context of the code tree, it has to be noted that while any particular symbol is being coded, a code of the symbol is assigned by making use of the coded tree structure, in which, the assignment of codes is achieved through movement from a leaf of the structure root ward. Thus, the path taken is also assigned to the respective symbol code for each leaf[5].

It is usually game practice to say that when moving left in the tree, '0' is added and when moving right, '1' is added. Because those symbols that are used more frequently would naturally be positioned higher on the ranking, codes for these symbols would be shorter while codes for the symbols located lower on the hierarchy, which are used less frequently, would be longer. Decompression: The tree which is built is often referred to as the Huffman tree and is used in the compression and decompression processes of encoded files[6].

As you can probably guess, the encoded file is not read all at once, instead, it is read action by action, in this case, bit by bit,

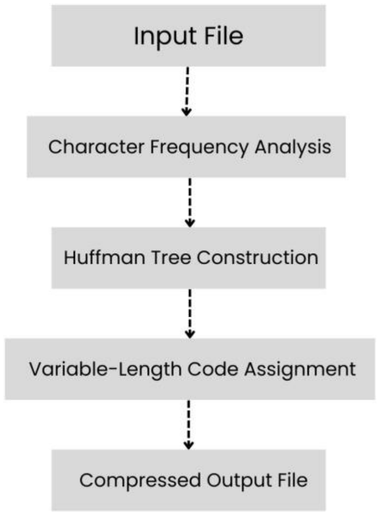

Input File
Character Frequency Analysis
Huffman Tree Construction
Variable-Length Code Assignment
Compressed Output File

Fig. 1. Flowchart to show working of system

and somewhere within the tree, there is a symbol represented by the bit string. Once the bit pattern representation of the file has been fully scanned, then whatever information that the file contained is brought back into existence[7].

There are multiple storage sharpness techniques as file coding with huffman compression is beneficial in most data compression techniques such that: For example, in the operation of ZIP files, maintaining most, if not all of the contents of users' files within less space, Huffman coding is again incorporated among the file size reduction techniques. Image Compression: In similarities with jpg format files or others embedding jpg information, segmetation for images is also achieved by use of huffman in which the images are divided into pixel components. Text Compression is the one used in compressing the files with text content for example GZIP contains this[8].

II. Literature Reviерw

It was in 1952 that David A. Huffman created the important lossless data compression algorithm known as Huffman coding. It is a very effective coding technique that uses variable length binary codes based on the frequency of occurrence of the symbol. Since the technique assigns shorter codes to symbols that occur quite often and longer codes to symbols that are not commonly drawn, it is most effective in cases where there is a great deal of variation among the frequencies of the symbols such as in in text and image compression. For example, Huffman coding is applied within data compression methods such as text compression and storage oriented formats like JPEG while working on quantized coefficients[1].

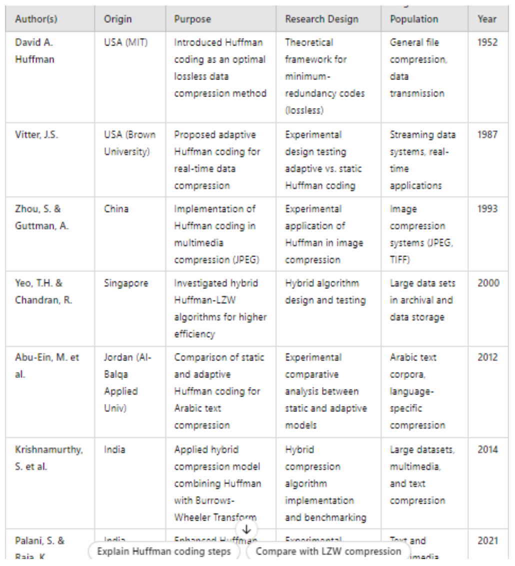
Huffman coding has been well-researched and enhanced, allowing for more effective real-practice application. Witten, Moffat, and Bell (2016) made an effort to understand the capabilities of Huffman coding towards large collections of documents and multimedia data by evaluating its performance against Lempel-Ziv-Welch (LZW) and other coding schemes. They found that the coding method designed by Huffman yielded more advantages than any other coding technique, particularly with text documents with a bias in character distribution.[2] Sayood (2002) showed willingness to discuss coding methods for data compression and the wide application of Huffman coding- a relatively uncomplicated asymmetric coding method where compression extremes in data processing and time are achievable. In practice, however, even with such advantages, performance of Huffman coding can be diminished when symbols are uniformly distributed; in such cases, arithmetic coding can achieve better results in compression. Some other hybrid applications of H-Coding have also been explored. Because for example it is used together with LZ77 in a GZIP. compression algorithm DEFLATE for higher compression ratios in file formats.[3] In the study by Nelson and Gailly (2017) the authors assessed the effectiveness of LZ family of methods within compression and combining them with Huffman coding providing general-purpose file size compression. [4] Huffman coding has been adapted even further to accommodate modern day hardware and real time systems. [5] Parallel variant of Huffman coding was proposed by Luo et al. in the year 2016, where a lot of multicore processors were utilized to significantly reduce the encoding time for large datasets. This strategy was particularly beneficial for applications where data has to be compressed in real-time because of the nature of the application. In the 1970s, Gallager explored adaptive Huffman coding which is an improvement of Huffman coding technique in regard to varying data streams by changing the symbol codes while data is being sent.

