RESEARCHERID
THOMSON REUTERS
[COTII 2019]
ISSN 2348 – 8034
Impact Factor- 5.070

GLOBAL JOURNAL OF ENGINEERING SCIENCE AND RESEARCHES

SATONIC COMPRESSION TECHNIQUE COMPRESSION TECHNIQUE FOR LOSSLESS DATA COMPRESSION

Dr. Shalini Lamba[*1], Archit Verma[2] & Akhand Chaurasiya[3]
[*1]Head of Computer Science Dept, National PG College
[2&3]Bachelor of Computer Application, National PG College

ABSTRACT

With the new innovative concept of big data conquering the markets at a rapid pace, the role of data is becoming more and more crucial. Huge amount of data is stored and transferred across the internet on the daily basis. This data needs to maintain the security, integrity and availability. One of the ways to ensure this is the compression and encryption of data. Compression involves reduction of redundant data to minimize the file size making it easier to transfer and store. This paper gives a comparison between the existing file compression techniques and their limitations. Further a new approach to file compression algorithms is given and it is termed as – "Satonic Compression Technique"

Keywords: *Huffman, Shannon, Lossy Compression, Lossless Compression, Run Length, Arithmetic Encoding*

I. INTRODUCTION

Data Compression is a procedure by which a record (Text, Audio, and Video) can be packed, to such an extent that the first document might be completely recouped with no loss of real data. This procedure might be valuable on the off chance that one needs to spare the storage room. The trading of compacted document over web is simple as they can be transferred or downloaded a lot quicker. Information compression has essential application in the zone of document stockpiling and dispersed framework. Information compression is utilized in sight and sound field, content archives, and database table.[1] Data Compression systems can be arranged in two noteworthy structures:

Lossy Data Compression
In lossy compression the decompressed information got is unique in relation to the first information. These compression methods are temperate and less demanding to actualize. They are utilized in applications where some loss of data is satisfactory. Lossy compression isn't utilized for information, for example, content-based records and programming, since they have to keep all their data. Lossy Compression is just powerful with media components that can at present 'work' without all their unique information.[10]

Lossless Data Compression:
In this strategy, information is compacted with no loss of information. Lossless compression[3] calculations decrease document measure with no misfortune in picture quality. At the point when the record is spared it is compacted, when it is decompressed (opened) the first information is recovered. The record information is just incidentally 'discarded', so the document can be exchanged. This sort of compression can be connected to designs as well as to any sort of PC information, for example, spreadsheets, content records, and programming applications. This is the reason the term lossless is utilized, as no information is lost. While the upside of this is it keeps up quality the fundamental drawback is it doesn't decrease the record estimate as much as lossy compression. Different lossless information compression calculation has been proposed and utilized. Some of the principle methods are Shannon-Fano, Huffman Coding, Run Length Encoding, and Arithmetic Encoding.[11]

II. LOSSLESS COMPRESSION TECHNIQUES

Huffman Coding:
A twofold code tree is created in Huffman Coding for given info. A code length is developed for each image of info information dependent on the likelihood of event. Huffman codes are a piece of a few information arranges as ZIP, GZIP and JPEG. Typically the coding is gone before by methodology adjusted to the specific substance. For

instance the wide-spread DEFLATE calculation as utilized in GZIP or ZIP beforehand forms the lexicon based LZ77 compression. It is a refined and productive lossless information compression system. The image with most astounding likelihood has the briefest double code and the image with least likelihood has the longest paired code.[5]

Shannon Coding:
This is one of a most punctual system for information compression that was concocted by Claude Shannon and Robert Fano in 1949.In this method, a parallel tree is produced that speak to the probabilities of every image happening. The images are requested in a way with the end goal that the most continuous images show up at the highest point of the tree and the most improbable images show up at the base.[7]

Arithmetic Coding:
Arithmetic encoding is the most dominant compression procedures. This converts the entire input data into a single floating point number.[8]

Run Length Coding:
Information regularly contains groupings of indistinguishable bytes. Supplanting these rehashed byte arrangements with the quantity of events, a decrease of information can be accomplished. This is known as Run-length Encoding. RLE fundamentally packs the information by lessening the physical size of a continuing series of characters. This continuing string is known as a run which is commonly encoded into two bytes where the primary byte speaks to the all out number of characters in the run and is known as the run tally and it replaces keeps running of at least two of a similar character with a number which speaks to the length of the run which will be trailed by the first character and single characters are coded as keeps running of 1. The clear (space) character in content is such an image; single spaces or matches of spaces are disregarded. RLE is valuable where excess of information is high or it can likewise be utilized in blend with other compression procedures moreover.[9]

III. SATONIC COMPRESSION CODING

Satonic compression coding is an improved version of the Shannon / Huffman Coding. The improvement is on the basis of both time complexity and space complexity. Satonic compression coding is currently only for textual data compression only.

