RESEARCH ARTICLE OPEN ACCESS

A Study on Data Compression Using Huffman Coding Algorithms

D.Jasmine Shoba [1], Dr.S.Sivakumar [2]
Research Scholar [1], Assistant Professor [2]
Department of Computer Science [1]
Department of Computer Applications [2]
Thanthai Hans Roever College, Perambalur
Tamil Nadu – India

ABSTRACT

Data reduction is one of the data preprocessing techniques which can be applied to obtain a reduced representation of the data set that is much smaller in volume, yet closely maintains the integrity of the original data. That is, mining on the reduced data set should be more efficient yet produce the same analytical results. Data compression is useful, where encoding mechanisms are used to reduce the data set size. In data compression, data encoding or transformations are applied so as to obtain a reduced or compressed representation of the original data. Huffman coding is a successful compression method used originally for text compression. Huffman's idea is, instead of using a fixed-length code such as 8 bit extended ASCII or DBCDIC for each symbol, to represent a frequently occurring character in a source with a shorter codeword and to represent a less frequently occurring one with a longer codeword.

Keywords:- Huffman Coding, Data Comparison, Data decompression

I. INTRODUCTION

Data compression is one of the most widespread applications in computer technology. Algorithms and methods that are used depend strongly on the type of data, i.e. whether the data is static or dynamic, and on the content that can be any combination of text, images, numeric data or unrestricted binary data. The compression and decompression techniques are playing main role in the data transmission process. The best compression techniques among the three algorithms have to be analyzed to handle text data file. This analysis may be performed by comparing the measures of the compression and decompression

II. RELATED WORKS

Both the JPEG and JPEG 2000 image compression standard can achieve great compression ratio, however, both of them do not take advantage of the local characteristics of the given image effectively. Here is one new image compression algorithm proposed by Huang [2], it is called Shape Adaptive Image Compression, which is abbreviated as SAIC. Instead of taking the whole image as an object and utilizing transform coding, quantization, and entropy coding to encode this object, the SAIC algorithm segments the whole image into several objects, and each object has its own local characteristic and color. Because of the high correlation of the color values in each image segment, the SAIC can achieve better compression ratio and quality than conventional image compression algorithm.

We also briefly introduce the technique that utilizes the statistical characteristics for image compression. The new image compression algorithm called Shape- Adaptive Image Compression, which is proposed by Huang [1], takes advantage of the local characteristics for image compaction. The SAIC compensates for the shortcoming of JPEG that regards the whole image as a single object and do not take advantage of the characteristics of image segments.

The term refers to the use of a variable-length code table for encoding a source symbol (such as a character in a file) where the variable-length code table has been derived in a particular way based on the estimated probability of occurrence for each possible value of the source symbol. Huffman coding is based on frequency of occurrence of a data item. The principle is to use a lower number of bits to encode the data that occurs more frequently [3].

The DCT separates the image into different frequencies part. Higher frequencies represent quick changes between image pixels and low frequencies represent gradual changes between image pixels. In order to perform the DCT on an image, the image should be divided into 8 × 8 or 16 × 16 blocks [4].

It is based on the idea to replace a long sequence of the same symbol by a shorter sequence. The DC coefficients are coded separately from the AC ones. A DC coefficient is coded by the DPCM, which is a lossless data compression technique.

While AC coefficients are coded using RLC algorithm. The DPCM algorithm records the difference between the DC coefficients of the current block and the previous block [5].

III. RESEARCH METHODOLOGY

This section describes the research Methodology generated by the proposed system. We have taken different dataset and conducted various experiments to determine the performance of the proposed system.

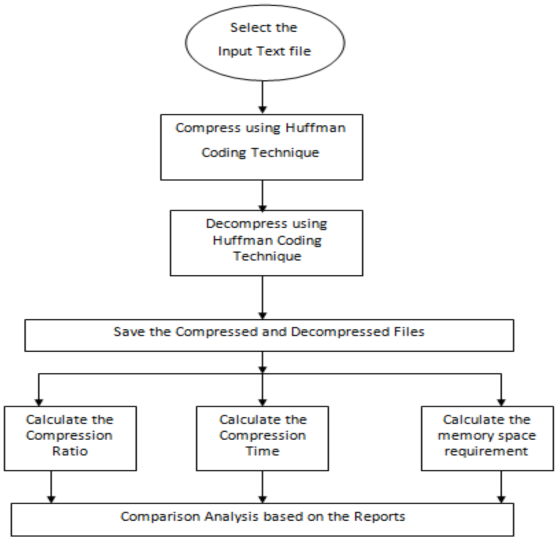

Select the Input Text file
Compress using Huffman Coding Technique
Decompress using Huffman Coding Technique
Save the Compressed and Decompressed Files
Calculate the Compression Ratio
Calculate the Compression Time
Calculate the memory space requirement
Comparison Analysis based on the Reports