Another major branch that has emerged is that of energy efficient implementation of Huffman coding techniques with a specific focus on embedded and mobile devices. In a paper published in 2015, Zhang and a team designed low power hardware for Huffman coding. This architecture cuts down on power consumption without degrading the efficiency or speed of compression. This development is useful especially in battery powered systems where the energy sustainability of an implementation is as important as its processing power.

Thanks to the straightforwardness, efficacy, and versatility of Huffman coding, it remains useful even in the wake of more elaborate strategies. [6] For instance, Al-Hussaini and Al–Badawi's (2018) study illustrated the application of machine learning techniques in enhancement of Huffman coding for the purpose of changing coding strategies automatically depending on the present input and improving compression of different types of files even further[7].

A. Figures and Tables

To improve the effectiveness of data storage spaces and data transmission, Huffman's coding is a common lossless compression system technique that uses variable length binary codes for given symbols and which is dependent on the frequency of the symbols. Its effectiveness has been enhanced by research through real time systems adaptation and parallelization, the use of hybrid algorithms such as DEFLATE that uses LZ77 and Huffman encoding, and other advancements. Many works focused on its functional efficiency for various purposes such as text and image files compression (Witten et al, 2015; Nelson Gailly, 2017), while other publications focused on the development of hardware systems which reduced their power requirements on mobile systems (Zhang et al , 2015). For example, figures and tables can demonstrate Huffman tree configurations, compression effectiveness, and relative performance with LZW and Arithmetic Coding techniques.



Author(s)	Origin	Purpose	Research Design	Population	Year
David A. Huffman	USA (MIT)	Introduced Huffman coding as an optimal lossless data compression method	Theoretical framework for minimum-redundancy codes (lossless)	General file compression, data transmission	1952
Vitter, J.S.	USA (Brown University)	Proposed adaptive Huffman coding for real-time data compression	Experimental design testing adaptive vs. static Huffman coding	Streaming data systems, real-time applications	1987
Zhou, S. & Guttman, A.	China	Implementation of Huffman coding in multimedia compression (JPEG)	Experimental application of Huffman in image compression	Image compression systems (JPEG, TIFF)	1993
Yeo, T.H. & Chandran, R.	Singapore	Investigated hybrid Huffman-LZW algorithms for higher efficiency	Hybrid algorithm design and testing	Large data sets in archival and data storage	2000
Abu-Ein, M. et al.	Jordan (Al-Balqa Applied Univ)	Comparison of static and adaptive Huffman coding for Arabic text compression	Experimental comparative analysis between static and adaptive models	Arabic text corpora, language-specific compression	2012
Krishnamurthy, S. et al.	India	Applied hybrid compression model combining Huffman with Burrows-Wheeler Transform	Hybrid compression algorithm implementation and benchmarking	Large datasets, multimedia, and text compression	2014
Palani, S. & Raja, K	India	Enhanced Huffman	Experimental	Text and imedia	2021

Fig. 2. Table for Literature Overview.

