ISSN:0975-9646

Lossless Data Compression Algorithm – A Review

Parekar P. M., Thakare S. S.

Electronics and Telecommunication Engineering, Amravati University
Government College of Engineering, Amravati (M.S.), India

***Abstract*— Data compression is more significant thing in recent world. Data compression is the science and skill of representing information in a compact form. Storage, transmission and processing of data are the integral part of information system. Enormous data demands additional resources. This leads to increase in hardware and transmission cost. Hence, resource optimization is the need of time. Instead of transmitting such data as it is, if we compress that data by applying some compression algorithm and to make sure that it will not hamper the quality of original data. Several lossless data compression algorithms are available, Lempel Ziv Marcov Chain Algorithm (LZMA) proves to be efficient in unknown byte stream compression for reliable lossless data compression which gives better compression ratio and can be hardware implementable.**

***Keywords*— Lossless Data Compression, RLE, LZ77, LZW, LZMA**

I. INTRODUCTION

Data is a combination of alphanumeric characters. This combination of elements is called a message. Data compression is nothing but the encoding of data. Data Compression algorithm compresses data file so that it takes less storage space. This is desirable for data storage and transmission application. Smaller files are desirable for data communication, because smaller the file, faster it can be transferred with less power and bandwidth. There are two types of compression techniques based on the recovery of data, lossless compression and lossy compression. But as we deal with the data, lossless compression technique is best suitable. There are various methods available to achieve lossless compression namely Run Length Encoding (RLE), Lossless predictive coding (LPC), Entropy coding and Arithmetic coding [1].

For real time data compression execution speed needed must be very high, but while implementing same algorithm over computer systems speed becomes processor specific. Therefore, in real time compression, a fast compressor with low hardware complexity is required [2]. Hence, the basic goal is to study and compare various data compression algorithm based on two categories. First, based on data compression technique, second, based on lossless data compression method. And need to select algorithm which is suited for real time data compression.

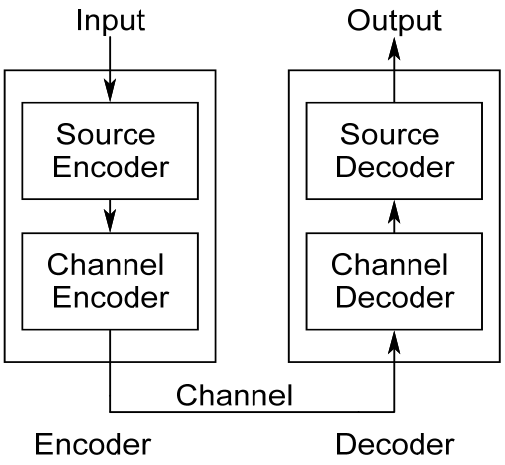

Fig. 1. Basic compressor decompressor model

Practical compression system consists of two blocks Encoder/compressor and Decoder/decompressor. Input is fed to the encoder which encodes the input to make it suitable for transmission. Decoder receives this transmitted signal and reconstructs the output. If the system is error free then output is an exact replica of input.

The encoder and decoder are made up of two blocks each. Encoder is made up of 'Source encoder' and 'channel encoder'. The source encoder removes the input redundancies while the channel encoder increases the noise immunity of source encoders output. The decoder consists of 'channel decoder' and 'source decoder'. The function of channel decoder is to ensure that the system is immune to noise. Hence, if the channel between the encoder and the decoder is noise free, channel encoder and channel decoder are omitted.

II. VARIOUS TECHNIQUES

Techniques available for data compression are: lossy compression and lossless compression. In lossy compression technique data loss occurs. After applying lossy compression algorithm it is impossible to recover exact data. Hence, when the compressed message is decoded it is not the exact replica of original message. It is not a good method of compression for critical data, such as textual data. By using this technique we can achieve higher compression ratio.

Lossless compression technique is free from loss of data. By using this technique original message can be exactly decoded. Lossless data compression works by finding repeated patterns in a message and encoding those patterns in an efficient manner. Hence, lossless data compression is also referred to as redundancy reduction. Because redundancy reduction is dependent on patterns in the message, random messages do not show such redundancy. Lossless data compression technique is ideal for text compression. By using this compression technique we achieve better compression ratio but less as compared to lossy technique.

