INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH & DEVELOPMENT

ISSN 2278 – 0211 (Online)

An Intelligent LZWS Compression Algorithm to Achieve High Compression by Using an Efficient Technique

Sonia Setia
Department of Computer Engineering
YMCA University of Science and Technology, Faridabad, India
Dr. jyoti
Department of Computer Engineering
YMCA University of Science and Technology, Faridabad, India

Abstract:
Data compression is a key component for data storage systems and for communication purposes. Lempel-Ziv-Welch (LZW) data compression algorithm is popular for data compression because it is an adaptive algorithm and achieves an excellent compromise between compression performance and speed of execution. LZW is a dictionary based data compression algorithm, which compress the data in a lossless manner so that no information is lost. But LZW algorithm fails in case of small amount of data. In this case it expands the data instead of compressing it. In this paper a system is proposed to achieve high compression even if data file contains small amount of data.

Key words*: Compression, Encryption, Adaptive*

1. Introduction

LZW [1] is a popular dictionary based algorithm for data compression, developed by Terry Welch in 1984. It is an effective and adaptive algorithm. Compression is a way to reduce the number of bits in a frame but retaining its meaning. Compressed data can be transferred faster to and from disk. It decreases space, time to transmit, and cost. It is a technique to identify redundancy and to eliminate it. Data compression increases disk bandwidth. But it does not provide any security. If intruder knows the decompression algorithm, intruder can easily access our message or information. So for providing security we encrypt the message before compressing the message, so that if intruder has decompression algorithm, intruder cannot see our message. LZW algorithm depends on redundancy of the data i.e. if the input file contains the high redundancy, high compression is achieved and if the input file contains low redundancy, low compression is achieved. If input file contains less amount of data, that means less redundancy and data will be expanded instead of compression. But a system is proposed which compress the data even input file has small amount of data.
The rest of the paper is organized as follows. Section 2 reviews the basic concept and related work of compression and encryption. Section 3 presents the compression techniques. Section 4 describes the proposed system. Section 5 deals with results. Section 6 concludes the paper. Section 7 presents the future work.

2. Basic Concepts

There are various techniques to compression.

2.1. Compression Techniques

- LZW compression
- Arithmetic coding
- Run length encoding
- LZ 77
- LZ 78
- LZFG
- LZRW1

- LZRW4 and so on.

2.2. Encryption Techniques

- Data encryption standard (DES)
- Advance Encryption Standard (AES)
- Caesar cipher technique

In our system we are using LZW technique for compression and Caesar cipher technique for encryption.

3. Compression Technique

3.1. LZW Compression

It is a dictionary based algorithm. It is a variant of Lempel Ziv 78 compression algorithm. In this we initialize the dictionary to all symbols in alphabet. In common case of 8-bit symbols, first 256 entries of the dictionary 0 through 255 are occupied before any data is input. LZW token consist a pointer to the dictionary. When the dictionary is initialized, the next input character finds in dictionary.
The idea of LZW is that the encoder input symbols one by one and accumulates them in string I. After each symbol is input and is concatenated to I, dictionary is searched for string I. As long as string I is found in dictionary, the process continues. But at some point if adding text symbol suppose x causes the search to fail; string I is in dictionary but string Ix is not. At this point the encoder outputs the dictionary pointer that points to string I, and saves string Ix in the next available entry in the dictionary and then initialize string I to symbol x.

3.1.1. LZW Encoding Algorithm

Algorithm:

- Step 1 Initialize dictionary to contain single character string.
- Step 2 Read first input character prefix string ω from the data file.
- Step 3 Read next input character k from the data file.

If no such k (input exhausted): output:=code(ω):Then EXIT
If ωk exists in dictionary: ω: = ωk; Repeat Step 3.
Else If ωk not in dictionary. Then output :=code (ω)
Dictionary: = ωk;
ω :=k;
Repeat Step 3
Step 4 End

3.1.2. LZW Decoding Algorithm

Algorithm:

- Step 1 Read first input code and CODE=OLD code=input code with CODE=code (k), output=k, Fin char=k.
- Step 2 Read next input code, CODE =INCODE=next input code.
  If no new code: EXIT. Else:
- Step 3 If CODE=code (ωk): stack = k
  CODE: =code (ω)
  Repeat Step 3
  Else if CODE = code (k): output=k, Fin char=k.
  Do while stack not empty:
  Output = Stack top, Pop stack.
  Dictionary=OLD code, k.
  OLD code=IN code;
  Repeat Step 2
  Step 4 End

4. Proposed System

LZW algorithm depends on size of the data file i.e. if the input file contains the large amount of data, which means there is more chance of redundancy and high compression is achieved and if the input file contains less amount of data, which means there is less or even no chance of redundancy, in this case LZW algorithm expand the data so no compression is achieved. But a system LZWS (Lempel-Ziv-Welch-Setia) algorithm is proposed to achieve good compression, even if data file contains fewer amounts of data.

