International Journal of Electronics and Communication Engineering.
ISSN 0974-2166 Volume 4, Number 2 (2011), pp. 233-241

http://www.irphouse.com

An Effective Two Stage Text Compression and Decompression Technique for Data Communication

[1]Raja P. and [2]Saraswathi D.

Assistant Prof., Dept. of Electronics and Communication Engg.,
[1]Sri Manakula Vinayagar Engineering College, Madagadipet, Puducherry, India
[2]Manakula Vinayagar Institute of Technology, India
E-mail: [1]rajashruthy@gmail.com, [2]vkvarunee@gmail.com

Abstract

Data compression is a method of encoding rules that allows substantial reduction in the total number of bits to store or transmit a file. Currently, two basic classes of data compression are applied in different areas .One of these is lossy data compression, which is widely used to compress image data files for communication or archives purposes. The other is lossless data compression that is commonly used to transmit or archive text or binary files required to keep their information intact at any time. In this paper, a two level text compression and decompression techniques for lossless data compression is proposed. The features of both LZW (Lempel-Ziv-Welch) and Huffman algorithms are combined to improve the compression ratio. The main advantage of this combined algorithm is that the percentage of data reduction increases more than 5% compared to the existing text compression techniques.

Key terms: Compression, decompression, LZW, Huffman coding, two stage compression

Introduction

Data Compression is the process of making numerical or other information represented in a form suitable for processing by computer more compact[1]. More concisely, data compression involves the identification and removal of redundant and unnecessary elements of source data. The main goal of data compression is to reduce the number of bits used to store or transmit information [2]. A mechanism to encode the data in order to reduce the redundancy could possibly provide a 30-80% reduction in the size of data in a large commercial database. Data compression provides additional benefits such as reduction in the cost of backup and recovery in computer

systems, increased security and efficiency in search operations on compressed index structure of files. In recent years, the demand for data compression and the need to develop faster and more efficient compression methods have increased considerably due to the increased use of data compression within scientific and statistical database, document delivery systems, and communications networks. Efficient data encoding schemes can also be very valuable to the design and performance of supercomputers and in enhancing the performance of database machines.

Text Compression and Decompression

Text compression is the process of encoding text information using fewer bits. Text decompression is the process of restoring compressed data back into a form in which it is again useful. Fig 1 shows the flow chart of text compression and decompression process.

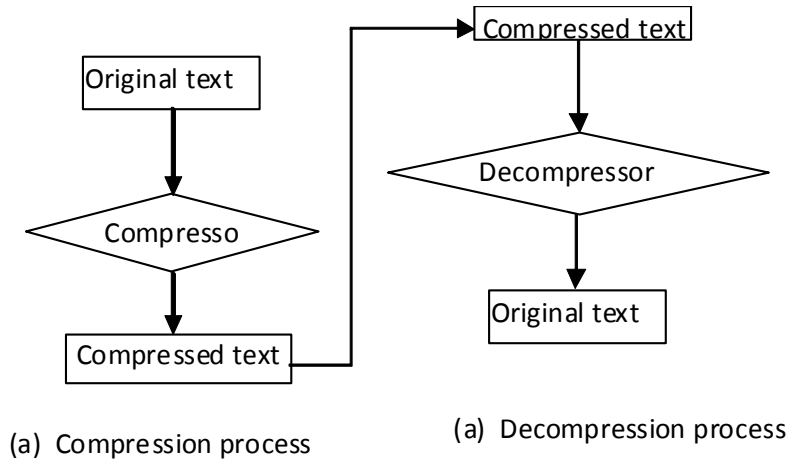

Figure 1: Flow chart of text compression and decompression process.

The original text has lot of redundancy bits and compression removes this redundancy by the hardware called as compressors. Thus the storage size required for the compressed text is also reduced. In the decompression end, the original text is retrieved back as shown in Fig. 1.

