International Journal of Engineering Applied Sciences and Technology, 2020
Vol. 4, Issue 11, ISSN No. 2455-2143, Pages 302-304
Published Online March 2020 in IJEAST (http://www.ijeast.com)

AN EFFICIENT METHOD TO REDUCE LZW ALGORITHM OUPUT CODE LENGTH

Srinivasa Rao Namburi, P A V Krishna Rao, Praveen Kumar Muvva, Suresh Kumar K
Asst. Prof Dept of IT
Bapatla Engineering College, AP, India

V Naveen Kumar
Dept of CSE
Bapatla Engineering College, AP, India

Abstract: Text compression techniques are essential techniques to use for data transmission from one location to another location and also used to keep the more data. LZW is an lossless text compression technique to using by most of people. This paper proposes to decrease the length of the LZW algorithm output, thus we can compress more data at a time.

Keywords: **Code Length, LZW, Text data Compression**

I. INTRODUCTION

LZW is a lossless compression algorithm [7] which gives best text data compression. This paper focus on decrease the output code length of the LZW algorithm by improving existing LZW algorithm. Concentrate mainly on text data compression [5] since text plays a crucial role in the present digital data world.

LZW is a dictionary based algorithm [8]. We are compress and decompress the file using dictionary. LZW dictionary contains strings and codes. LZW get a 8 bit input character and produce the compressed bit is in size of 12 bits. Approach is reduce the size of compressed character size in bitwise.

II. LITERATURE SURVEY

The important criterion for compression evaluation is compression ratio which is expected to be raised. The data compression is of two types: Lossy and lossless [6]. Lossy [10] 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.

2.1 Lossless verses Lossy compression

1. The advantage of lossy [9] 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.

3. 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.

2.2 LZW Data Compression

Lempel-Ziv-Welch (LZW [1]) is a universal lossless data compression algorithm created by Abraham Lempel, Jacob Ziv, and Terry Welch.Lempel- Ziv-Weltch (LZW) is one of the powerful existing compression algorithms. It finds in many important applications like win zip, 7zip and etc.

1. 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.

2. Every LZW code word is a reference to a string in the dictionary.

3. LZW compression[12] 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. Compression occurs when a single code is output instead of a string of characters.

Basic idea [1]

(1) Replaces strings of characters with single integer codes.
(2) A table of string/code pairs is built as the compression algorithm reads the input file.
(3) The table is reconstructed as the decompression algorithm reads.

2.3 Compression

The LZW compression algorithm [1] in its simplest form is shown below. A quick examination of the algorithm shows that LZW is always trying to output codes for strings that are already known. And each time a new code is output, a new string is added to the string table. The output code size is 12 bits for character.

Algorithm 1: LZW Compression Algorithm

```
1: if (STR = get input character) is not EOF then
2: while there are still input characters do
3: CHAR = get input character
4: if STR+CHAR is in the string table then
5: STR = STR+CHAR
6: else
7: output the code for STR
8: add STR+CHAR to the string table
9: STR = CHAR
10: end if
11: end while
12: Output the Code for STR
13: end if
```

2.4 Decompression

The companion algorithm for compression is the decompression algorithm [1].It needs to be able to take the stream of codes output from the compression algorithm, and use them to exactly recreate the input stream.

The table can be built exactly as it was during compression, using the input stream as data. This is possible because the compression algorithm always outputs the STRING and CHARACTER components of a code before it uses it in the output stream. This means that the compressed data is not burdened with carrying a large string translation table.

Algorithm 2: LZW Decompression Algorithm

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

III. DESIGN

3.1 Design Approach

3.1.1 Providing options for selection of code length
Our approach facilitates the user to select the code length based on file size.
For example:
1. 9bit: small files.
2. 12bit: medium files.
3.14bit: large files.
As code length decreases file we can obtain better text compression.

3.2 Modified LZW Compression Algorithm [11]

Algorithm3: Modified 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
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
```

3.3 Modified LZW Decompression Algorithm

Algorithm4: Modified LZW 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
21: end if
```

IV. IMPLEMENTATION

4.1 Providing options for selection of code length

As our approach lets the user to select the code length[4], compression ratio increases for smaller files. For example if code length 9 is sufficient for a smaller file then each character can be replaced by 9 bits, which is 3 less than 12 (original LZW code length). The file contains more text this approach can reduce more bits.

