Malaysian Journal of Mathematical Sciences 7(S): 117-129 (2013)
Special Issue: The 3rd International Conference on Cryptology & Computer Security 2012 (CRYPTOLOGY2012)

MALAYSIAN JOURNAL OF MATHEMATICAL SCIENCES

Journal homepage: http://einspem.upm.edu.my/journal

A New Efficient Analytically Proven Lossless Data Compression for Data Transmission Technique

[1]M. A. Daud and [2]M. R. K. Ariffin

[1]Al – Kindi Cryptography Research Laboratory, Institute for Mathematical Research, Universiti Putra Malaysia, 43400 UPM Serdang, Selangor, Malaysia

[2]Mathematics Department, Faculty of Science, Universiti Putra Malaysia, 43400 UPM Serdang Selangor, Malaysia

E-mail: azlanmath@gmail.com and rezal@putra.upm.edu.my

*Corresponding author

ABSTRACT

A new lossless data compression method for data transmission is proposed. This new compression mechanism does not face the problem of mapping elements from a domain which is much larger than its range. Our new algorithm sides steps this problem via a pre-defined code word list. The algorithm has fast encoding and decoding mechanism and is also proven analytically to be a lossless data compression technique.

Keywords: lossless data compression method, pre-defined code word list, coding techniques.

1. INTRODUCTION

Compression is the processes of reducing the size of a file by doing some alteration to the structure. In real world applications, compression is very useful because it helps to reduce the consumption of expensive resources, such as memory space, total time for data transfer over network and communication costs by using available bandwidth effectively.

There are 2 categories. The first category is known as the lossless data compression technique. By this technique compressed data can be decompressed without any loss of data. This means that, the information

after being decompressed does not change from its original structure before compression. It is also known as reversible compression since the original data is reconstructed by decompression process. An example is the ZIP file mechanism. Since the original data becomes smaller, it is easy to be transmitted trough today's public bandwidth. The second category is lossy data compression techniques. Through this technique the decompression process of compressed data produces results with loss of some information. This compression technique is called irreversible compression since it is not possible to reconstruct 100% the original message during the decompression process.

Lossless data compression quality usually depends on the following criteria's:

1. Total time for compression
2. Total time for reconstruction
3. Size of the compressed data when compared to the original data

The above list can be measured through entropy and compression ratio.

Definition 1 (Entropy)

Let M be a random message (random experiment) with outcomes $M = \{m_1, \dots, m_n\}$ having probabilities $p_1, \dots, p_n$.

The entropy of the message M is given by

$$H(M) = -\sum_{m \in M} p(m) \log_2 p(m)$$

$H(M)$ measures the uncertainty of the outcome of M. Since we know that $\log_2 p(m) \leq 0$, we have $H(M) \geq 0$. We also define $0 \log_2 0 = 0$ since $\lim_{x \to 0} m \log_2 m = 0$.

Remark 1

Whenever average length of the compressed data is approximately equal to the entropy then we can conclude that the corresponding algorithm has a good compression rate.

Definition 2 (Compression Ratio)

Compression ratio is defined as

$$\text{Compression ratio, } CR = \frac{\text{Uncompressed size} - \text{Compressed Size}}{\text{Uncompressed size}}$$

Remark 2

From Definition 2, it implies that if $C_R \to 1$ (i.e. Compressed size $\to 0$), the algorithm has an excellent compression rate.

Many proposed algorithms are defined as lossless data compression. For lossless data compression, main stream techniques are based on entropy coding (aka statistical coding). Notable are the Huffman (Roman (1997)) and arithmetic coding algorithms (Langdon (1979)). An entropy based coder must work in conjunction with a modeler that estimates that probability of each possible event at each point in the coding. The probability model need not describe the process that generates the data; it merely has to provide a probability distribution for data items. The probabilities do not even have to be particularly accurate, but the more accurate they are, the better the compression will be (Hellebrand and Wurtenberger (2002)).

Another lossless compression technique is based on dictionary coding. Lempel-Ziv is an example of dictionary coding (Ferreira, Oliveira and Figueiredo (2009)). In an attempt to produce efficient algorithms, researchers have merged both the entropy and dictionary coding techniques. The Alternating Run Length (ARL) coding is one such case. It is based on the Frequency Directing Run-length (FDR) coding efficiently exploits the fact that typical test data contain shorter runs of zeros or ones with higher frequencies than longer runs and prior to producing the output it refers to a coding table. The ARL encoding algorithm is advantageous when it does not require any extra information to distinguish between runs of zeros and ones (Hellebrand and Wurtenberger (2002)).

