Intensification of Lempel-Ziv-Welch Algorithm

A.Deepa, Nitasha, Namrata Chopra

Abstract*: There is a necessity to reduce the consumption of exclusive resources. This is achieved using data compression. The data compression is one well known technique which can reduce the file size. A plethora of data compression algorithms are available which provides compression in various ratios. LZW is one of the powerful widely used algorithms. This paper attempts to propose and apply some enhancements to LZW, hence comes out with an efficient lossless text compression scheme that can compress a given file at better compression ratio. The paper proposes three approaches which practically enhances the original algorithm. These approaches try to gain better compression ratio. In approach1, it exploits the notion of using existing string code with odd code for a newly encounter string which is reverse of existing. In approach2 it uses a choice of code length for the current compression, so avoiding the problem of dictionary overflow. In approach3 it appends some selective set of frequently encountered string patterns. So the intensified LZW method provides better compression ratio with the inclusion of the above features.

***Index Terms*: Algorithm, compression, decompression Intensification.**

I. INTRODUCTION

Text plays a vital role in the digital world. Text compressions demand less loss. LZW is a popular lossless compression algorithm which gives a better practical compression ratio. There are some ways to enhance the existing LZW algorithm. This paper suggests an enhancement of LZW lossless text compression scheme that tries to compress a given file better thus makes better compression ratio. The project proposes and applies some enhancements to the existing algorithm LZW. We expect better compression ratio with our enhancements over the original LZW data compression algorithm. Data compression has got much importance in the domain of data storage and transmission. Data compression reduces the number of bits of a file to be archived/transmitted. Compression is useful because it helps reduce the overall file size. Data compression is a great deal of space can be saved while producing an output which is nearly indistinguishable from the original. Lossless compression algorithms usually exploit statistical redundancy in such a way as to represent the sender's data more concisely without error. Lossless compression is possible because most real-world data has statistical redundancy. There are two types of compression. One is lossy and the other is lossless compression. The advantage of lossy methods over lossless methods[1] is that in some cases a lossy method can produce a much smaller compressed file than any known lossless method, while still meeting the requirements of the application. Lossless compression schemes are reversible so that the original data can be reconstructed, while lossy schemes accept some loss of data in order to achieve higher compression.

II. Related Papers

Data compression [2] is the process of encoding information using fewer bits than a representation which is not encoded. Examples include the ZIP format and the gzip utilities. As with any machine, compressed data easily reduce the all resources. On the other side, compressed data must be decompressed to be used[3-4], and this extra processing may be detrimental to some applications. The important criterion for compression evaluation is compression ratio which is expected to be raised. The data compression is of two types: loss and lossless. Loss is preferable for audio, video, and images since it is bearable of having low quality.

Whereas text compressions strongly recommend lossless because nobody wants to have some meaningless or even sometimes horrible messages instead of correct ones. Data compression ratio [5-7] is the criteria to know reduction size of the com- pressed file over uncompressed file. The paper[8] attempts to improve LZW with three techniques include redundancy Encoding String Indexes, Estimating Probabilities for String Numbers, Exploiting Possibilities for Adaptive Loading of the Dictionary. The paper[9-10] presents modified LZW algorithms that support fast random access to the compressed text. Instead of fully decompressing the text and outputting the results selectively, the algorithms allow random access and partial decoding of the compressed text and displaying the relevant portion.

Compression ratio = uncompressed size /compressed size (1)

Thus a representation that compresses a 10MB file to 2MB has a compression ratio of 10/2 = 5. The file size is reduced to 5th portion of its original size. This increases the number by using efficient algorithms. The paper [11-12] discusses about better compression ratio with respect to time required for compression. As so many digital data uploading and downloading is done in recent days the time required for uploading and downloading the data should be done in such a way that the data is compressed in a rapid manner.

III. PROPOSED METHOD

Before LZW Data Compression Lempel-Ziv-Welch (LZW[4]) is a universal

Revised Manuscript Received on June,15 2019..