Figure 1. Architecture of research methodology

3.1 Huffman Coding Technique

A more sophisticated and efficient lossless compression technique is known as "Huffman coding", in which the characters in a data file are converted to a binary code, where the most common characters in the file have the shortest binary codes, and the least common have the longest [9]. To see how Huffman coding works, assume that a text file is to be compressed, and that the characters in the file have the following frequencies:

A: 29 B: 64 C: 32 D: 12 E: 9 F: 66 G: 23

In practice, we need the frequencies for all the characters used in the text, including all letters, digits, and punctuation, but to keep the example simple we'll just stick to the characters from A to G. The first step in building a Huffman code is to order the characters from highest to lowest frequency of occurrence as follows:

66	64	32	29	23	12	9
F	B	C	A	G	D	E

First, the two least-frequent characters are selected, logically grouped together, and their frequencies added. In this example, the D and E characters have a combined frequency of 21:

```
      |
  +--+--+
  |   21  |
  |          |
         66        64        32        29         23        12        9
         F          B        C          A        G          D        E
```

This begins the construction of a "binary tree" structure. We now again select the two elements the lowest frequencies, regarding the D-E combination as a single element. In this case, the two elements selected are G and the D-E combination. We group them together and add their frequencies. This new combination has a frequency of 44:

```
            |
    +------+-----+
             |     44          |
             |               |
    |        +--+--+
             |      |  21  |
             |       |        |
         66        64        32        29           23      12        9
         F          B         C          A           G       D        E
```

We continue in the same way to select the two elements with the lowest frequency, group them together, and add their frequencies, until we run out of elements.

In the third iteration, the lowest frequencies are C and A and the final binary tree will be as follows:

```
                     |
         +-------------------------+----------------------------+
         | 0                                                          |1
         |                                                            |
         |                                   +-------------------+-----------------+
         |                                   | 0                                     | 1
         |                                   |                                       |
         |                                   |                            +----------+----------+
         |                                   |                            | 0                       | 1
         |                                   |                            |                         |
+-------+-------+              +-------+-------+           |                   +--+--+
| 0                   |1                   | 0                  | 1                 |                  | 0         | 1
|                     |                    |                    |                   |                   |            |
F                     B                    C                    A                   G                  D           E
```

Tracing down the tree gives the "Huffman codes", with the shortest codes assigned to the characters with the greatest frequency:

F: 00
B: 01
C: 100
A: 101
G: 110
D: 1110
E: 1111

The Huffman codes won't get confused in decoding. The best way to see that this is so is to envision the decoder cycling through the tree structure, guided by the encoded bits it reads, moving from top to bottom and then back to the top. As long as bits constitute legitimate Huffman codes, and a bit doesn't get scrambled or lost, the decoder will never get lost, either.

3.2 Algorithm: Huffman encoding

Step 1: Build a binary tree where the leaves of the tree are the symbols in the alphabet.
Step 2: The edges of the tree are labeled by a 0 or I.
Step 3: Derive the Huffman code from the Huffman tree.
INPUT : a sorted list of one-node binary trees (t_1 , t_2,... , t_n) for alphabet

(S_l , . - . , S_n) with frequencies (W_l , . . . , W_n)
OUTPUT : a Huffman code with n code words