In terms of storage, the capacity of a storage device can be effectively increased with methods that compress a body of data on its way to a storage device and decompress it when it is retrieved [3]. At any given time, the ability of the internet to transfer data is fixed. Thus, if data can effectively be compressed wherever possible, significant improvements of data throughput can be achieved [4]. Many files can be combined into one compressed document which is easier for sending. In computer graphics, reducing the size of a block of graphics data so that more information can fit in a given physical storage space.

The LZ Compression Technique

The LZ (Lempel Ziv) technique proposed by Ziv and Lempel [ZIV77] for data compression involves two basic steps which are: (i) Parsing and (ii) Coding. In the *parsing* step, string of symbols were split into substrings of variable length according to certain rules [3]. In the *coding* step, each substring was coded sequentially into a fixed length code [4].

This LZ compression technique involves transformation of variable length sub-strings to fixed length codewords. The codewords should be as short as possible so as to achieve maximum compression. The buffer size determines the codeword size and is crucial since it affects the compression ratio.

Ziv and Lempel proposed the LZ77 scheme, also known as the LZ l-type technique, for lossless data compression. In this family of algorithms, the repeating phrases are replaced with pointers to where they have occurred earlier in the text. For decompression, each pointer is replaced with the already decoded text that it points to and thus could be done very fast. The three factors that differentiate the various versions of LZ are:

- Whether there is a limit to how far back a pointer can reach,
- Which substrings within this limit may be the target of a pointer, and
- How the pointer and the length are encoded that form the codewords of the compressed text.

The reach of a pointer into earlier text may be unrestricted (growing window), or it may be restricted to a fixed-size window of the previously coded characters. The choice of substrings can either be unrestricted or limited to a set of phrases chosen according to some heuristic. The LZ78- or the LZ2-type scheme proposed by Ziv and Lempel in 1978 is different from LZ77 in that a dictionary of phrases distinct from the input data is independently maintained, and compression is achieved by replacing repeating phrases by index pointers to the dictionary [6].

The Expansion Problem

Software implementation and testing of the LZ algorithm on a number of text files showed that the direct application of the LZ algorithm does not perform well in all cases. In some cases there was expansion rather than compression. This is known as compression problem. Methods to overcome the expansion problem led to the development of the LZW Compression Technique [12].

The LZW Compression Technique

The LZW data compression algorithm is a powerful technique for lossless data compression that gives high compression efficiency for text as well as image data [6], [7]. However, conventional LZW algorithm requires quite a lot of time for adjusting and searching the dictionary. To improve this, a variety of alternatives of LZW were proposed. The DLZW (dynamic LZW) algorithm uses a hierarchy of dictionaries with successively increasing word widths, so the data compression time of DLZW is dominated by indexing times of child dictionaries. But due to the variable word width of the dictionary set, using hash function for faster searching and updating dictionary

is impossible as a result the compression time of DLZW is still long for real-time application[7]. The WDLZW (word-based DLZW) implements LRU (least recently used) policy to update dictionaries, however the manipulation of the dictionary is too complicated [13].

DLZW and WDLZW improve LZW algorithm in the following ways. First, it initializes the dictionary with different combinations of characters instead of single character of the underlying character set. Second, it uses a hierarchy of dictionaries with successively increasing word widths. Third, each entry associates a frequency counter. That is, it implements LRU policy. It was shown that both algorithms outperform LZW. However, it also complicates the hardware control logic

Huffman Coding

Huffman coding is widely used technique for data compression. It can save nearly from 20% to 90% of the amount of storage or the communication channel bandwidth needed, depending on the characteristics of the input being compressed. No information loss occurs after decoding [15].

Huffman's greedy algorithm uses a table of frequencies of occurrence of each input symbol to build up an optimal way of representing each symbol by a binary string. Using this, the encoder assigns a variable length binary string to each fixed length input symbol such that the input symbols with higher frequency have shorter lengths. For example, consider coding six symbols [a, b, c, d, e, f]. Assume their frequencies (repetitions) are [45, 13, 12, 16, 9, and 5]. The word which repeats often is coded with fewer numbers of bits. Their Huffman codes are [0, 101, 100, 111, 1101, and 1100].