First Author name, His Department Name, University/ College/ Organization Name, City Name, Country Name.
Second Author name, His Department Name, University/ College/ Organization Name, City Name, Country Name.
Third Author name, His Department Name, University/ College/ Organization Name, City Name, Country Name.

lossless data compression algorithm created by Abraham Lempel, Jacob Ziv, and Terry Welch. Lempel- Ziv- Welch (LZW) is one of the powerful existing compression algorithm. It finds in many important applications like win zip, 7zip and etc. LZW is a fixed length coding algorithm. Uses 12bit unsigned codes. First 256 codes are the entire ASCII character set. Lateral entries in the LZW dictionary are strings and codes. Every LZW code word is a reference to a string in the dictionary. LZW compression replaces strings of characters with single codes. It does not do any analysis of the incoming text. Instead, it just adds every new string of characters it sees to a table of strings.

A. Algorithm of intensified LZW compression method

In the modified algorithm logic is added to assign codes for reverse string pairs. This logic exploits the notion of using existing string code with odd code for a newly encountered string which is reverse of existing. This modification plays an important role to get a better compression file. This is one of the implementation approaches which provide better accuracy as in Fig.1. The input data is obtained byte by byte and stored. The type of input data is checked and based on the result, the string value is assigned for it. The data in the form of byte is obtained and stored and when the bytes are all read, the code for the particular string is displayed. The Table I and II explains the steps involved in the process of compression and decompression.

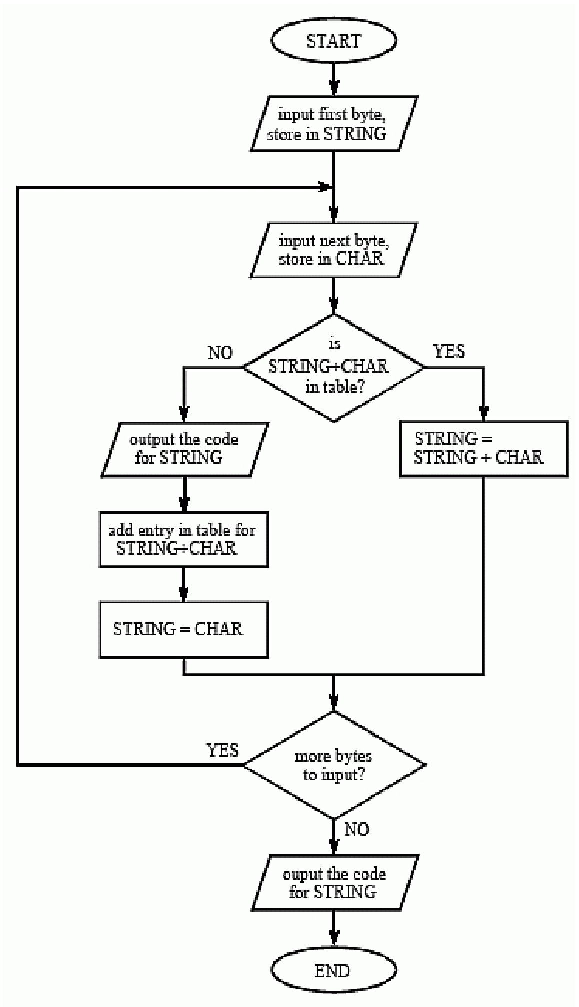
START
input first byte, store in STRING
input next byte, store in CHAR
is STRING+CHAR in table?
NO
YES
output the code for STRING
STRING = STRING + CHAR
add entry in table for STRING+CHAR
STRING = CHAR
more bytes to input?
YES
NO
ouput the code for STRING
END