In this work, we produce a new lossless type compression method that models after the dictionary type coding technique. It is based on a coding table that has a one-to-one mapping of a list of integers less than a predetermined data size (i.e. $1,2,\dots,n$) to its corresponding codeword. The algorithm is also proven analytically to be unique, hence overruling the possibility of decoding failure.

This paper will be structured as follows. In Section 2, we will introduce the new compression encoding algorithm based on the dictionary coding strategy. It is of lossless data compression type. In this section we will explain our compression algorithm details and we also conduct an example. Following this in Section 3, we will prove that the decoding

process is unique. Hence, ensuring a 100% success rate in restructuring the compressed data back into its original form (i.e. lossless). An example will be provided in Section 4. In Section 5, discussion on the compression ratio will be put forward via examples. Discussion on transmission advantages is in Section 6. Finally we conclude in Section 7.

2. HOW THE COMPRESSION ALGORITHM WORKS

In this section we begin with observing the following code word. The right column (binary code) is the code word for its counterpart residing within the same row in the left column (number).

TABLE 1: Binary codes to represent the integers

Number	Binary Code
1	1
2	10
3	100
4	1000
5	10000
6	100000
⋮	⋮
$n-1$	$1[(n-1)-1]0's$
n	$1[(n-1)]0's$

Definition 3 (Compression Algorithm)

Prior to the encoding process, to ensure correct decoding the size of the original data, n is known to both the encoder and decoder. We denote $\|b\|$ to be the length of the corresponding data string $b = \{0,1\}^n$. For $j = 1,2,3,\ldots$ we define the j-th data string as $b_j = \left(2^{\|b_{j-1}\|} - 1\right) - b_{j-1}$. Given a data string input b_0, we will do the following

i. Convert b_0 to its decimal value.

ii. Compute, $b_1 = \left(2^{\|b_0\|} - 1\right) - b_0$

iii. Code the difference between the length of $\|b_0\|$ and $\|b_1\|$ as w_1 (refer Table 1)

iv. Continue the loop $b_j = \left(2^{\|b_{j-1}\|} - 1\right) - b_{j-1}$ for $j = 1,2,3,\ldots,k$ until $0 \leq b_j \leq 3$ (observe that $\|b_j\| = 2$). In each loop a codeword w_j will

be produced based on the difference between the length of $\|b_{j-1}\|$ and $\|b_j\|$. Observe that the values of $b_j$are strictly decreasing, and as soon as it reaches $0 \leq b_j \leq 3$ the algorithm will terminate.

v. From the codeword list $\{w_1, w_2, \dots, w_{k-1}, w_k\}$ we will append b_k at the end of the codeword to gain $m = \{w_1, w_2, \dots, w_{k-1}, w_k, b_k\}$. Once again observe that $\|m\| = n$. Then, focus on the last codeword w_k b_k will be shifted to the left according to the number of zeros in w_k. The result is compressed data denoted by m_c.

vi. The encoder will then send the compressed data m_c. Notice that the zero's within w_k is excluded in the corresponding sequence which constructs m_c. Hence, $\|m_c\| < \|m\|$.

3. UNIQUENESS OF THE DECODING PROCESS

Proposition 1 (Decompression Algorithm)

The following decoding process of an encoded information by Definition 3 is unique.

1. Expand m_c to the original size $\|m\|$ by shifting back b_k to the right by padding in zero's until we have $\|m_c\| = \|m\|$. To decode we have to decide where each code begins and ends, since they do not have the same length. During the encoding process we utilized the codeword list as given by Table 1. As a result, we only need to scan through the input string of m_c from right to left until we recognize the first codeword. Then, we are able to determine the corresponding value and start looking for the next codeword. Observe that from Table 1 all cases will begin with 0 from the right and stop with 1 on the left.

2. Excluding b_k, start by extracting the codeword from the LSB of m_c. Translate the codeword from Table 1.

3. Compute, $b_{k-1} = \left(2^{\|p_0\|} - 1\right) - b_k$ where $\|p_0\| = \|w_k\| + \|b_k\|$.

4. Next, compute, $b_{k-2} = \left(2^{\|p_1\|} - 1\right) - b_{k-1}$ where $\|p_1\| = \|w_{k-1}\| + \|p_0\|$.

5. Continue until $b_{k-i} = \left(2^{\|p_{i-1}\|} - 1\right) - b_{k-i+1}$, where $i = 1,2,3, \dots, k$. The original data is b_0.