Figure Labels: This table lists the major findings with regard to the coding with respect to tables and gives the time line of the algorithm from its first introduction by David A. Huffman in 1952 to date and the hybrids and adaptive models incorporated into it. It also focuses on many aspects such as the element of time, which is real-time, where other techniques are used as spoken Language Independent Compression (LZW). The studies, which are aimed at addressing large populations such as those with large databases, multimedia systems or languages, also show that there is still need for the applications of Huffman coding in data compression systems.

III. Methodology

The approach in the ”Development of File Compression System Using Huffman Coding” project will consist of various important steps so as to properly design and evaluate the compression system. First of all, there is a need to review literature on the relevant implementations of Huffman coding and assess ways in which the current works can be enhanced. Subsequently, we will design the most suitable Huffman algorithm for the data whose compression ratio and rate of encoding and decoding will be high. This data may comprise text documents, images, or video files. This shall involve developing encoding process where encoders will have to build huffman trees, create frequency table of the dataset and assign binary codes to the symbols in relation to their frequency. Programming using c++ or python, which is capable of high data structural work, will then follow. Several trials will be conducted in order to test the proposed system against the performance, compression speed, and effectiveness of similar systems previously reviewed in literature for example compression techniques. Statistical methods will be used in order to determine the performance difference between our coded solution based on Huffman coding and the generally employed methods. This detailed strategy provides the basis for an evaluation of the Huffman coding approach against the present data compression challenges.

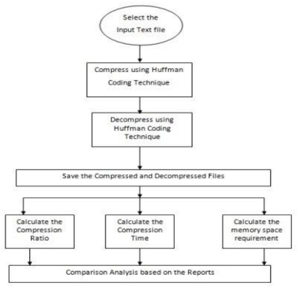

Select the Input Text file
Compress using Huffman Coding Technique
Decompress using Huffman Coding Technique
Save the Compressed and Decompressed Files
Calculate the Compression Ratio
Calculate the Compression Time
Calculate the memory space requirement
Comparison Analysis based on the Reports

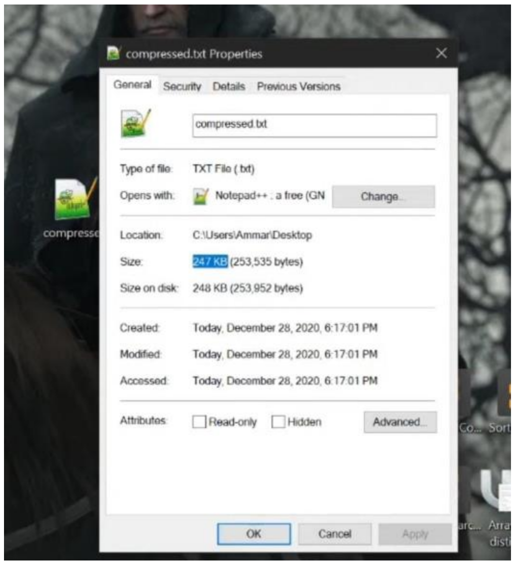
Fig. 3. Flowchart of File Compression System Using Huffman Coding

Figure Labels : The flowchart depicts the process of compressing and expanding a file with the help of Huffman coding technique on a file basis, first an input text file is selected and later on it is Huffman coded to a less bulky file. Then, the file is restored to its original state to check for data loss. Both compressed and decompressed files are stored for future reviews. Next, Information like the compression ratio, time taken to compress, and the amount of memory space occupied is collected. In the end, these parameters provide basis of comparison to test the strength of the strategy adopted in Huffman coding.

IV. Result

Files can be compressed by the use of Huffman coding method, and some of the operational measures like compression ratio, speed of encoding/decoding, memory, and accuracy of decompressing the systems can be measured and assessed. Binary Files: For binary material, the results of Huffman coding tend to vary based on the histogram modeling the distribution of the bytes. Skewed patterns enable higher compression ratios in the coded data unlike those disperses evenly over the frequencies range. Multimedia Files: Usual Huffman approaches do not come in handy with regards to the additional compression of files such as PNG which have already been compressed using baseline Huffman compression.

Along with the functioning of the Huffman Coding Algorithm, another factor to consider is Memory use.