Existing Two-Stage Compression Technique

The two-stage data compression architecture combines features from both PDLZW and Adaptive Huffman (AH) algorithms. In order to reduce the hardware cost, a simplified DLZW architecture called parallel dictionary LZW (PDLZW) was introduced. The resulting architecture shows that it outperforms the AH algorithm in most cases and requires only one-fourth of the hardware cost of the AH algorithm. In addition, its performance is competitive to the compress utility in the case of the executable files. Furthermore, both compression and decompression rates are greater than those of the AH algorithm even in the case realized by software [13].

The AH algorithm as the second stage not only increases the performance of the PDLZW algorithm but also compensates the percentage of data reduction loss due to the anomaly phenomenon occurred in the PDLZW algorithm[15]. In addition, the proposed scheme is actually a parameterized compression algorithm because its performance varies with different dictionary-set sizes but the architecture remains the same [11].

Compression End

The Compression model is the two stage compression process as shown in Fig 2. The first stage of this model, compression is achieved by replacing the input string into

fixed length codes by PDLZW algorithm. In the second stage, the fixed length codes are replaced into variable length codes by AHDB algorithm.

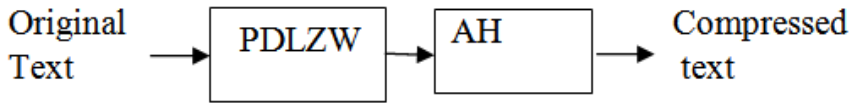

Figure 2: Two stage Compression.

Decompression end

The two stage decompression process is shown in Fig 3. In the first stage, decompression is achieved by replacing the input string into variable length codes by AHDB algorithm. In the second stage, the variable length codes are replaced into fixed length codes by PDLZW algorithm. The data compression efficiency by using PDLZW and AHDB in successive stages can be achieved up to 41.50%.

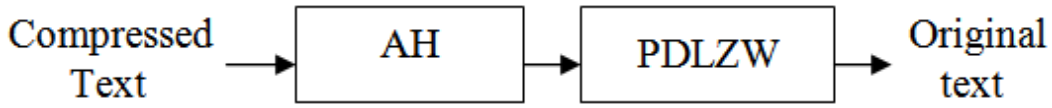

Figure 3: Two stage decompression.

Existing Model Performance

It can be observed from table 1 that fourteen file indexes of different sizes have been taken and various compression techniques have been combined with PDLZW technique and the average is taken. Of that the last one yields the maximum efficiency i.e., PDLZW+AHDW gives the efficiency of around 41.50% which is highest compared with that of others.

Table 1: Compression efficiency of various algorithms.

File index	Text files			
	AH	PDLZW+		
		AHAT	AHFB	AHDB
1	42.53	35.70	39.59	42.64
2	39.38	32.53	37.12	40.07
3	43.00	39.93	37.78	40.98
4	40.02	40.18	38.88	41.85
5	33.70	30.67	36.14	39.42
6	35.96	31.94	43.71	46.27

7	41.52	39.85	39.58	42.51
8	37.29	30.65	37.47	40.40
9	42.03	32.26	38.64	41.83
10	41.14	30.04	37.45	40.63
11	40.07	26.73	37.60	40.77
12	36.86	25.93	37.13	40.25
13	36.97	30.75	38.73	41.56
14	42.79	39.77	38.72	41.85
Avg	39.50	33.35	38.47	41.50

The Proposed Model for Two Stage Compression Technique

The new two stage compression model is shown in Fig 4. In the first stage, compression is achieved by replacing the input text into variable length codes by the Huffman algorithm. In the second stage, the variable codes are replaced into fixed length codes by LZW algorithm.

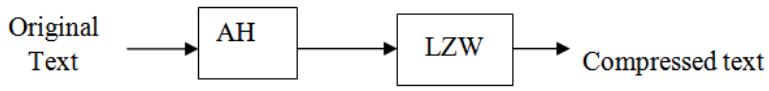

Figure 4: Two stage Compression.

