International Journal of Engineering Applied Sciences and Technology, 2020
Vol. 4, Issue 10, ISSN No. 2455-2143, Pages 141-144
Published Online February 2020 in IJEAST (http://www.ijeast.com)

A NEW APPROACH TO INCREASE LZW ALGORITHM COMPRESSION RATIO

Srinivasa Rao Namburi
Asst. Prof, Dept of IT
Bapatla Engineering College, Bapatla,
AP, India

Praveen Kumar Muvva
Asst. Prof, Dept of IT
Bapatla Engineering College, Bapatla,
AP, India

P A V Krishna Rao
Asst. Prof, Dept of IT
Bapatla Engineering College, Bapatla,
AP, India

Prasad G
Asst. Prof, Dept of IT
Bapatla Engineering College, Bapatla,
AP, India

Abstract: Data compression techniques are widely used as it reduces the consumption of exclusive resources. LZW is one of the widely used lossless compression algorithm for this purpose. The paper attempts to increase compression ratio of the LZW algorithm by proposing new approach.

Keywords: **Data Compression, Dictionary, LZW, Compression ratio.**

I. INTRODUCTION

LZW is a popular lossless compression algorithm [7] which gives a better practical compression ratio. This project aims to increase the compression ratio by enhancing existing LZW algorithm. Concentrate mainly on text compression [5] since text plays a vital role in the digital world.

LZW is a dictionary based algorithm [8]. We are compress and decompress the file using dictionary. In this dictionary first 256 codes are reserved for entire ASCII character set. Lateral entries in the LZW dictionary are strings and codes. Approach is appends some selective set of frequently encounter string patterns.

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
(2) smaller compressed file than any known lossless method, while still meeting the requirements of the application.

(2) 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 Data Compression Ratio

Data compression ratio [2] is the criteria to know reduction size of the compressed file over uncompressed file.

$$\text{Compression ratio} = \frac{\text{Uncompressed size}}{\text{Compressed size}}$$

Thus a representation that compresses a 10MB file to 2MB has a compression ratio of 10/2 = 5. Meaning that the file size was cut down to 5th portion of its original size. Always we try to increase this number by using efficient algorithms.

2.3 Related research paper

(1) The R. Nigel Horspool [3] attempts to improve LZW with three techniques that include Redundancy Encoding String Indexes, Estimating Probabilities for String Numbers, Exploiting Possibilities for Adaptive Loading of the Dictionary.

2.4 LZW Data Compression

Lempel-Ziv-Welch (LZW [4]) is a universal lossless data compression algorithm created by Abraham Lempel, Jacob Ziv, and Terry Welch.Lempel- Ziv-Weltch (LZW) is one of

International Journal of Engineering Applied Sciences and Technology, 2020
Vol. 4, Issue 10, ISSN No. 2455-2143, Pages 141-144
Published Online February 2020 in IJEAST (http://www.ijeast.com)

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 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.

2.5 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.

Algorithm 1: LZW Compression Algorithm [12]
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

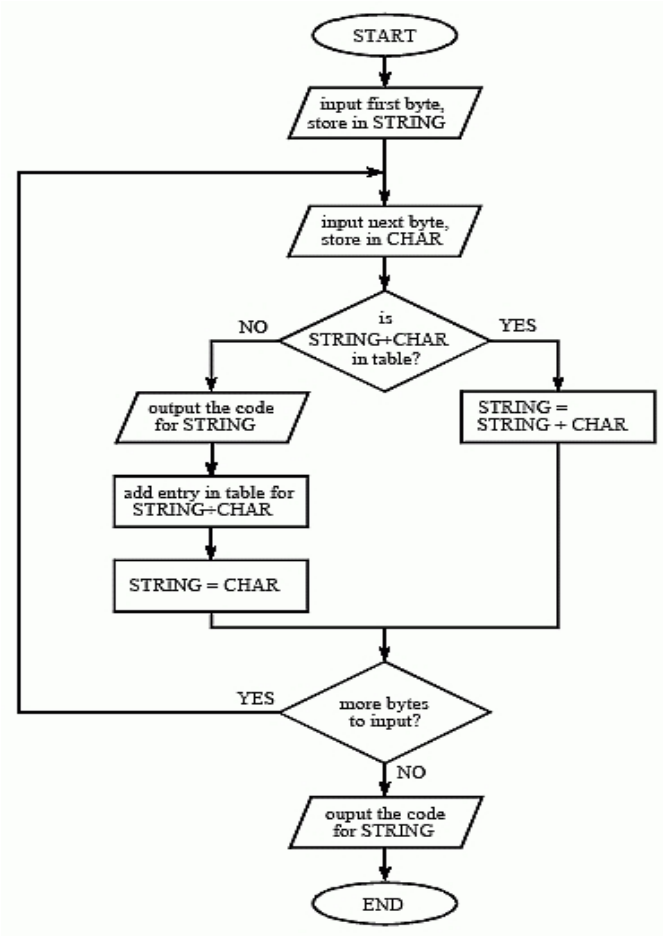

Fig 2.1 LZW Compression algorithm

2.6 Decompression

The companion algorithm for compression is the decompression algorithm [4].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

