A Unique Approach in Data Compression: Frequency Triggered Efficient Probabilistic Formula Applied with Parallel Binary Representation

Adib Kabir Chowdhury and Veeramani Shanmugan

Department of Electrical and Computer Engineering, Curtin University, Sarawak Malaysia, CDT 250, 98009 Miri, Sarawak, Malaysia

Abstract. This Paper essentially reflects on representing a unique way of compressing data, using the concept of Parallel Binary Representation. Parallel Binary Representation is a unique concept, which focuses on representing binary sequences in parallel, based on the higher frequency of data which is to be compressed. Data Compression in this methodology works as allocating data of higher frequencies at the beginning of the Parallel Binary Representation following the data of lower frequencies. This Paper analyzes famous Huffman Coding which is widely being used for Data Compression. Also, this paper emphasizes and focuses on the result which retrieves data encoding for both Huffman Coding and Parallel Binary Representation. Mathematically it has been proven in this paper that, the unique concept of Parallel Binary Representation achieves higher compression rate than Huffman Coding.

Keywords: Probability, Parallel Binary Representation, Huffman Coding, Frequency, Data Compression, Binary Tree.
PACS: 89.70.Cf

INTRODUCTION

Data compression has become one of the crucial aspects in the computing industry. The need to keep a lot of data is important to many people. Hence there come the peripherals required to keep the data such as hard disk drives. Unfortunately these peripherals alone are not enough to keep data effectively without increasing the cost of data keeping. This brings us to data compression which is an essential part in order to make data keeping cost-effective and friendly to the environment. Data compression works when redundant data are removed to make the information shorter without losing its original function and meaning [1]. The compression is important as the size of the data can be made smaller also shorten the time to transmit the data. In this paper, we will discuss a new method of data compression using parallel binary representation.

LITERATURE REVIEW

Gilschrist proposed data compression using parallel computing as it allows the software to run faster. By using the parallel computing, multiple processors can be used at the same time. So, compressing data in shorter time is achievable. This will benefit data compression especially the large ones. Gilschrist also discussed various ways of implementing parallel version of Burrows-Wheeler Transform (BWT). The parallel data compression process explained in this paper uses bzip2 compression program which provides good compression plus it is available free under open source [2]. Klinc et al. discussed the compression of data encrypted with block ciphers, for example the Advanced Encryption Standard (AES). It is described that such data can be compressed without knowledge of the secret key. Block ciphers operating in various chaining modes are considered and it is shown how compression can be achieved without putting the security of encryption scheme into risk. Also, it is shown that there exists a fundamental limitation to the practical compressibility of block ciphers when no chaining is used between blocks [3]. The proposed methodology in this paper is different than previous researches that have been done in data compression.

FREQUENY TRIGGERED EFFICIENT PROBABILISTIC FORMULA

Every single action that is intended to take place may have a probability of whether it may or may not take place. Every existing physical product in this world is not absolute and therefore always has a probability. Let us assume there exists a text book and all the pages of the text book are filled in with symbols

(Alphabets or Characters). If we consider only one symbol from the text book, we are left out with only two things. One is the symbol we are considering and two, all the other symbols that we are not considering. Now, let us assume, X is the set of all the symbols in a text book. Let, $\{x_1, x_2, x_3 \ldots x_n\}$, be all the symbols in the text book. So, we can say that,

$$X = \{x_1, x_2, x_3 \ldots x_n\} \quad and \quad Y \in X \tag{1}$$

Y is the symbol that we are considering and represents one symbol from set *X*. While considering a symbol in a text book, to find its probability of future appearance, its existing occurrence must be taken under consideration. The number of existing occurrence of the symbol *Y* can be denoted by the notation f and now if we consider set *X*, and go through all its symbol members to find the total number of existing symbols (all Alphabets including repeating and non repeating) in the text book can be represented by $\sum f$ [4-6]. So the probability of *Y* appearing and not appearing can be shown by the equations below:

$$P(Y_a) = f \square \sum f \tag{2}$$

$$P(Y_{na}) = \frac{\sum f - f}{\sum f} \tag{3}$$

Now, as we know that, if *Y* does not appear in the text book, there is possibility that it might have appeared or might not have appeared. And there lies the probability of {1/*n*}, meaning probability of 0.5 for each outcome [4]. So, if we consider the possibility variable N to be 2 we can consider the probability of the Y symbol and can be represented as below:

$$P(Y) = \sum f - \left(f \square N \square \sum f \right) \tag{4}$$

PARALLEL BINARY REPRESENTATION

Parallel Binary Representation uses unique binary sequence represents it in parallel forming a sequential tree. The sequential tree represents the entire structure in two parts. In this paper the Parallel Binary Representation will be named as PBR in some of the sections. The main two division of PBR depends on the Odd and Even bits that forms the combination in the sequential tree. The Left side of the PBR is considered as the Odd section and right side of the PBR is considered as the Even section.

The Odd section forms combination in Odd bits as, $\{1,3,5,7,9,11 \ldots N\}$. And the Even section forms combination in Even bits as, $\{2,4,6,8,10,12 \ldots N+1\}$. Let O_b the set of all odd bits, So, $O_b = \{1,3,5,7,9,11 \ldots N\}$. And, Let E_b the set of all even bits, So, $E_b = \{2,4,6,8,10,12 \ldots N+1\}$. Every row of the PBR consists of Odd bit combination and Even bit combination. Let A_b the set of all odd and even bits in the entire sequential tree of PBR. So, we can say that, $A_b = \{O_b \cup E_b\}$ [4-5]. And all the bits that form the combinations are distributed in a sequential manner, such that every row has maximum of two sorted bits. Any bit combination in a Row in the PBR other than the 1st Row must follow on the previous bit that it extends from. Only exception would be the 1st Row 1 bit combination "0 bit" from which no other combination derives. For example a 2nd Row PBR can be represented by the Figure below:

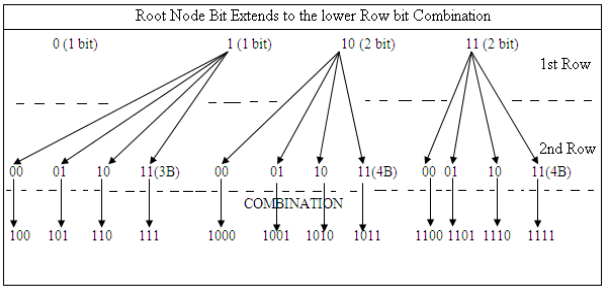

FIGURE 1. Representation of a 2 row PBR odd, even bit combination.

In the Figure 1 above, 1st Row is the basis of the entire sequential tree of the PBR. The first Row consists of 1 odd and 1 even bit combination. Odd bit Combination "1 bit" = 0 and 1, Even bit Combination "2 bit" by default has been set to 2 and the combination is 10 and 11. As 1 bit already represents combination 0 and 1, the necessity for adding extra "0 bit" in 2 bit combination has been eliminated and only 10 and 11 as 2 bit combination is being represented, whereby all the binary combinations remain unique.

In every Row in the PBR, other than the 1st Row, the same sequence of "00 01 10 11" will be post fixed with the previous parent Bit to form unique binary sequences. As we can see that, in PBR the same unique binary representation can be achieved, involving fewer Rows from the Root or the Source. Now it can be observed that how much branch formation is involved for each row of binary tree and Parallel binary Representation. The number of branch

formation for each row of a PBR can be denoted by PBR_b, below shown is the mathematical representation:

$$PBR_b = 2^{2n} \quad (5)$$

Graphical Illustration of Parallel Binary Representation

The Graphical Illustration assists in visualizing the Parallel Binary Representation. The entire structure of the PBR can be observed and understanding the expansion of PBR can be achieved through the graphical representation as shown below. A 2nd Row (Involving up to 2nd Row) Graphical representation of PBR is shown below in the Figure 2. The Root node is the source node, from where the PBR forms and later it extends to n^{th} row. However, the first row nodes "*S1*" represents combination of odd and even bit for the first row. Similarly the second row nodes "S2" represents each combination of odd and even bit for the second row. The binary sequence at the bottom of each row nodes (*S1*,*S2*...) represents the combination that belongs to the odd/even bit.