Basic Principle:
The basic principle behind the algorithm is-

Objects take less space when double or triple folded. These objects can be recovered completely by unfolding them.

Algorithm:
As its principle indicates, Satonic Coding involves folding of data in 4 layers so that they take up less space. Suppose we have a string of data:

Satonic compression coding
In order to compress the text, we follow the following steps-

Step 1: Count the repeated content of the data i.e. the redundant characters that occur in the string.
In the above string we can see that –

1. **S, O, N, I, C, E** occurs 3 times each.
2. **T** occurs twice.
3. **A, M, P, R, H, Q, U** occurs only once.

Step 2:
Now group these characters in descending order of redundancy. For example-
S,O,N,I,C,E,T,A,M,P,R,H,Q,U

Step 3:
Now arrange these characters in a 4 row grid as shown below.

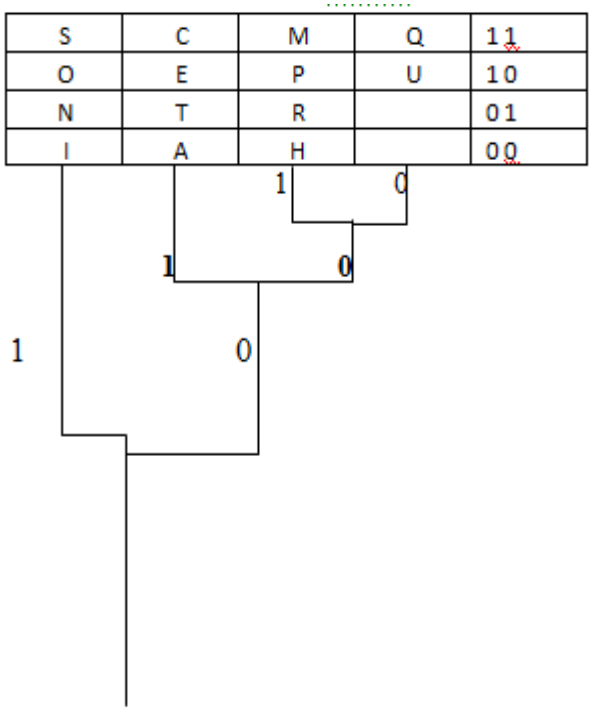
S	C	M	Q
O	E	P	U
N	T	R	
I	A	H	

Step 4:
Now serialize the rows of this grid in binary as 00, 01, 10, 11 as shown below.

S	C	M	Q	1 1
O	E	P	U	1 0
N	T	R		0 1
I	A	H		0 0

Step 5:
Now group the columns in the following manner-
And number each of the branch as 0 and 1.

ISSN 2348 – 8034
Impact Factor- 5.070



Step 6:
Now we have to write the binary codes of each character of the string. To find out the binary code of each character we have to write the consequent branch bits and then the row number.

Example-

1. S = 111
2. O = 110
3. N = 101
4. I = 100
5. C = 0111
6. E = 0110
7. T = 0101
8. A = 0100
9. M = 00111
10. P = 00110
11. R = 00101
12. H = 00100
13. Q = 00011
14. U = 00010

Step 7: Now we represent the original string in the form of bit codes we have obtained –
So now –

SATONIC COMPRESSION TECHNIQUE

Will be written as –
111 0100 0101 110 101 100 0111 0111 110 00111 00110 00101 0110 111 111 100 110 101 0101 0110 0111 00100 101 100 00011 00010 0110

Hence, This bit string will be our compressed data.

When we send this data over the internet we also send a dictionary of characters and their corresponding bit codes as shown in step 6.

IV. FEASIBILITY OF SATONIC TECHNIQUE

Number of characters in original string = 27
Size of each character = 1 byte = 8 bits.
Total size of original string = 27 * 8 = 216 bits
No. of characters in the compressed string = 102
Size of each bit = 1 bit.
Total size of compressed data = 102 * 1 =102 bit.

Data saved
= Size of original string – size of compressed string
= 216 – 102 = 114 bits
Bit by character ratio
$= \frac{number\ of\ bits\ in\ compresed}{number\ of\ characters} = \frac{102}{27}$ = 3.77 B/C

Thus each character takes only 3.77 bits which would have otherwise taken 8 bits.