Fig.1. Flowchart of intensified LZW compression algorithm

TABLE I. INTENSIFIED LZW COMPRESSION ALGORITHM

```
1:  DEFINE CODE LENGTH
2:  if  (STR = get input character) = EOF then
3: while there are still input characters do
4: CHAR = get input character
5: if  STR+CHAR is in the String table then
6: STR = STR+CHAR //Return EVEN code
7: else if  STRREV(STR + CHAR) is in the string table
8: then  STR = STR+CHAR //Return ODD code
9: else
10:   output code for STR
11:   add STR + CHAR into the String table
12:   STR = CHAR
13:   end if
14:   end while
15:   Output the code for STR
 16:  end if
```

TABLE II. INTENSIFIED LZ DECOMPRESSION ALGORITHM

```
1:  DEFINE CODE LENGTH
2:  Read OC = OLD CODE
3:  if  OC is not EOF then
4: OC = get translation of OC
5: output OC
6: CHARACTER = OC
7: while there are still input characters do
8: Read NC = NEW CODE
9: if  NC is in not DICTIONARY then
10:   STRING = get translation of OC
11:   STRING = STRING + CHARACTER
12:   else
13:   STRING = get translation of NC
14:   end if
15:   output STRING
16:   CHARACTER = first character in STRING
17:   add OC + CHARACTER into the DICTIONARY
18:   OC = NC
19:   end while
20:   Output string for code and end if
```

B. Results

The plotted graph in Fig.2 shows the difference of the original LZW compression ratio with three approaches and intensified LZW compression ratio.

Original file size	Compressed file size	Compression ratio
2kb	1.35	1.489
3kb	1.94	1.554
4kb	2.60	1.538

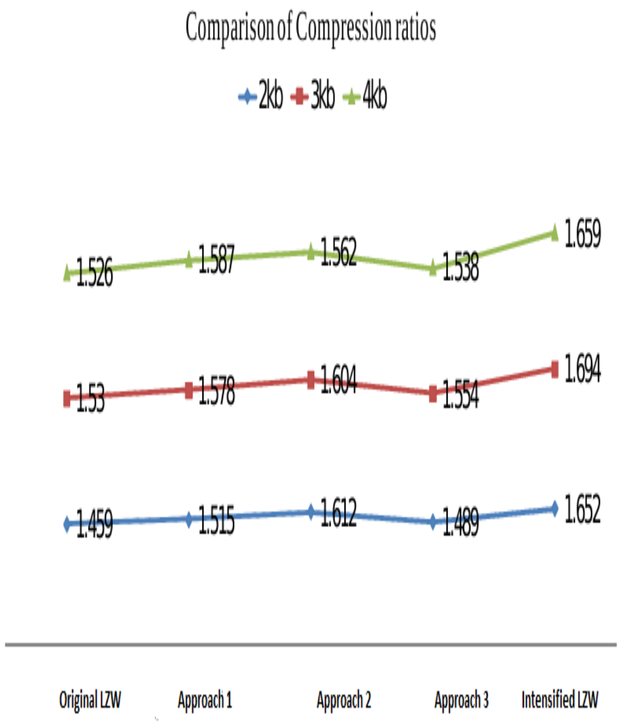

Fig.2. Comparison of all techniques compression ratios

Three input text files [7] are taken for compress with the original LZW. The results are shown below in Table.III. To reduce the compression file size and dictionary entries this technique using even and odd codes. The modified LZW is compression the data better than the original LZW. Now, we getting better compression ratio compare to original LZW compression ratio. Here used bench mark text files [7] for calculating the performance between enhanced LZW and original LZW. The results are shows the enhanced LZW and the plotted graph is showing the difference of the original LZW and Modified LZW compression ratio as in Table. IV. enhancement in compressing the data.