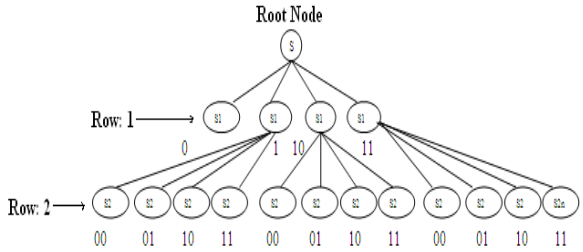

FIGURE 2. A 2nd Row Parallel Binary Representation

Distribution of a Unique Binary Sequence for Parallel Binary Representation

The distribution of a unique binary sequence for PBR depends upon the odd and even bits, which are distributed in the sequential tree in a sorted order, from lowest to highest. For example, the first row contains only odd 1 bit combination and even 2 bit combination. The following Table 1 illustrates the scenario:

TABLE 1. Representation of unique binary sequence in each row of PBR

Row	Odd Bit	Odd Bit Combination	Even bit	Even Bit Combination
1	1	0 1	2	10 11
2	3	100 101 110 111	4	1000 1001....
3	5	10000 10001	6	100000
.	.	10010.....	.	100001...
r^{th} Row	n^{th} bit	. 2^{2r}	$(n+1)^{th}$bit	. 2.2^{2r}

So it can be realized that for each Row, the number of combinations for a PBR is half of the number of the combination for a usual binary combination. And if the Parallel combination can be expressed by *Pc*, then for any bit (meaning Odd or Even bit: All bit) the combination can be represented as follows:

$$P_c = 2^n \left(1 - \frac{1}{2}\right) \quad (6)$$

The Graphical Representation of how the entire PBR internally works and linked to each other can be represented by Figure 3 below. The diagram below represents an entire PBR up to N^{th} Row, and uses arrow to its child bit combination. The left side of the diagram from the source represents Odd combination and the right side of the diagram from the source represents even combination. The very first row has two parent odd and even bit combinations. For odd 1 bit combination, the parent combinations are 0 and 1, and so forth. The child combination that extends from this parent combination, the parent bit must be taken under consideration to get the unique binary combination.

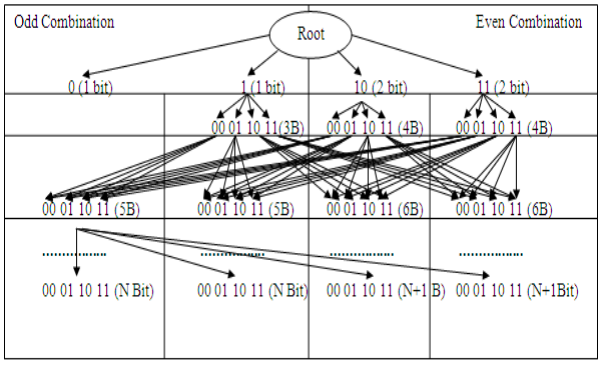

FIGURE 3. Internal representation of a PBR including the odd and even combination scenario.