V. HUFFMAN CODING TECHNIQUE

Huffman coding is one of the techniques currently used for lossless data compression. It operates of the principle that the most redundant characters should be represented by minimum possible bits while less redundant bits can take more bits.

Let us compress our previous string using Huffman technique:
SATONIC COMPRESSION TECHNIQUE

In order to compress the text, we follow the following steps-

Step 1:
Count the repeated content of the data i.e. the redundant characters that occur in the string.
In the above string we can see that –

1. **S, O, N, I, C, E** occurs 3 times each.
2. **T** occurs twice.
3. **A, M, P, R, H, Q, U** occurs only once.

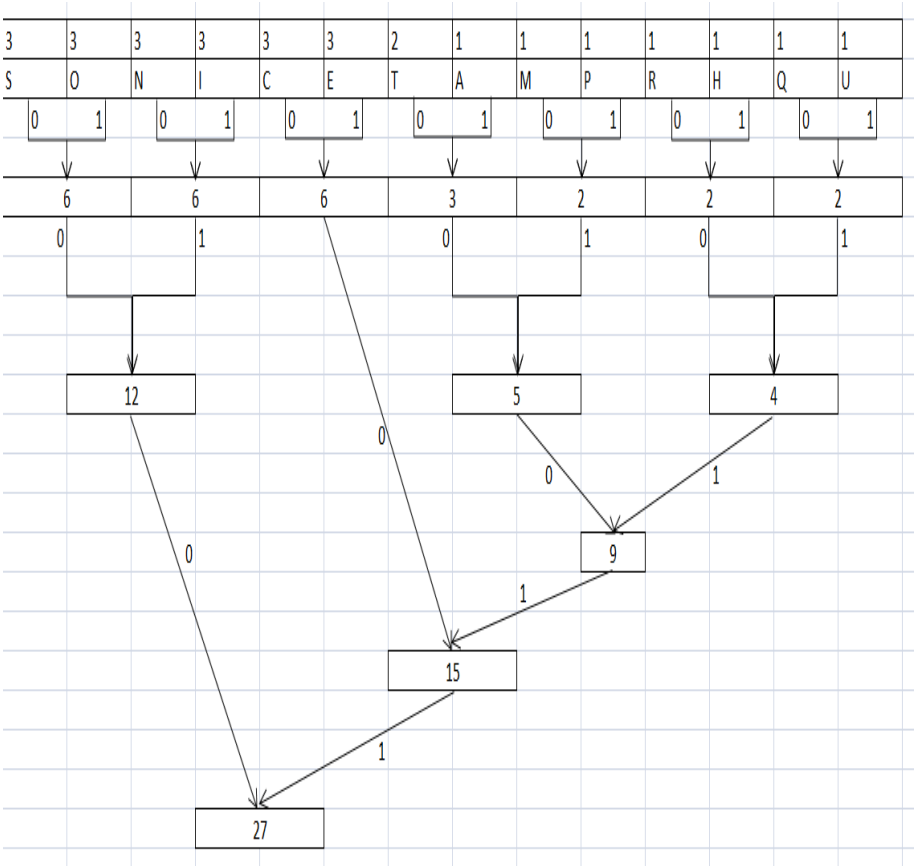
Step 2:
Now group these characters in descending order of redundancy. For example-
S,O,N,I,C,E,T,A,M,P,R,H,Q,U

Step 3:
Make a grid where you write these characters along with the number of times they have been repeated.

3	3	3	3	3	3	2	1	1	1	1	1	1	1
S	O	N	I	C	E	T	A	M	P	R	H	Q	U

Step 4:
Now find the two least redundant bits from the right side and add them up.

Proceed through this step recursively until the tree is complete.



Step 6:
Now we have to write the binary codes of each character of the string. To find out the binary code of each character we have to write the consequent branch bits and then the row number.

Example-

1. S = 000
2. O = 001
3. N = 010
4. I = 011
5. C = 100
6. E = 101
7. T = 11000
8. A = 11001
9. M = 11010
10. P = 11011
11. R = 11100
12. H = 11101
13. Q = 11110
14. U = 11111

Step 7: Now we represent the original string in the form of bit codes we have obtained –
So now –

SATONIC COMPRESSION TECHNIQUE

Will be written as –
000 11001 11000 001 010 011 100 100 001 11010 11011 11100 101 000 011 001 010 11000 101 100 11101 010 011 11110 11111 101

VI. FEASIBILITY OF HUFFMAN CODING

Number of characters in original string = 27
Size of each character = 1 byte = 8 bits.
Total size of original string = 27 * 8 = 216 bits
No. of characters in the compressed string = 99
Size of each bit = 1 bit.
Total size of compressed data = 99 * 1 =99 bit.