TABLE III. ORIGINAL LZW COMPRESSION RATIO

Original file size	Compressed file size	Compression ratio
2kb	1.37	1.459
3kb	1.96	1.530
4kb	2.62	1.526

TABLE IV. INTENSIFIED LZW COMPRESSION RATIO

ORIGINAL FILE SIZE	COMPRESSED FILE SIZE	COMPRESSION RATIO
2kb	1.32	1.515
3kb	1.90	1.578
4kb	2.52	1.587

There are three approaches which are used in this algorithm. In approach 1, the codes are reversed and reverse codes are also used in the process of compression. The technique of exploiting reverse of the string using even and odd codes are used in this algorithm instead of signed codes. The normal drawback of using signed codes is that the first 256 codes become unusable. When the reverse data codes are used, the size of the file is reduced such as nearly 40 percent of original size is reduced. The results are shown Table V.

TABLE V. INTENSIFIED LZW COMPRESSION RATIO

The second approach is the option to select the code length. The technique is implemented with 9 bit to 16 bit. But using 9bit to 14bit better compression ratio is obtained. Each character carries 8 bit in the original file and original LZW renders 12bit to each character. If 16bit is used for compression, it carries the data of size which is double the original character size. So it is better to use in case of large size files. Betterment ratio is obtained with the usage of 9bit to 12bit code length for gaining better compression ratio from the small and medium files. By this approach nearly 20 percent of storage space is released. The results are shown in Table.VI. The third approach is to append frequently encountered string patterns in to the dictionary entries. These entries are entered in to the dictionary after 255. Because 0 to 255 entries are reserved for ASCII character set. ASCII characters can be special symbols, numbers and alphabetic in both cases. Codes are assigned for lateral entry string patterns when frequently encountered string patterns are finished. The results are shown in Table. VII. By using all these three techniques such as exploiting the reverse string compression, selecting the code length compression and using frequently entered string length compression, the intensified LZW algorithm provides better compression ratio. The compression of data is visible as in Fig.3. The data explains the process in which the compression ratio is improved. With the availability of prior information regarding the original data a static dictionary can be used. When the previous information is not available, dynamic dictionary provides solution to compression. Lempel–Ziv algorithm (LZ) is a dictionary based coding technique commonly used in lossless file compression. This is broadly accepted due to the inclusion of adaptations of different file formats. When enhancements are made to the algorithm with different modules for different specifications, the significance and compression ratio is increased. This method is suitable for larger files and of less significance in case of small file. The data is compressed in a form such that the content is recovered as per the original data content by means of proper compression and decompression techniques. Though the data is compressed in a fine manner, the original content is exactly obtained after decompression. Hence this intensification is found to be a better compression technique with the inclusion of these three approaches.

TABLE VI. EXPLOITING REVERSE OF THE STRING COMPRESSION

TABLE VII. PROVIDING OPTIONS FOR SELECTION OF CODE LENGTH COMPRESSION

Original file size	Compressed file size	Compression ratio
2kb	1.24	1.612
3kb	1.87	1.604
4kb	2.56	1.562

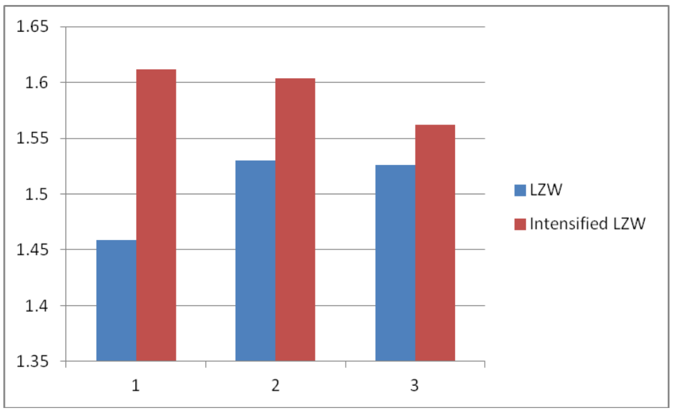

Fig.3. Intensified LZW Performance

The ratio of compression achieved in LZW is shown in blue color and the red color indicates the compression ratio of intensified LZW algorithm. The X axis indicates the three different approaches which are used and Y-axis indicated the ratio obtained for each of the consideration. Hence it is clearly visible from the figure that the compression ratio is increased in our method.