The following table of results were obtained by experimental the algorithm on different file size with different input code lengths**.**

Original file size	Compressed file size (Old LZW)	Compressed file size (New LZW)
2 Kb	1.37 Kb	1.24 Kb
3 Kb	1.96 Kb	1.87 Kb
4 Kb	2.62 Kb	2.56 Kb

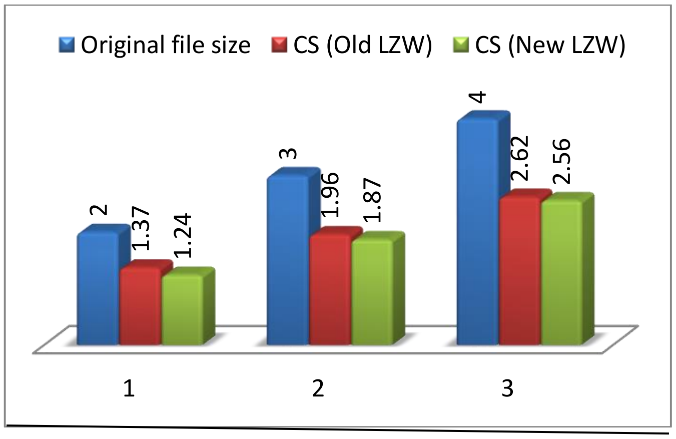

Fig 4.1 Providing options for selection of code length compression

V. CONCLUSION & FUTURE WORK

An Improved LZW algorithm is presented in this report. An evaluation result states that this modified LZW algorithm performs better compression than the existing LZW algorithm in terms of file size in Kilo Bites.

The suggested future work is to reduce the size of the character output of the LZW and make better use of the dictionary.

VI. REFERENCES

[1] David Solomon. (2004). Data compression: The Complete reference book, Pub-SV 3rd Edition.
[2] Michael Dipper stein Lempel-Ziv-Welch (LZW) Encoding Discussion. http://michael.dipperstein.com/lzw/.
[3] Ms. Agrawal Arohi K, Prof. V. S. Kulkarn (2014), FPGA Based Implementation of Data Compression using Dictionary based "LZW"Algorithm, IJIREECE, Vol. 2, Issue 4, April
[4] Srinivasa Rao N, Praveen Kumar. (2020), A New Approach to Increase LZW Algorithm Compression Ratio, IJEAST, Vol. 4, Issue 10,(pp: 141-144).
[5] J. Abel, W. Teahan. (2005), Universal text preprocessing for data compression, IEEE Trans. Comput., 54 (2005), pp. 497- 507.
[6] Ezhilarasu P, Karthik Kumar P (2015) ,LZW Lossless Text Data Compression Algorithm – A Review International Journal Of Computer Science & Engineering Technology (IJCSET), Vol. 6.
[7] H. Amri, A. Khalfallah, M. Gargouri, N. Nebhani, J.-C. Lapayre, M.-S. Bouhlel (2017) Medical image compression approach based on image resizing, digital watermarking and lossless compression, J. Signal Process. Syst., 87,(pp. 203-214).
[8] Simrandeep kaur, V.Sulochana Verma (2012), Design And Implementation of LZW Data Compression Algorithm, International Journal of Information Sciences and Techniques (IJIST) Vol.2, No.4, July.
[9] Sawsan A. Abu Taleb , Hossam M.J. Musafa , Asma'a M.Khtoom(2010), Improving LZW Image Compression, European Journal of Scientific Research 1450-216X Vol.44 No.3, (pp.502-509).
[10] Evon Abu-Taieh1, Issam AlHadid (2018), A New Lossless Compression Algorithm, Modern Applied Science, Canadian Center of Science and Education, Vol. 12, No. 11.
[11] Li & Drew (2003) , Fundamentals of Multimedia: Lossless Compression Algorithms, @ Prentice Hall .
[12] Simrandeep kaur, V.Sulochana Verma (2012), Design and Implementation of LZW Data Compression Algorithm, (IJIST) Vol.2, No.4, July.