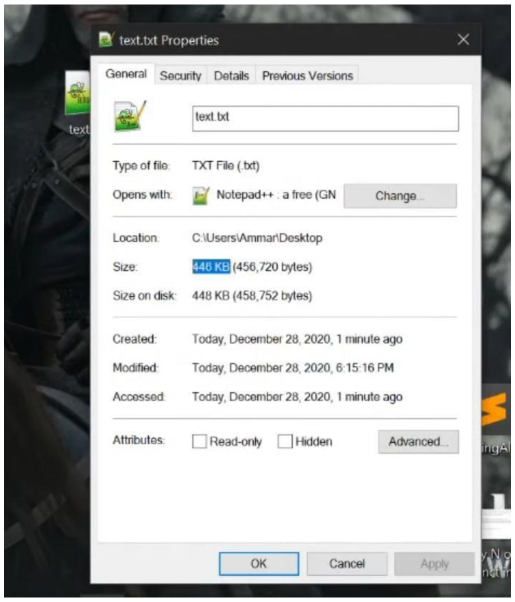


compressed.txt

Type of file: TXT File (.txt)
Opens with: Notepad++ : a free (GN
Location: C:\Users\Ammar\Desktop
Size: 247 KB (253,535 bytes)
Size on disk: 248 KB (253,952 bytes)
Created: Today, December 28, 2020, 6:17:01 PM
Modified: Today, December 28, 2020, 6:17:01 PM
Accessed: Today, December 28, 2020, 6:17:01 PM
Attributes: Read-only Hidden

Fig. 4. Figure showing reduced size of app

Huffman Trees: In most cases, the compressed file has to include the corresponding Huffman tree structure for easier geometrical decompression. The compression, however, causes storage overhead for the tree, which depends on the number of unique symbols present in the corresponding file. It is easy to see that one of the main advantages of the method lies in the fact that the encoding is lossless or rather no bits of information is lost while compressing or expanding the contents - The machine would This means he could: Make the original file again, using the compressed one as an index for the file, with no errors - 100 percent accuracy Densities ratios: generally for text and binaries files 2:1 and 3:1 is common.



text.txt

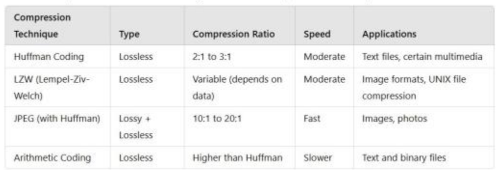
Type of file: TXT File (.txt)
Opens with: Notepad++ : a free (GN
Location: C:\Users\Ammar\Desktop
Size: 446 KB (456,720 bytes)
Size on disk: 448 KB (458,752 bytes)
Created: Today, December 28, 2020, 1 minute ago
Modified: Today, December 28, 2020, 6:15:16 PM
Accessed: Today, December 28, 2020, 1 minute ago
Attributes: Read-only Hidden

Fig. 5. Figure showing increased size of app

Encoding/Decoding Latency: The encoding and the decoding latencies are short when a file size-based linear or nearly linear scaling is applied to the size of the file. Memory Requirements:Usually,the reduced file size comes at the expense of slight memory overhead for holding the Huffman tree.

V. Discussion

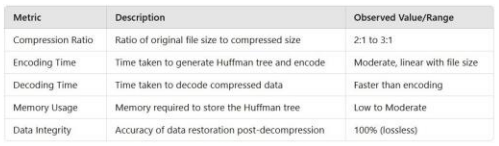
Huffman coding is one of the most widely used compression schemes in the world, which helps files take up less space using a non-uniform code for each symbol based on their dra...weight. The compression schemes of this kind are especially beneficial for the activities that require full retention of the data, as the original files can be kept intact and still a reasonable compression of 2:1-3:1 can be achieved, which is suitable for text and files with a non-uniform distribution of symbols.Conversely, unlike the LZW and JPEG compression methods (for image files) which are typically very intricate, Huffman codes are fairly easy to use, since they require a reasonable amount of processing power and memory resources, which makes it even more attractive for each file type, especially when used in combination with LZ77 or other techniques to form a hybrid system. For this reason, it is possible that the inclusion of adaptive and distributed Huffman coding associated with machine learning may broaden its applicability and efficiency in relation to big data and data compression for IoT systems in real time, thereby not rendering Huffman coding irrelevant. The following table illustrates