III. Related Work

Lossless compression is usually two-step algorithms. The first step transforms the original image to some other format in which the inter-pixel redundancy is reduced. The second step uses an entropy encoder to remove the coding redundancy. The lossless decompressor is a perfect inverse process of the lossless compressor. Until 1980, most general compression scheme is statistical modelling. But in 1977 and 1978, Jacob Ziv and Abraham Lempel described a pair of compression methods using an adaptive dictionary. These two algorithms sparked a flood of new techniques that used dictionary-based methods to achieve notable new compression ratios.

Run Length Encoding (RLE) replaces data by a (length, value) pair, where "value" is the repeated value and "length" is the number of repetitions. This technique is especially successful in compressing bi-level images since the occurrence of a long run of a value is rare in ordinary gray-scale images [1]. In [1] the authors have obtained the compression ratio of 93.7% for the binary file by using RLE method. But it is not good method for gray scale image. A solution to this is to decompose the gray-scale image into bit planes and compress every bit-plane separately. Efficient run-length coding method is one of the variations of run length coding.

In [3], [4], [5], the authors used the concept of selective Huffman coding for high speed data compression and decompression. Huffman coding is based on the concept of mapping an alphabet to a different representation composed of strings of variable size such that symbols with a high probability of occurrence have a smaller representation than those that occur less often. There are two variations of Huffman encoding: static and dynamic. Static Huffman encoding uses a look-up table that stores a pre-defined frequency for each character in the alphabet. Dynamic Huffman encoding calculates the actual frequency of characters based on the data in the file to be compressed. Once the frequency data has been determined, the two variations are identical. The two elements with the lowest weights are selected. These two elements are inserted as leaf nodes of a node with two branches. The frequencies of the two elements selected are then added together and this value becomes the frequency for the new node. The algorithm continues selecting two elements at a time until a Huffman tree is complete with the root node having a frequency of 100 %.

The hardware implementations for the LZ77 encoders and Huffman encoders that form the basis for a full hardware implementation of a GZIP encoder are done in [4]. It will be possible to merge these designs to build an encoder that is capable of generating compressed files that may be decompressed using a standard implementation of GZIP [4]. LZ77 encoding algorithm is the first simple compression algorithm described by Ziv and Lempel in 1977 is commonly referred to as LZ77. The dictionary consists of all the strings in a window into the previously read input stream. When new groups of symbols are being read in, the algorithm searches for matches with strings found in the previous data already read in. Then the matches are encoded as pointers and sent as the output. LZ77 encoding is simple and faster. LZ77 is effective only when the input data is highly redundant or repetitive. In [4], the authors used LZ77 algorithm.

In [5], the authors have proposed data compression algorithms based on Huffman and Lempel- Ziv techniques and compare the efficiency of different algorithms in a wireless sensor network scenario. The previous compression algorithm is compared with the minimum variance Huffman compression algorithm, adaptive Huffman coding algorithm and variations of Huffman coding algorithm. By cascading LZW and Arithmetic coding technique a higher compression ratio was achieved [5]. From the analysis it is found that Huffman coding techniques are better than LZW techniques for achieving higher compression ratio in wireless sensor network. Among Huffman coding techniques and its variations, minimum variance Huffman coding technique performs better than others [5].

In [6] the authors have designed the LZW compression algorithm for the prediction of new compression ratio. This helps the storage system decide early on whether or not to continue with the compression. The proposed prediction algorithm can be used with any hardware implementations of the LZW algorithm and help them to improve their compression speed, at the expense of a negligible decrease in compression ratio. In [6], [7], the authors have used the concept of LZW compression method. LZW is a dictionary based data compression that compresses content character by character. These characters are combined to form a string. A special code is assigned to new strings and strings are added to the dictionary. When the string is repeated it can be assigned with that code.

In [8], a new two-stage hardware architecture that combines the features of both parallel dictionary LZW (PDLZW) and approximated adaptive Huffman (AH) algorithms is implemented in [8]. An ordered list instead of the tree- based structure is used in the AH algorithm for speeding up the compression data rate. The resulting architecture in [8] shows that it not only outperforms the AH algorithm at the cost of only one-fourth the hardware resource but it is also competitive to the performance of LZW algorithm (compress). In addition, both compression and decompression rates of the proposed architecture in [8] are greater than those of the AH algorithm even in the case realized by software.