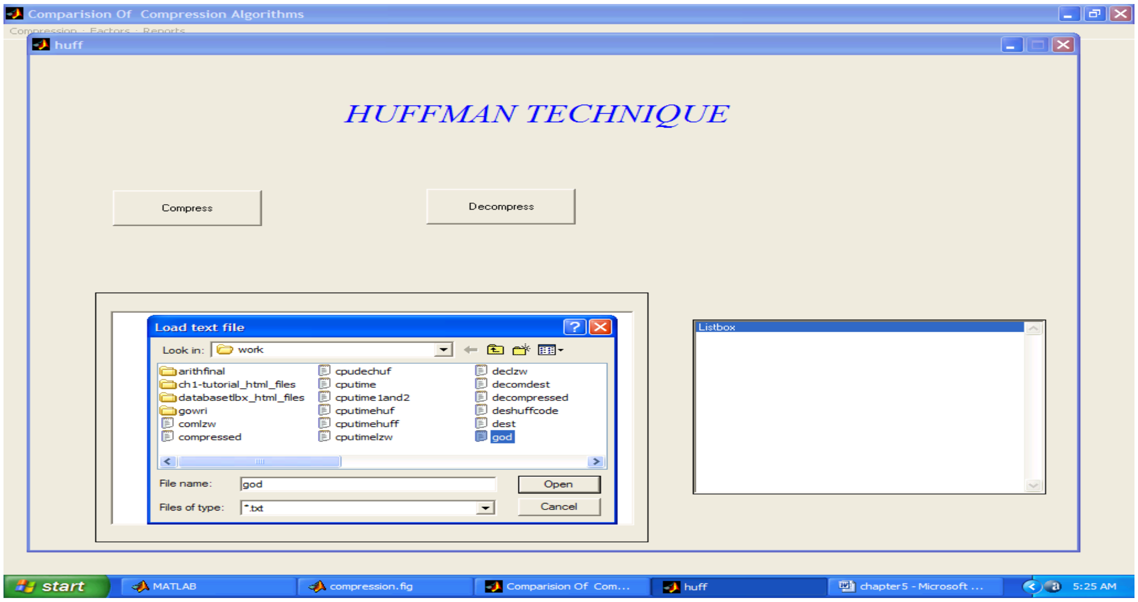
3.3 Algorithm: Huffman decoding

Step 1: Read the coded message bit by bit. Starting from the root, we traverse one edge down the tree to a child according to the bit value. If the current bit read are 0 moves to the left child, otherwise, to the right child.
Step 2: Repeat this process until reach a leaf. If a leaf is reached, decode one character and restart the traversal from the root.
Step 3: Repeat this read-and-move procedure until the end of the message.
INPUT : a Huffman tree and a 0-1 bit string of encoded message
OUTPUT: decoded string

IV. RESULT AND DISCUSSION

This system consists of three major modules namely, compression process, comparison factors and Reports. The first module compression process is used to compress the given input text file and decompress the compressed file. The second module comparison factors is used to compare the Huffman Coding Technique based on the calculated metrics namely compression ratio, Transmission time and memory utilization. The next module Reports is used to view the various reports about the analysis process.

The input file which is to be compressed using Huffman Technique is selected from the specific location and displayed in the List box as shown in the following figure 2



File name: god
Files of type: *.txt

Figure 2. Huffman Technique Running window

4.1 Text Decompression

The text decompression is taking the compressed file and expanding it into its original form. The decompression part provides the original file as output. That is converting an input data stream into another data stream that has an original size of input file. In the decompression process, the decompression process is activated through decompression button using Huffman coding. The decompressed text file is stored in a separate file and it is shown in list box as shown in Figure .