International Journal of Engineering Applied Sciences and Technology, 2020
Vol. 4, Issue 10, ISSN No. 2455-2143, Pages 141-144
Published Online February 2020 in IJEAST (http://www.ijeast.com)

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

The design plan is with respect to the salient considerations discussed below.

3.1 Design Considerations

Performance considerations:

- Compression ratio: number of bits reduced.

Dictionary decisions:

How large should we make the dictionary?
What do we do if the dictionary fills?

The size of the dictionary is limited by the code length of the algorithm and if the dictionary overflow occurs then no more new entries into the dictionary is possible after compress the rest by using the dictionary constructed so far.

Data decisions:

Can we shape the dictionary to improve compression?
Can we shape the data to make it easier to compress?

We will shape the dictionary by appending some more strings at initial stage and by switching from unsigned to signed codes. We shouldn't shape the data because in lossless the requirement is "nothing should be lost because of compression and as it is should be retrieved by decompression [13]".

3.2 Design Approach

3.2.1 Appending frequently encountered string patterns to the dictionary.

The words having the high probability of occurrence in the general text will be added to the dictionary selectively. For example, words like as, at, an, in, on and etc. Using this we can reduce the number of bits required.

Eg: Communication, working and etc.

3.3 Modified LZW Compression Algorithm[11]

Algorithm3: Modified LZW Compression Algorithm

1: if (STR = get input character) = 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 code for STR
8: add STR + CHAR into the String table
9: STR = CHAR
10: end if
11: end while
12: Output the code for STR
13: end if

3.4 Modified LZW Decompression Algorithm

Algorithm4: Modified 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

IV. IMPLEMENTATION

4.1 Appending frequently encountered string patterns to the dictionary

The results obtained after applying techniques which is explained in section IV is tabulated as follows:

International Journal of Engineering Applied Sciences and Technology, 2020
Vol. 4, Issue 10, ISSN No. 2455-2143, Pages 141-144
Published Online February 2020 in IJEAST (http://www.ijeast.com)

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

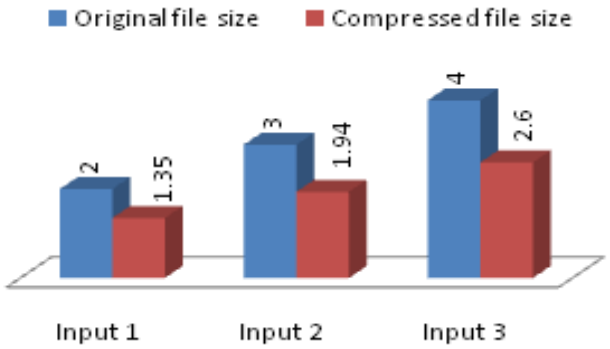

Fig 4.1 Appending frequently encountered string patterns compression

V. CONCLUSION

An enhanced LZW algorithm is presented in this report. An experimental result indicates that this algorithm performs better than the existing LZW algorithm in terms of compressed file size and compression ratio. Limitations of this work include dictionary overflow with large files and increased searching time.

VI. FUTURE WORK

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

VII. REFERENCES

[1] David Solomon. Data compression: The Complete reference book, Pub-SV 3rd Edition, 2004.
[2] Michael Dipper stein Lempel-Ziv-Welch (LZW) Encoding Discussion. http://michael.dipperstein.com/lzw/.
[3] R. Nigel Horspool. Improving LZW, IEEE, pages : 332-341, 1991.
[4] Christina Zeeh. The Lempel Ziv Algorithm, Seminar Famous Algorithms, 16th January, 2003.
[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 Improving LZW Image Compression, European Journal of Scientific Research 1450-216X Vol.44 No.3 (2010), pp.502-509.
[10] Evon Abu-Taieh1, Issam AlHadid, A New Lossless Compression Algorithm, Modern Applied Science, Canadian Center of Science and Education, Vol. 12, No. 11, 2018.
[11] 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.
[12] J.Uthayakumar,T.Vengattaraman,P.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.