Proof. The compression algorithm consists of a sequence of subtractions. Assume that the decoding process is not unique, then for unique b_{k-i+1} and $\|p_{i-1}\|$ there exists $b_{k-i} = \left(2^{\|p_{i-1}\|} - 1\right) - b_{k-i+1}$ and $b'_{k-i} = \left(2^{\|p_{i-1}\|} - 1\right) - b_{k-i+1}$ such that $b_{k-i} \neq b'_{k-i}$. This would imply that $b_{k-i+1} \neq b_{k-i+1}$. This is a contradiction. Hence, assumption is false and the decoding process provides a unique output.■

4. EXPERIMENTAL EXAMPLES

Example 4.1

Our example focus on encoding process, decoding process is reversing the encoding process. This example will show you how our algorithms work to encode and decode easily.

Suppose we have an input string
$b_0 = \{11111011110101100000000000000011\}$, $\|b_0\| = n = 32$ and $b_0 = 4260036611$.

Compression process

1. Compute, $b_1 = \left(2^{\|b_0\|} - 1\right) - b_0 = (2^{32} - 1) - 4260036611 = 34930684$, $\|b_1\| = 26$

2. Compute the difference between $\|b_0\|$ and $\|b_1\|$, then our first codeword is $w_1 \rightarrow \|b_0\| - \|b_1\| = 32 - 26 = 6 \rightarrow 100000$.

3. Compute, $b_2 = \left(2^{\|b_1\|} - 1\right) - b_1 = (2^{26} - 1) - 34930684 = 32178179$, $\|b_2\| = 25$ and $w_2 \rightarrow \|b_1\| - \|b_2\| = 26 - 25 = 1 \rightarrow 1$.

4. Compute, $b_3 = \left(2^{\|b_2\|} - 1\right) - b_2 = (2^{25} - 1) - 32178179 = 1376252$, $\|b_3\| = 21$ and $w_3 \rightarrow \|b_2\| - \|b_3\| = 25 - 21 = 4 \rightarrow 1000$.

5. Compute, $b_4 = \left(2^{\|b_3\|} - 1\right) - b_3 = (2^{21} - 1) - 1376252 = 720899$, $\|b_4\| = 20$ and $w_4 \rightarrow \|b_3\| - \|b_4\| = 21 - 20 = 1 \rightarrow 1$.

6. Compute, $b_5 = \left(2^{\|b_4\|} - 1\right) - b_4 = (2^{20} - 1) - 720899 = 327676$, $\|b_5\| = 19$ and $w_5 \rightarrow \|b_4\| - \|b_5\| = 20 - 19 = 1 \rightarrow 1$.

7. Compute, $b_6 = \left(2^{\|b_5\|} - 1\right) - b_5 = (2^{19} - 1) - 327676 = 196611$, $\|b_6\| = 18$ and $w_6 \rightarrow \|b_5\| - \|b_6\| = 19 - 18 = 1 \rightarrow 1$.

8. Compute, $b_7 = \left(2^{\|b_6\|} - 1\right) - b_6 = (2^{18} - 1) - 196611 = 65532$, $\|b_7\| = 16$ and $w_7 \rightarrow \|b_6\| - \|b_7\| = 18 - 16 = 2 \rightarrow 10$.

9. Compute, $b_8 = \left(2^{\|b_7\|} - 1\right) - b_7 = (2^{16} - 1) - 65532 = 3$, $\|b_8\| = 2$ and $w_8 \rightarrow \|b_7\| - \|b_8\| = 16 - 2 = 14 \rightarrow 10000000000000$.

10. Then, we have the codeword $m = \{w_1 w_2 w_3 \cdots w_7 w_8 b_8\} = \{10000011000111101000000000000011\}$, $\|m\| = 32$.

11. Next, shift b_8 to the left base on how much zero's in w_8, now we have the compressed data given by $m' = \{1000001100011110111\}$, $\|m'\| = 19$.

Decompression process

Given $m' = \{1000001100011110111\}$, $\|m'\| = 19$ and $\|b_0\| = \|m\| = n = 32$.

1. Expand the code word $\|m'\|$ to the original size $\|m\|$ by a shift back b_8 to the right, $m = \{10000011000111101000000000000011\}$.

2. $w_8 \rightarrow 10000000000000 \rightarrow 14$, $\|p_0\| = \|w_8\| - \|b_8\| = 14 + 2 = 16$.