Parallel Binary Representation Mathematical Background

The total combination for each row of PBR depends explicitly on the current row of the PBR. Total combination for each Row of the PBR can be denoted by "T_c". As it has been shown previously that a PBR start from a Root node. The first row of a PBR is exceptional and solely based on four combinations: 0, 1, 10 and 11. This combination of the 1st row is denoted by "R_{ic}" and always added to find total combination of a PBR. Thus R_{ic} = 4 is a constant. T_c has been defined from the 2nd row. If current row is denoted by R_c, it can be stated that, in order to evaluate T_c, current row $R_c \geq 2$. The entire formulation cannot be defined for $r < 2$, as for the 1st Row R_{ic} has been initialized. So by default the entire 1st Row (R_{ic}) can be set to total combination 4(Odd "2" and even "2"). Now the derivation of the Row Summation is represented as below:

Let us assume, T_c is the Total Combination for a Row, as we know, Odd Combination C_{odd}: $2^{2(r-1)}$, Even Combination C_{even}: $2.2^{2(r-1)}$. Therefore, for any $R_c \geq 2$, total combination for a row can be represented by:

$$T_c = 2^{2(r-1)} + 2 \square 2^{2(r-1)} \tag{7}$$

Considering 3 as the only constant alpha "α" in the equation,

$$T_c = \alpha \square 2^{2(r-1)} \tag{8}$$

Now it is known that, the initial combination of the PBR for the first Row is 4(Odd "2" and even "2"). Therefore we can denote R_{ic} as the initial combination. Let "R_{tc}" be the total combination of the entire binary representation. And also if the number of row "r" starts from 1 and ends at any i^{th} row we can represent the entire summation of combinations by the equation:

$$R_{tc} = R_{ic} + \sum_{r=2}^{i} Tc \tag{9}$$

Or,
$$R_{tc} = R_{ic} + \alpha \square \sum_{r=2}^{i} 2^{2(r-1)} \tag{10}$$

If we consider B_e to be the number bits which is to be used for encoding a symbol and C_r to be the range of combination for that sequence of bits, then for any Combination (Either be odd or even) the specific amount of bits involved in the combination in a PBR, can be represented by the equation below:

$$B_e = \log_2 (Cr) + 1 \tag{11}$$

Where C_r can be either Odd combination of a row, C_{odd} or Even Combination of a row, C_{even}. And B_e is defined as the number of bits involved in the combination. C_r is defined as the range for the combination.

RESULTS AND COMPARISON

A sentence had been considered for the initial experiment. "themangoissweet" is the test bench for the initial testing and comparison between Huffman and PBR. In order to get the frequency of each of the alphabets, every alphabet needs to be represented in terms of their occurrence in the sentence. Frequencies and their related probability of the alphabets are shown in Table 2 as below:

TABLE 2. The use of frequency triggered Efficient Probabilistic Formula in determining probability

Alphabets: Elements (E)	Frequency (f)	$P(e) = \frac{\sum f - f}{N.\sum f}$
T	2	0.43
H	1	0.47
E	3	0.40
M	1	0.47
A	1	0.47
N	1	0.47
G	1	0.47
O	1	0.47
I	1	0.47
S	2	0.43
W	1	0.47
	$\sum f = 15$	

If total bits encoded denoted by T_{BE}, Frequency of the symbol denoted by F_r and number of encoded bits by N_{EB}, then the equation can be represented in the form as below:

$$T_{BE} = F_r \square N_{EB} \tag{12}$$

Parallel Binary Representation Approach

In order to get the best outcome of the encoding in Parallel Binary Representation Approach, the probability of the symbols would be sorted in order of lowest to highest. In order to assign bits for encoding, the lowest probability symbol would be assigned the first bit in PBR which is: 0, and the second would be assigned 1 and so forth. According to the Table 2

above it can be understood that, lower the probability, higher is the frequency of the symbol.

In case, multiple symbols happen to have the same probability, then the assignment would be based on the appearance of the symbols in the test bench (In our case, it is only a sentence, can be a text book etc.), which have the same probability. And then the assignment would go on again based on the probability. For example, the above sentence as a test bench, and all the probability calculations from the Table 2 above, the Parallel Binary Representation can be shown as below in Figure 4:

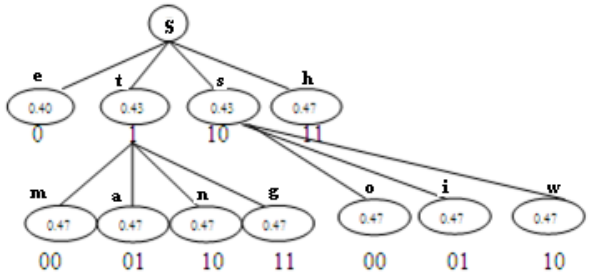

FIGURE 4. Internal representation of a PBR including the odd and even combination scenario.