4.1. LZWS Encoding Algorithm

Algorithm:

- Step 1. Initially Dictionary is empty.
- Step 2. Scan the input word by word and check that it is in dictionary or not.
- Step 3 If (it is not in dictionary)

then:

Add this string in dictionary and assign a unique ASCII code starting from 256 and scan next string till end.

Else if (it is already in dictionary)

then:

Use that code in place of string and scan next string till end.

In this way we can compress the small amount of data also. Depending upon the redundancy, it will decide whether to compress the data or not. But it will not expand the data even if there is no redundancy in input data file.

5. References

1. Terry Welch, "A technique for high performance data compression," IEEE Computer, vol-17, pp. 8-19, June 1984.
2. J.Ziv and A. Lempel, "A universal algorithm for sequential data compression," IEEE Transaction on Information Theory, vol-23, pp 337-343, May 1977.
3. J.Ziv and A. Lempel, "Compression of individual sequences via variable length coding," IEEE Transaction on Information Theory, vol-24, pp 530-536, 1978.
4. H.K.Reghbati, "An overview of data compression techniques," Computer, Vol-14, No. 4, pp. 71-76, Aprl. 1981.
5. Holger Kruse and Amar Mukherjee, "Data compression using text encryption," IEEE Data Compression Conference, pp 447, 1997.
6. Chuanfeng Lv and Qiangfu Zhao, "Integration of data compression and cryptography: another way to increase the information security," IEEE Computer Society, vol-2, pp 543-547, 2007
7. Jianmin Jiang,"Pipeline algorithm of RSA data encryption and data compression," IEEE Proceeding of International Conference on Communication Technology, vol-2, pp 1088-1091, 1996
8. M.B.Lin, and Y.Y.Chang, "A new architecture of a two-stage lossless data compression and decompression algorithm," IEEE Transaction on VLSI Systems, Vol-17, No. 9, pp. 1297-1303, 2009
9. Parvinder Singh, Sudhir Batra, and HR Sharma, "Evaluating the performance of message hidden in 1st and 2nd bit plane", WSEAS Trans.on Information Science and Applications, Issue 8, vol: 2, pp: 1220- 1227,August 2005
10. Shen Jian-Hua, "Series of MSP430 16-bit Ultra-low Power Microcontroller Principles and Applications," Tsinghua University Press, pp: 57-98, 2004.
11. Tong Lai Yu, "Data compression for PC software distribution", Software Practice and Experience, vol 26, pp. 1181-1195, November 1996.
12. M. J. Handy, M. Haase, D. Timmermann, "Low Energy Adaptive Clustering Hierarchy with Deterministic Cluster-Head Selection," in Proc.4th IEEE International workshop on Mobile and Wireless Communications Network, Stockholm, pp: 368-372, September 2002.
13. Naoto Kimura, Shahram Latifi, "A Survey on Data Compression in Wireless Sensor Networks," In Proceedings of the International Conference on Information Technology: Coding and Computing, Vol. 2, pp: 8- 13, 2005.
14. Mo Chen and M. L. Fowler, "Data compression trade-offs in sensor networks," Conference on Information Sciences and Systems, Vol. 55, pp: 96- 107, 2004.
15. Zhou Si-Wang, Lin Ya-Ping, Zhang Jian-Ming, "Wavelet-based Data Compression Algorithm based on Ring Model in Sensor Networks," Journal of Software, vol: 18, pp: 669-680, 2007.
16. Li Lei-Ding, Ma Tie-Hua, You Wen-Bin, "Analysis of common lossless compression algorithm," Electronic Design Engineering, vol: 17, pp 49-53, 2009.
17. Deng Hong-Gui, Wang Jin-Xiu, Cao Ling-Ii, "Improved LZW Compression Algorithm based on BWT and its Application to Sensor Networks," Chinese Journal of Sensors and Actuators , Vol: 21, pp: 1047-1051, 2008