The new two stage decompression model is shown in Fig 5. In the first stage, decompression is achieved by replacing the input text into fixed length codes by the LZW algorithm. In the second stage, the fixed length codes are replaced into variable length codes by Huffman algorithm.

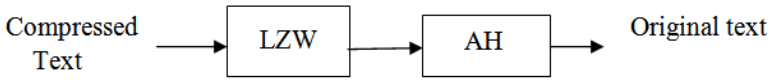

Figure 5: Two stage Decompression.

Proposed model performance

Proposed model has an efficiency of 46.19%, which is 5% higher than the existing model. In the existing model the features of PDLZW and AHDB are combined and the efficiency is found to be 5% more than other combinations, i.e. fixed length codes are converted into variable length codes. Whereas in the proposed model the order of compression stages are changed i.e. variable length codes to fixed length codes (AH+LZW) and an efficiency of 46.19% is obtained which is 5% percent more than the existing model.

Table 2: Compression efficiency for various Algorithms.

File Index	File Size (KB)	Huffman + LZW Compression	% of compression
1	2	1	50
2	6	5	16.6
3	11	8	27.7
4	21	17	19.9
5	27	9	66.6
6	31	22	30
7	36	26	27.7
8	48	34	30
9	51	20	60
10	92	54	78.2
11	122	77	36.8
12	168	84	50
13	201	83	58.7
14	357	160	55.5
15	511	121	76.32
16	820	200	75.60
17	1156	283	75.77
Average			46.19

Results and Discussions

This paper has discussed the performance evaluation of two-stage lossless compression scheme for effective data communication. The compression performance results achieved for the multi-stage scheme was higher, as expected. The proposed model which is the intrinsic feature of the both LZW and AH implemented in "C" language and this model is tested for various file size as an input text. Since the proposed model is a text based compression technique which has to be a lossless one. More emphasis has been given to text compression. The compression ratio for existing model is shown in Fig 6. From the Fig. 6, we can see that the compression efficiency by using PDLZW and AHDB in successive stages can be achieved up to 41.50%.

However, the performance of the proposed schemes in terms of processing time is rather surprising, especially as it was significantly faster despite the addition of an additional stage to the LZW algorithm. Experimental results show that the proposed two stage (LZW and AH) algorithm has best compression ratio for text data.From the Fig4.2 we can notice that the compression efficiency by using AH and LZW in successive stages can be achieved up to 46.19%.

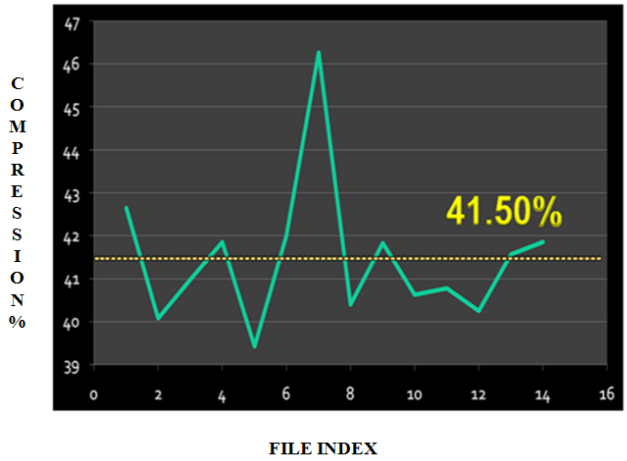

Figure 6: Existing Model Performance [6].

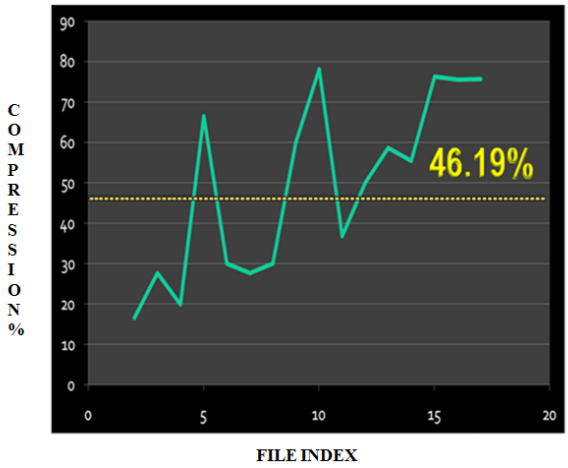

Figure 7: Proposed model Performance.