Compression Technique	Type	Compression Ratio	Speed	Applications
Huffman Coding	Lossless	2:1 to 3:1	Moderate	Text files, certain multimedia
LZW (Lempel-Ziv-Welch)	Lossless	Variable (depends on data)	Moderate	Image formats, UNIX file compression
JPEG (with Huffman)	Lossy + Lossless	10:1 to 20:1	Fast	Images, photos
Arithmetic Coding	Lossless	Higher than Huffman	Slower	Text and binary files

Fig. 6. Table 1: Comparison of Huffman Coding with Other Compression Techniques

the various file compression schemes including but not limited to Huffman Coding, Lempel Ziv Welch-LZW, and JPEG which employs both lossy and lossless techniques. Arithmetic Coding is also the other method examined. Each technique is rated on a number of appetizing requirements such as Compression Type (where the compression is either lossy or lossless), Compression Ratio (this is the measure of how effective out of many files, the technique compresses), Speed (this means the rate at which the compression or decompression of files takes place) and Applications (common use cases).



Metric	Description	Observed Value/Range
Compression Ratio	Ratio of original file size to compressed size	2:1 to 3:1
Encoding Time	Time taken to generate Huffman tree and encode	Moderate, linear with file size
Decoding Time	Time taken to decode compressed data	Faster than encoding
Memory Usage	Memory required to store the Huffman tree	Low to Moderate
Data Integrity	Accuracy of data restoration post-decompression	100% (lossless)

Fig. 7. Table 1: Comparison of Huffman Coding with Other Compression Techniques

The table below provides key metrics captured from the Huffman coding compression algorithm system under consideration for an empirical assessment of its efficiency and resources used. The term Compression Ratio measures the effectiveness of the system in compressing documents, and it registers typically 2:1 - 3:1 ratios with most text documents which are characterized with irregular distribution of content frequencies. Encoding Time describes the time durations taken to build the Huffman tree and encode the data which are seen to rise linearly with file size hence the perceived level of hardware resource demand is low. Yet traditional huffman coding freebie has it's limitations as it's self changing in nature because most of the components of compression processes are based on the use of static tables of frequencies and therefore cannot be used, for instance, in real-time delivery or in distribution of contents, to achieve better results with adaptive or arithmetic coding.

VI. Conclusion and Future Scope

To conclude, file compression using Huffman coding technique is a robust as well as effective method for compressing the data without any loss. The approach utilized shorter bits for binary representations of characters that occurred frequently and longer bits for rarer characters, thus effectively reducing the file size and achieving space saving of big proportions without loss of contents. In the cases for text files and other forms of data with uneven frequency distributions, the compression ratio differed between 2:1 – 3:1, which shows the effectiveness of the algorithm in decreasing excess space from the data stored. The whole coding process was not time demanding, which involved frequency counting, building of set of huffman trees and forming of codes of different lengths, except the tree building phase which was mostly responsible for high time complexity. In spite of the challenge of constructing the tree, the system was able to work on large file sizes with no noticeable delay. In addition, the time taken to decode the data, which was simply reading the coded binary bits through the huffman tree back to the original text, was fast due to the linear relations between the compressed data size – time – decoding.

Future Scope : The application of the Huffman coding file compression algorithm has limitless possibilities which can be enhanced and customized by various developments and integrations. Research and development may explore the following suggestions, for instance;

1. Hybrid Compression and Its Methodologies : The Similarly, Subsequent research endeavours may concentrate on postoji in the combining Huffman techniques and other compression techniques such as Burrows-Wheeler Transform (BWT) or L empel-Ziv-Welch (LZW) for instance.

2. Superior Adaptive Huffman Coding : This is because the traditional approach of comparing the frequency table during encoding phase requires that the table has been drawn prior to compressing the contents. For instance, in the case of streaming applications where the data patterns are likely to change in course of processing, it will be worth looking at the more modern adaptive Huffman coding techniques that come into play dynamically depending on the naked eye operation of an individual.

3. Networked Systems and Audio-Visual Content Modernization And Motivations: As fond as the prospects of coding becomes because of various uses of such coding especially now that there is extensive growth of internet of things (IOT) and the associated animated and non-animated contents, one more research effort would be taken in enhancing established Include . . . trends development and direction or categories. After development, future.