3. Compute, $b_7 = \left(2^{\|p_0\|} - 1\right) - b_8 = (2^{16} - 1) - 3 = 65532$.

4. Then, continue the loop until $b_0 = 4260036611$, $\|p_7\| = 32$.

With this example we provide empirical evidence of our compression method showing its execution as a lossless compression method.

5. COMPRESSION RATIO

From example 4.1, the original message $n = 32$ bits length and compressed message = 19 bits, then from Definition 2 its compression ratio, CR = 0.40625. In this section we provide more experimental results for discussion.

We took data of different sizes: 8-bits, 16-bits, and 32-bits and compared their compression ratio by using our algorithm coding scheme for data compression. Suppose we want to compress 8-bit messages "10111101", "00011110", "11000111" and "00000011".

Example 5.1

Change to decimal, $b_0 = 10111101 \rightarrow 189$

TABLE 1: Compression process for $n = 8$-bits data sample

b_i	w_i	Codeword before shift	Codeword after shift	CR
$[2^8 - 1] - 189 = 66$	1	11100010	11110	0.375
$[2^7 - 1] - 66 = 61$	1			
$[2^6 - 1] - 61 = 2$	1000			
Remainder = 2	10			

Example 5.2

Change to decimal, $b_0 = 00011110 \rightarrow 30$

TABLE 2: Compression process for $n = 8$-bits data sample

b_i	w_i	Codeword before shift	Codeword after shift	CR
$[2^5 - 1] - 30 = 1$	100	00010001	000101	0.25
Remainder = 1	01			

Example 5.3

Change to decimal, $b_0 = 11000111 \rightarrow 199$

TABLE 3: Compression process for $n = 8$-bits data sample

b_i	w_i	Codeword before shift	Codeword after shift	CR
$[2^8 - 1] - 199 = 56$	10	10100100	10100100	0
$[2^6 - 1] - 56 = 7$	100			
$[2^3 - 1] - 7 = 0$	1			
Remainder = 0	00			

Example 5.4

Change to decimal, $b_0 = 00000011 \rightarrow 3$

TABLE 4: Compression process for $n = 8$-bits data sample

b_i	w_i	Codeword before shift	Codeword after shift	CR
		00000011	00000011	0
Remainder = 3	11			

Example 5.5

Change to decimal, $b_0 = 1011110100011110 \rightarrow 48414$

TABLE 5: Compression process for $n = 16$-bits data sample

b_i	w_i	Codeword before shift	Codeword after shift	CR
$[2^{16} - 1] - 48414 = 17121$	1	1110001110010001	11100011100101	0.125
$[2^{15} - 1] - 17121 = 15646$	1			
$[2^{14} - 1] - 15646 = 737$	1000			
$[2^{10} - 1] - 737 = 286$	1			
$[2^{9} - 1] - 286 = 225$	1			
$[2^{8} - 1] - 225 = 30$	100			
$[2^{5} - 1] - 30 = 1$	100			
Remainder = 1	01			

Example 5.6

Change to decimal, $b_0 = 10111101000111101100011100000011 \rightarrow 3172910851$

TABLE 6: Compression process for $n = 32$-bits data sample

b_i	w_i	Codeword before shift	Codeword after shift	CR
$[2^{32} - 1] - 3172910851 = 1122056444$	1	1110001110010001 1010010010000011	11100011100100 0110100100111	0.156
$[2^{31} - 1] - 1122056444 = 1025427203$	1			
$[2^{30} - 1] - 1025427203 = 48314620$	1000			
$[2^{26} - 1] - 48314620 = 18794243$	1			
$[2^{25} - 1] - 18794243 = 14760188$	1			
$[2^{24} - 1] - 14760188 = 2017027$	100			
$[2^{21} - 1] - 2017027 = 80124$	1000			

TABLE 6 (continued): Compression process for $n = 32$-bits data sample

b_i	w_i	Codeword before shift	Codeword after shift	CR
$[2^{17} - 1] - 80124 = 50947$	1	1110001110010001 1010010010000011	11100011100100 0110100100111	0.156
$[2^{16} - 1] - 50947 = 14588$	10			
$[2^{14} - 1] - 14588 = 1795$	100			
$[2^{14} - 1] - 1795 = 252$	100			
$[2^{8} - 1] - 252 = 3$	100000			
Remainder = 3	11			

6. ADVANTAGES DURING TRANSMISSION

In this section we tabulate according to the following items:

1. Speed of transmission without partitioning the data
2. Speed of transmission after partitioning data
3. Number of data transmission needed (i.e. effort) by machine to transmit (i.e. direct relation with number of partitions on the data to be compressed)