A simple and efficient data compression algorithm particularly suited to be used on available commercial nodes of a WSN, where energy, memory and computational resources are very limited. The algorithm designed in [9]

has compared with S-LZW and achieve higher compression ratios than S-LZW, despite a lower memory acquisition and a less computational effort. This algorithm outperforms gzip and bzip2.

In [10], authors proposed the implementation of Lampel Ziv Markov Chain (LZMA) algorithm. LZMA is the recent variant of LZ coding. The Lempel Ziv Markov Chain algorithm which is used in 7zip (an application used to compress files) was proved to be effective in unknown byte stream compression for reliable lossless data compression. And this is used to achieve high compression ratio. It is based on LZ77 and uses a delta filter, a sliding dictionary algorithm and a range encoder. The delta filter is used to make the data better suited for compression with the sliding window dictionary. The output from the delta filter is in the form of difference from previous data. The sliding dictionary algorithm is then applied to the output of the delta filter. Finally, the output from the sliding dictionary is used in a range encoder which encodes the symbols with numbers based on the frequency at which the symbols occur.

IV. Conclusions

Among the various lossless data compression algorithms, LZMA algorithm proves to be superior over its counterparts. LZMA algorithm improves the efficiency of compression on text. It is suitable for real time data compression. If this algorithm is implemented on hardware then its execution speed is compatible with the arrival of real time data. Hardware implementation of any compression algorithm is important for power, timing and area analysis.

Acknowledgment

I express my sense of gratitude and sincere regards to my guide S. S. Thakare. I would like to thanks all the staff members of Electronics and Telecommunication Department.

References

[1] P.Yellamma and Dr.NarasimhamChalla “Performance Analysis Of Different Data Compression Techniques On Text File”, *International Journal of Engineering Research & Technology (IJERT)* ISSN: 2278-0181 Vol. 1 Issue 8, October – 2012.

[2] Armein Z. R. Langi “An FPGA Implementation of a Simple Lossless Data Compression Coprocessor”, 2011 *International Conference on Electrical Engineering and Informatics* 17-19 July 2011, Bandung, Indonesia.

[3] Satish Kannale , KavitaKhare , Deepak Andore and MallikarjunMugli “FPGA implementation of selective Huffman coding for high speed data compression and decompression”, *World Journal of Science and Technology* 2011, 1(8): 89-93 ISSN: 2231 – 2587

[4] Suzanne Rigler, William Bishop, Andrew Kennings “FPGA-Based Lossless Data Compression using Huffman and LZ77 Algorithms”, 2007 *IEEE*.

[5] S. Renugadevi , P.S. NithyaDarisini “Huffman and Lempel-Ziv based Data Compression Algorithms for Wireless Sensor Networks”, 2013 *International Conference on Pattern Recognition, Informatics and Mobile Engineering (PRIME)* February 21-22.

[6] AlirezaYazdanpanah , Mahmoud Reza Hashemi “A New Compression Ratio Prediction Algorithm for Hardware Implementations of LZW Data Compression”, 2010 *IEEE*.

[7] Md. Mamun, Md. ArifSobhan, Ahmad Ashrif, A. Bakar and Hafizah Hussain “Hardware Approach of Lempel-Ziv-Welch Algorithm for Binary Data Compression”, *World Applied Sciences Journal 22* (1): 133-139, 2013 ISSN 1818-4952 © IDOSI Publications, 2013.

[8] Ming-Bo Lin, Jang-Feng Lee and Gene Eu Jan “A Lossless Data Compression and Decompression Algorithm and Its Hardware Architecture”, *IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION (VLSI) SYSTEMS*, VOL. 14, NO. 9, SEPTEMBER 2006.

[9] Francesco Marcelloni, Massimo Vecchio “A Simple Algorithm for Data Compression in Wireless Sensor Networks”, *IEEE communications letters*, vol. 12, no. 6, June 2008.

[10] E.JebamalarLeavline, D.Asir Antony Gnana Singh “Hardware Implementation of LZMA Data Compression Algorithm” *International Journal of Applied Information Systems (IJAIS)* – ISSN : 2249-0868 Foundation of Computer Science FCS, New York, USA Volume 5– No.4, March 2013