IV. CONCLUSION

The paper proposed new enhancements for increasing the compression ratio over the original LZW compression ratio. These enhancements are giving better compression ratio over the original LZW. The enhanced LZW is compressing the text more than existing LZW. The experimental results show the performance of enhanced LZW. The paper discusses the compression of data and the enhancements in formal data compression methods. The compression is based on three different considerations such as code length, reverse code and frequent pattern. This makes the enhancement in the compression of data. The results achieved are better since the compression ratio is high

REFERENCES

[1] G David Solomon, Data compression: The complete references book,

Original file size	Compressed file size	Compression ratio
2kb	1.21	1.612
3kb	1.77	1.604
4kb	2.41	1.562

Third edition, 2004.

[2] Zhou Yan-li, Fan Xiao-ping, Improved LZW algorithm of lossless data compression for WSN, Conference proceedinmgs, 3rd International Conference on Computer Science and information Technology, 2010

[3] M. Abu Alsheikh, S. Lin, D. Niyato, H.P. Tan, Rate-distortion balanced data compression for wireless

sensor networks, IEEE Sens. J., 16 (2016), pp. 5072-5083.

[4] J.UthayakumarT.VengattaramanP.Dhavachelva, 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.

[5] J. Abel, W. Teahan, Universal text preprocessing for data compression, IEEE Trans. Comput., 54 (2005), pp. 497-507.

[6] Ezhilarasu P,Karthik Kumar P,LZW Lossless Text Data Compression Algorithm – A ReviewInternational Journal of Computer Science & Engineering Technology (IJCSET, Vol. 6 No. 11 Nov 2015.

[7] H. Amri, A. Khalfallah, M. Gargouri, N. Nebhani, J.-C. Lapayre, M.-S. Bouhlel Medical image compression approach based on image resizing, digital watermarking and lossless compression, J. Signal Process. Syst., 87 (2017), pp. 203-214.

[8] Simrandeep kaur, V.Sulochana Verma, Design and Implementation of LZW Data Compression Algorithm, International Journal of Information Sciences and Techniques (IJIST) Vol.2, No.4, July 2012

[9] Sawsan A. Abu Taleb Hossam M.J. Musafa Asma'a M. Khtoom Islah K. Gharaybih, Improving LZW Image Compression, European Journal of Scientific Research, Vol.44 No.3 (2010), pp.502-509.

[10] Sawsan A. Abu Taleb , Hossam M.J. Musafa , Asma'a M. Khtoom Improving LZW Image CompressionEuropean Journal of Scientific Research 1450-216X Vol.44 No.3 (2010), pp.502-509

[11] Evon Abu-Taieh1, Issam AlHadid, A New Lossless Compression Algorithm, Modern Applied Science, Canadian Center of Science and Education, Vol. 12, No. 11, 2018.

[12] Restu Maulunida*1, Achmad Solichin, Optimization of LZW Compression Algorithm With Modification of Dictionary Formation , Indonesian Journal of Computing and Cybernetics Systems) Vol.12, No.1, January 2018, pp. 73~82.

AUTHORS PROFILE

A.Deepa, B.E,M.Tech,(Ph.D) was awarded B.E degree from M.S University and M.Tech from Prist university and is pursuing PhD from Sathyabam University. She is an Assistant professor in Chandigarh Engineering College, Landran.She has published many papers and has published books for engineering subject. Her research area is image processing, data processing and data management.

Nitasha has received her UG and PG degree in Electronics Engineering. She is working as Assistant professor in Chandigarh engineering College. She has good experience of teaching in engineering college.

Namrata Chopra has received her UG and PG degree in Electronics Engineering. She is working as Assistant professor in Chandigarh engineering College. She has more than 10 years of experience in teaching.