We can see that the compression ratios of all data sizes data are different. The compression ratio of data in table 1 is 0.375 and data in table 5 is 0.125. This is a result of conducting the compression algorithm upon the data without any partitioning of the data. To search for better effiency in deploying the compression algorithm, we partition the 16-bits data in table 5 to 2 new blocks (it has to be noted that for easy reference the data in Table 5 is actually a concatenation of data from Table 1 and Table 2) and we can have compression ratio 0.267. The new compression ratio is better than before. However, if we take data from table 6 which has 0.156 compression ratio and split the data into 4 8-bit blocks, the compression ratio is still same because of block 3 and block 4 does not have any compression and is shown in Table 3 and Table 4. However, maybe there will be a "give and take" situation. That is, to achieve better speed by partitioning the data (in order achieve better compression), the effort by the device to relay the data will increase.

The following is a flow chart of the compression process. As an example, let the data be partitioned into $n = 8$ bits. If the remainder is not a multiple of 8, just pad it with zeroes. For $i = 1,2, \ldots, k$ do

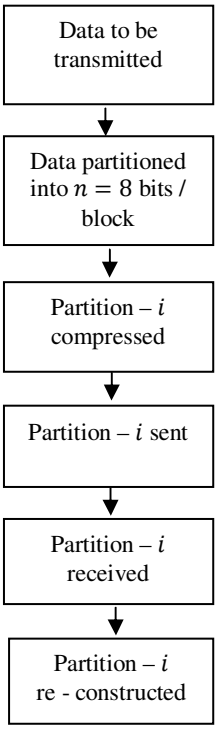

Figure 1: Compression flow chart

Data is transmitted by blocks, and will be reconstructed by block received. It is easy to observe that the flow of our suggested compression method is in line with current data transmission process. To implement our compression mechanism, one only has to determine the corresponding block size according to their needs.

Next, we consider data transmission with the ability to transfer 1 bit data per second. We also assume that the transmission process preceding the compression algorithm will relay $n = 8$ bits per transmission. We can have the following table.

TABLE 7: Comparative table

Data before compression	**Table 1**	**Table 5**	**Table 6**
CR (without partitioning)	0.375	0.125	0.156
Number of partitions	NO	2 blocks of 8 bits	4 blocks of 8 bits
CR (with Partitioning)	-	0.267	0.156
Speed without compression	8	16	32
Speed (compression without partitioning)	5	14	19
Speed (compression with partitioning)	-	11	19
Efforts Speed (compression with partitioning)	1	2	4

Hence, from the above table, it is clear that in some cases partitioning the data will actually result in better speed (at the expense of increased effort by the device).

7. CONCLUSION

This compression scheme gives us many options. Some data will have better compression without partitioning the original data, but there exists cases where partitioning data gives better compression. However better compression means more effort must be put on the transmitting device – even though it is transmitted faster. To this end, this new compression algorithm is certainly advantageous since it is proven analytically to decompress into the exact data. This certainly differs from existing lossless compression methods that are probabilistic in nature (i.e. being able to decompress into exact data by probabilistic mechanisms). Through this work, the scheme can easily be visualize on current transmission technology and would be efficient for live data streaming.

REFERENCES

Ahmadi, O. and Menezes. 2005. Irreducible Polynomials of Maximum Weight. *CACR Technical Reports.*

Burrows, M. and Wheeler, D. J. 1994. A Block-sorting Lossless Data Compression Algorithm. *SRC Research Report.* **124**: 1-18.

Ferreira, A. J., Oliveira, A.L. and Figueiredo, M. A. T. 2009. On the Suitability of Suffix Arrays for Lempel-Ziv Data Compression, *DCC 2009*: 444.

Hellebrand, S. and Wurtenberger, A. 2002. Alternating Run-Length Coding- A Technique for Improved Test Data Compression. *Handout* 3rd *IEEE International Workshop on Test Resource Partitioning, Baltimore, MD, USA.*

Langdon, G. G. 1979. Arithmetic Coding. *IBM J. Res. Develop*. **23**: 149-162.

Pathak, S., Singh, S., Jain, M. and Sharma, A. 2011. Data Compression Scheme of Dynamic Huffman Code for Different Languages. *2011 International Conference on Information and Network Technology.* **4***: 201-205*.

Roman, S. 1997. *Introduction to Coding and Information Theory*. Springer: 51-63.

Ward, M. D. 2005. Exploring Data Compression via Binary Trees, 143-150.