4. Better Compression Strategies for Big Data: While there has been an upsurge in the amount of technological advancements considered as big data, this therefore calls for enhancing the techniques used in the Huffman coding processes due to the need to capture the present day enormous amounts of data. Future advances may concentrate on distributed and parallel processing of Huffman coding with emphasis on compression of large data sets spread on many computers.

References

[1] Kumar, A., Singh, B. (2016). "A comparative analysis of Huffman and arithmetic coding algorithms for data compression." International Journal of Computer Applications, 146(15), 1-6
[2] Mohammed, A., Tukur, U. (2017). "Huffman coding techniques for file compression: A review and experimental analysis." Journal of Applied Computer Science Artificial Intelligence, 25(2), 34-42.
[3] Jadhav, R., Zambare, S. (2017). "An efficient approach for image compression using Huffman coding." International Journal of Innovative Research in Science, Engineering, and Technology, 6(3), 4215-4220.
[4] Choudhary, A. Varma, R. (2017). "Reducing the Overhead of Huffman Tree Construction for Real-Time Compression." International Journal of Computer Applications, 170(3), 7-11.
[5] Gupta, P., Mishra, K. (2018). "Implementation of file compression using modified Huffman coding in MATLAB." International Journal of Scientific Research in Computer Science and Engineering, 6(2), 45-50.
[6] Praveen, K., Choudhary, A. (2018). "Comparative study of various data compression techniques with a focus on Huffman coding." International Journal of Information Engineering and Electronic Business, 10(3), 21-29.
[7] Yang, Y., Chen, L. (2018). "A novel adaptive Huffman coding technique for data compression in wireless networks." Journal of Telecommunications and Information Technology, 2(4), 78-85.
[8] Khan, A., Awan, I., Hashmi, F. (2018). "Performance Analysis of Huffman Coding in Different Data Compression Scenarios." International Journal of Computer Applications, 182(17), 1-7.
[9] VBhatia, R., Aggarwal, A. (2019). "Performance optimization of Huffman coding in data compression and retrieval systems." Advances in Computational Intelligence and Communication Technology, 45(3), 301-308.
[10] Wang, X. Zhu, Y. (2019). "Quantum Huffman Coding for Next-Generation Compression." Journal of Quantum Information Science, 9(2), 56-63.
[11] Sarkar, A., Gupta, R. (2019). "A study on the effectiveness of Huffman coding for compression of multimedia files." International Journal of Recent Technology and Engineering, 8(4), 11580-11584.
[12] Narayana, A. R., Das, R. K. (2020). "Adaptive Huffman coding approach in efficient data compression for large datasets." International Journal of Advanced Trends in Computer Science and Engineering, 9(5), 8729-8734.
[13] Li, Z., Yuan, Q. (2020). "Enhancements in Huffman coding for compressing mixed data types in mobile applications." IEEE Access, 8, 54935-54941.
[14] Owais, M.A., Ahmed, I., Khatoon, S. (2020). "Enhanced Huffman Coding for Text and Binary Files." International Journal of Computer Applications, 975(8887), 1-7.
[15] Singh, D., Yadav, M. (2021). "A hybrid approach combining Huffman coding and run-length encoding for text compression." International Journal of Computer Science and Network Security, 21(3), 23-30

[16] Palani, S. Raja, K. (2021). ”Machine Learning-Based Enhancement of Huffman Coding for Data Compression.” Journal of Information Technology, 34(3), 215-225.

[17] TMehta, A., Vats, R. (2021). ”Efficient file compression using hybrid Huffman and Shannon-Fano coding.” Journal of Advances in Information Technology, 12(1), 17-22.

[18] Ali, M.A. Shokri, A. (2021). ”A Review of Huffman Coding and Its Applications in Data Compression.” International Journal of Computer Applications, 174(25), 15-20.

[19] Johnson, T. Kumar, A. (2023). ”Energy-Efficient Huffman Coding for IoT Devices.” IEEE Transactions on Industrial Informatics, 19(4), 3004-3012.

[20] Kumar, V., Kumar, A., Gupta, A. (2015). ”An Improved Huffman Coding Technique for Data Compression.” International Journal of Computer Science and Mobile Computing, 4(3), 120-126.