International Journal of Electrical Electronics & Computer Science Engineering
Special Issue - ICSCAAIT-2018 | E-ISSN : 2348-2273 | P-ISSN : 2454-1222
Available Online at www.ijeecse.com

Multilevel Dictionary Based LZW Algorithm

Sonia Setia[1], Neha Sewal[2]
[1-2]Assistant Professor, School of Computer Science, Lingaya's University, Faridabad, Haryana, India
[1]setiasonia53@gmail.com, [2]neha. sawal@gmail.com

Abstract: ***LZW is one known popular dictionary based data compression adaptive algorithm. It is appropriate for communication, because it require no extra communication between sender and receiver. In this paper we present a scheme for dynamic dictionary size to overcome dictionary flushing out problem. We use multi levels for dictionary. In the first level we encode any string in dictionary using just 10 bits. In second level we encode any string in dictionary using just 11 bits. In general case for*** 2^n ***dictionary size we encode any string in dictionary using just (n+1) bits, n bits for data and 1 bit (M.S.B bit) used for parity bit.***

Keywords: Multilevel, Compression, Dictionary, LZW and Dynamic.

I. INTRODUCTION

Most of the data compression technique based on adaptive dictionary have direct or some indirect relation with basic algorithm LZW [1]. LZW is lossless dictionary based adaptive compression algorithm. It is adaptive in the sense that it starts with an empty dictionary of strings and builds the dictionary during both the compression and decompression processes at encoder and decoder site. The first 256 entries for 0 to 255 characters are occupied in the dictionary or symbol table before any data is input. Because the dictionary is initialized, the next input character will always be found in dictionary. This adaptivity results in poor compression during the initial portion of each message or data, as a result, the message must be long enough for the procedure to build enough symbol frequency experience to achieve good compression over the full message. But major drawback of LZW algorithm is dictionary flushing out problem. In this paper we have overcome this problem.

The rest of the paper is summarized as follows. Section 2 presents the compression techniques. Section 3 describes the proposed system. Section 4 concludes the paper. Section 5 presents the future work.

II. COMPRESSION TECHNIQUES

There are various techniques to compression.

Compression Techniques:

a) LZW compression
b) Adaptive Arithmetic coding
c) LZ 77
d) LZ 78
e) LZFG
f) LZRW1
g) LZRW4 and so on.

In our paper we are using LZW technique.

LZW is a wonderful technique for general purpose data compression due to its simplicity, adaptivity and versatility. It is a dictionary based algorithm. It uses a dictionary or code table with some fixed entries. The maximum entries depend on per entry bit size for code table. If per entry size for code table is 10 bit, then total number of entries in code table is 2^{10} that is 1024 entries. Compression is achieved by using 256 entries onwards.

A. LZW Encoding Algorithm:

Step 1 Initialize dictionary to contain 0 to 255 single character strings.

Step 2 Read first input character prefix string ω from the data file.

Step 3 Read next input character k from the data file.

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

Step 4 End

B. LZW Decoding Algorithm:

Step 1 Read first input code and CODE=OLD code=input code with CODE=code (k), output=k, Finchar=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, Finchar=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

C. Advantages of LZW Algorithm:

1. LZW compression is the best suited technique for reducing the size of files containing more repetitive data.
2. LZW compression is fast and simple to implement and apply.
3. LZW is a lossless compression technique; none of the contents in the file are lost during or after compression.
4. LZW algorithm is efficient and adaptive because it does not need to pass the string table to the decompression code. The table can be reconstructed as it was during compression, using the input stream as data. This avoids attaching of large string translation table with the compressed data.

III. PROPOSED SYSTEM

The major drawback of LZW algorithm is dictionary flushing out problem. To overcome this problem we used multilevel dictionary or dynamic concept. So in the first level, we used 10 bit per entry in the dictionary, of which 1 bit (M.S.B bit) is used for parity bit and remaining 9 bit is used for data or character, if parity bit is set that means first level dictionary is full and second level dictionary starts. In second level dictionary. In second level dictionary we use 11 bit per entry in the dictionary, of which same process follows ie 1 bit (M.S.B bit) is used for parity bit and remaining 10 bit is used for data or character., If parity bit is set that means first level dictionary is full and third level dictionary starts and same process follows upto nth level dictionary which is required for our data file.

A. Multilevel LZW Encoding Algorithm:

Step 1 Initialize dictionary to contain 0 to 255 single character strings.

Step 2 Read first input character prefix string ω from the data file.

Step 3 Read next input character k from the data file.

a). If no such k (input exhausted): output:=code(ω):Then EXIT

b). If ωk exists in dictionary: ω: = ωk; Repeat Step 3.

c). Else If ωk not in dictionary. Then output :=code (ω)

If (new dictionary entry = current level last entry)

{

Then set M.S.B bit of k of (ωk) string equals to 1

Dictionary: = ωk;

ω: =k;

Second level dictionary starts.

Repeat Step 3.

}

else

{

Dictionary: = ωk;

ω: =k;

Repeat Step 3.

}

Step 4 End

B. Multilevel LZW Decoding Algorithm:

Step 1 Read first input character.

Left shift input character by 1, Right shift input character by 1.

And CODE=OLD code=input code with CODE=code (k), output=k, Finchar=k.

Step 2 Read next input character,

Left shift input character by 1, Results stored in carry bit. Right shift input character by 1.

If carry bit is set: next level dictionary starts.

CODE =INCODE=next input code.

If no new code: EXIT. Else:

Step 3 If CODE=code (ωk):

Left shift k of (ωk) by 1, Results stored in carry bit. Right shift (ωk) by 1.

If carry bit is set: next level dictionary starts.

stack = k

CODE: =code (ω)

Repeat Step 3

Else if CODE = code (k): output=k, Finchar=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.

IV. CONCLUSION

The work present in this paper can be summarized as:

In this paper we accomplished a system which overcome the dictionary flushing out problem from the data file so that we can achieve high compression.

V. FUTURE WORK

The proposed system can be improved so that a good compression ratio is achieved even if data is small and proposed system can be combined with any steganography technique for security.

VI. REFERENCES

[1] Terry Welch, "A technique for high performance data compression," IEEE Computer Society, 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.

[18] R. N. Horspool, and G. V. Cormack, "Construction word-based text compression algorithms", in 2nd IEEE Data Compression Conference, Snowvird, 1992.