References

[1] Ibrahim Akman, Hakan Bayindir, Serkan Ozleme, Zehra Akin, and Sanjay Misra "Lossless Text Compression Technique using Syllable Based Morphology," International Arab Journal of Information Technology, Vol. 8, No. 1, January 2011.

[2] Martha R. Quispe-Ayala , Krista Asalde-Alvarez , Avid Roman-Gonzalez "Image Classification Using Data Compression Techniques," IEEE 26-th Convention of Electrical and Electronics Engineers in Israel, 2010.

[3] N. K. Kasumov "The universal coding method in the data compression algorithm", Automatic Control and Computer Sciences, Springer Link, Volume 44, Number 5, 279-286, 2010.

[4] Roi Blanco, Alvaro Barreiro “Probabilistic static pruning of inverted files”, Volume 28 Issue 1, January 2010.
[5] Ming-Bo Lin Yung-Yi Cha “A New Architecture of a Two-Stage Lossless Data Compression and Decompression Algorithm,” IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol.17, no. 9, pp 1297 – 1303, August 2009.
[6] N. Sriraam and C. Eswaran “Performance Evaluation of Lossless Two-stage Compression Schemes for EEG Signal,” International Journal of Information and Communication Engineering, vol1, pp. 89-92, 2005
[7] M.-B. Lin, “A parallel VLSI architecture for the LZW data compression algorithm,” J. VLSI Signal Process., vol. 26, no. 3, pp. 369–381, Nov. 2000.
[8] J. L. Núñez and S. Jones, “Gbit/s lossless data compression hardware,” IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 11, no. 3, pp. 499–510, Jun. 2003.
[9] T. A. Welch, “A technique for high-performance data compression,” IEEE Comput., vol. 17, no. 6, pp. 8–19, Jun. 1984.
[10] H. Park and V. K. Prasanna, “Area efficient VLSI architectures for Huffman coding,” IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process. vol. 40, no. 9, pp. 568–575, Sep. 1993.
[11] J. Jiang and S. Jones, “Word-based dynamic algorithms for data compression,” Proc. Inst. Elect. Eng.-I, vol. 139, no. 6, pp. 582–586, Dec.1992.
[12] B. Jung and W. P. Burleson, “Efficient VLSI for Lempel-Ziv compression in wireless data communication networks,” IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 6, no. 3, pp. 475–483, Sep. 1998.
[13] J. Ziv and A. Lempel, “A universal algorithm for sequential data compression,” IEEE Trans. Inf. Theory, vol. IT-23, no. 3, pp. 337–343, Mar.1977.
[14] M. Gonzalez and J. A. Storer, “Parallel algorithms for data compression.” J. ACM, vol. 32, no. 2, pp. 344-373, Sep1991
[15] G. Langdon, “An introduction to arithmetic coding,” IBM J. Res. Develop., vol. 28, no. 2, pp. 135–149, Mar. 1984.
[16] H. K. Reghbati, "An Overview of Data Compression Techniques," Computer, Vol. 14, No. 4, pp. 7 1-76, Apr. 1981.
[17] F. Rubin, "Experiments in Text File Compression," Comm. ACM, Vol. 19, No. 11, pp. 617-623, Nov. 1976.
[18] M. Pechura, "File Archival Techniques Using Data Compression," Comm. ACM, Vol. 25, No. 9, pp.605-609, Sept. 1982.
[19] M. Rodeh, V. R. Pratt, and S. Even, "Linear Algorithm for Data Compression via String Matching," J. ACM, Vol. 28, No. 1, pp. 16-24. Jan. 1981.