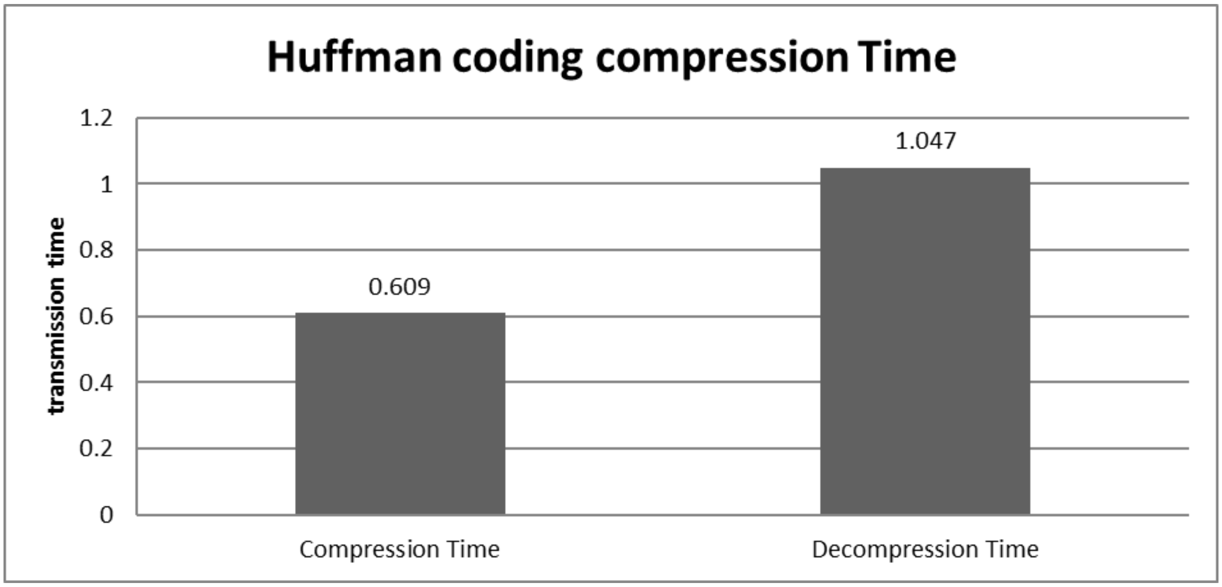


Figure 3. Time efficiency

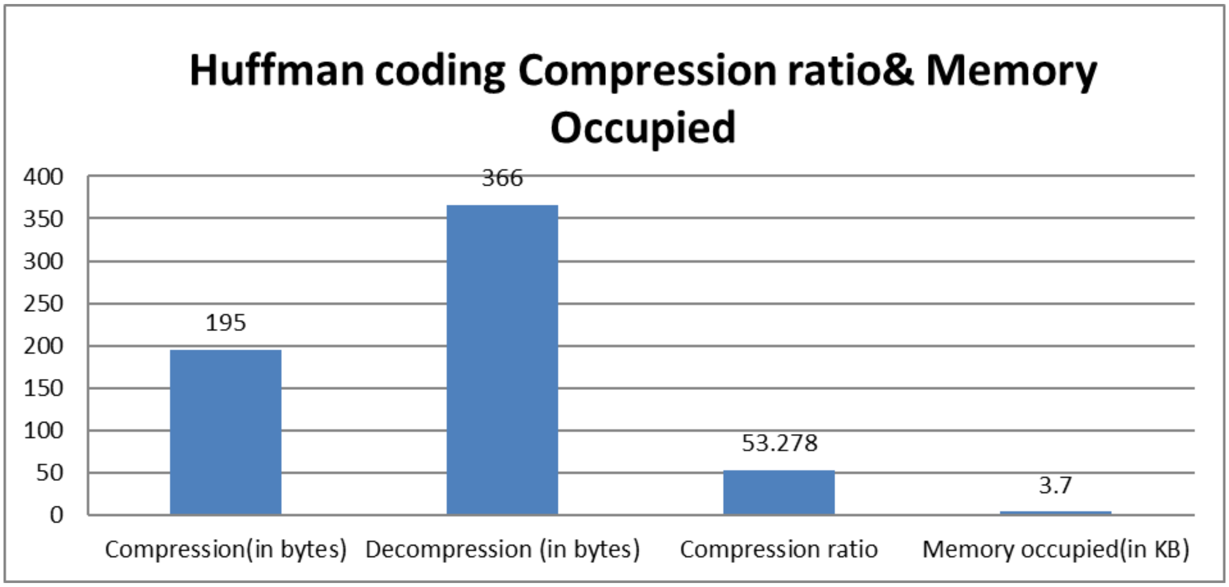


Figure 4. Compression ratio & Memory Occupied

The above graph is shows the comparison of compression ratios and memory occupied of the existing systems and the proposed system. In the graph, the horizontal axis represents the length of input string in bytes and vertical axis represents the Compression Ratio in percentage.

V. CONCLUSION

In this Paper Huffman coding compression techniques are compared. This system uses three metrics such as compression ratio, transmission time and memory utilization to compare and analyze the results. It is found that the Huffman coding technique shows the compression performance better techniques. Surely this technique will open a scope in the field of text compression where every bit of information is significant.

REFERENCES

[1] Jian-Jiun Ding and Tzu-Heng Lee, "Shape-Adaptive Image Compression", Master's Thesis, National Taiwan University, Taipei, 2008

[2] Jian-Jiun Ding and Jiun-De Huang, "Image Compression by Segmentation and Boundary Description", Master's Thesis, National Taiwan University, Taipei, 2007.

[3] Sharma, M.: 'Compression Using Huffman Coding'. International Journal of Computer Science and Network Security, VOL.10 No.5, May 2010.

[4] O ’ Hanen, B., and Wisan M. : ' JPEG Compression'. December 16, 2005.

[5] Blelloch, G.: 'Introduction to Data Compression'. Carnegie Mellon University, September , 2010.