Data saved
= Size of original string – size of compressed string
= 216 – 99 = 117 bits
Bit by character ratio

$$= \frac{number\ of\ bits\ in\ compresed}{number\ of\ characters} = \frac{99}{27} = 3.66 \text{ B/C}$$

Thus each character takes only 3.66 bits which would have otherwise taken 8 bits.

Thus, we can see the difference in the B/C ratio of Huffman and Satonic Coding.
For Huffman Coding, B/C = 3.66
For Satonic Coding, B/C = 3.77

Huffman produces a better B/C ratio that Satonic so Huffman is more feasible in terms of Space Complexity.

VII. TIME COMPLEXITY

Satonic coding vs huffman coding

Time complexity of Huffman coding was found to be O(n log n), where n is the number of characters that are sent for the generation of Huffman tree.

Thus, the time taken to compress the file using Huffman coding is directly proportional to the number of characters in the Huffman tree.

i.e $T(H) \propto n$

where T(H) is the time taken to compress files using Huffman coding.

Since, T(H) = n log(n)

As we can see from both the algorithms, the criteria of formation of tree are mostly the same. Hence the time complexity of Satonic Coding = O(n log n)

But for the same set of data, number of characters to take part in the tree is different. Since Satonic follows the folding approach, the number of characters in the tree is reduced to one fourth.
Hence, for a data string with n characters

$$T(H) = n\log(n)\text{…….Eq. (1)}$$

$$T(S) = \frac{n}{4}(\log\frac{n}{4})$$

$$T(S) = \frac{n}{4}(\log n - \log 4)$$

$$T(S) = \frac{n\log n}{4} - \frac{n}{4}\log 4$$

$$T(S) = \frac{1}{4}(T(H) - n\log 4)$$

Thus, we can see that time taken to compress using Satonic technique is one – fourth of Huffman technique. Thus in terms of Time Complexity, Satonic Coding surpasses Huffman.

Advantages of satonic coding

1. With a bit – character ratio of about 3.77, it compresses the data to nearly half the original size.
2. Since the original data can be recovered fully, there is no reduction in data content or quality.
3. The time complexity of the algorithm is lower than that of Huffman as number of calculations are reduced.

VIII. CONCLUSION

Since Huffman Coding is better in terms of space complexity but Satonic proves to be better in terms of time complexity. So in order to decide which is better we need to find the Weissman score(W).[4]

$$W = \propto \frac{r\log T_s}{r_s\log T}$$

Let the r =2.1, T=35 and alpha=1 for standard compressor.
Now, For Huffman Coding –
Compression ratio (r) = (216 / 99) =2.18
Time = n log(n).
Here n = 27, so T = 27 log (27)=38.64 units.

$$w = 1 \times \frac{2.18(\log 35)}{2.1(\log 38.64)} = 1.0099$$

Now, For Satonic Coding –
Compression ratio (r) = (216 / 102) =2.117
Time = (n/4) log(n/4).
Here n = 27, so T = (27/4) log(27/4)=5.597 units

$$w = 1 \times \frac{2.117(\log 35)}{2.1(\log\ 5.597)} = 2.08109$$

Since Satonic Coding has a better Score, we can say that Satonic Coding is more feasible than Huffman approach.

IX. FUTURE WORK

As Satonic is currently only a text based compression, we will try to implement it on textual as well as video and 3d videos.[7]

This would be helpful in transmitting data over the internet where lot of data is to be transmitted at high speed such as video chatting and live streaming.

REFERENCES

1. *A Research Paper on Lossless Data Compression Techniques IJIRST –International Journal for Innovative Research in Science & Technology| Volume 4 | Issue 1 | June 2017 ISSN (online): 2349-601.*
2. *The Design and Analysis of Efficient Lossless Data Compression Systems*
3. *https://cs.stanford.edu/people/eroberts/courses/soco/projects/data-compression/lossless/index.htm*
4. *https://en.wikipedia.org/wiki/Weissman_score*
5. *Analysis of Data Compression Techniques using Huffman Coding and Arithmetic Coding*
6. *https://www.cse.ust.hk/~golin/Talks/Shannon_Coding_Extensions.pdf*
7. *Image Compression Using Run Length Encoding (RLE)*
8. *A critical study on the applications of run length encoding techniques in combined encoding schemes.*
9. *ARITHMETIC CODING FOR DATA COIUPRESSION IAN H. WIllEN, RADFORD M. NEAL, and JOHN G. CLEARY*
10. *https://en.wikipedia.org/wiki/Lossy_compression*
11. *https://en.wikipedia.org/wiki/Lossless_compression*