Encoded message is: **1 11 0 100 101 110 111 1000 1001 10 10 1010 0 0 1**. It is being observed from the binary string above that the information (message) is being encoded into binary 1's and 0's according to PBR. The length of the encoded string is 35 bits long.

Huffman Coding Approach

The same sentence "themangoissweet" is encoded using the Huffman Coding. The encoded message is: **010 0110 00 0111 1101 1110 1111 1010 1011 100 100 1100 00 00 010**[7-8]. It is being observed from the binary string above that the information (message) is being encoded into binary 1's and 0's according to Huffman Coding. The length of the encoded string is 50 bits long.

DISCUSSION

According to David Salomon, in his book "Data Compression: The Complete Reference" he mentioned about compression factor. The inverse of the compression ratio is called the compression factor. The compression factor can be represented by the equation below [9]:

$$Compression\quad Factor = \frac{Input\quad Stream\quad Size}{Output\quad Stream\quad Size} \quad (13)$$

In this case, values which are greater than 1 indicate compression and values which are less than 1 indicates expansion. This measurement is distantly related to the growing ratio or the sparseness ratio, a performance measurement. As a test-bench the string ("themangoissweet") which was considered for the compression purpose, consists of 15 symbols (characters). If each of the symbols in the memory occupies 8 bits (1 byte), for 15 symbols it occupies 120 bits or 15 bytes. According to the proposed methodology, PBR Compression gives 35 bits for the test-bench. After compression which is 35 bits or 4.375 bytes. For the Huffman Coding after compression can be achieved up to 50 bits or 6.25 bytes. Table 3 demonstrates the comparison of both Huffman Coding and PBR methodology:

TABLE 2. Compression comparison of Huffman Coding and PBR

	Original Text	Huffman Coding	PBR Methodology
Size of Input Stream	15 bytes	15 bytes	15 bytes
Size of Output Stream	15 bytes	6.25 bytes	4.375 bytes
Compression Factor	1	2.40 bytes	3.43 bytes

From the Table 3 above, it is observed that the Compression Factor for PBR Methodology (3.43) is higher than the Compression Factor for Huffman Coding (2.40), which indicates better data compression using PBR Methodology.

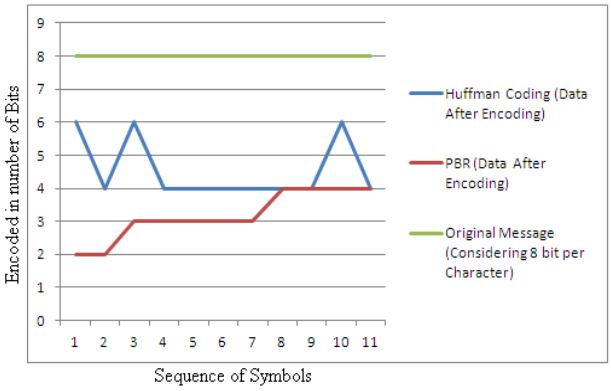

FIGURE 5. Result comparison of Parallel Binary Representation and Huffman Coding

Figure 5 above illustrates the efficiency of PBR Compression methodology over Huffman Coding. The Sequences on the x-axis indicates each symbol that the test bench consists. Duplicated symbols are considered

only once considering the redundancy of the appearance of the symbols. Y-axis consists of bits, which is being used per symbol. A generalization of uncompressed data flow is shown in the figure 5, which shows its constant memory usage in 8 bits per symbol. Lower usage of memory using PBR methodology can be observed as its compression is higher than Huffman Coding.

CONCLUSION

In the conclusion it can be stated that Parallel Binary Representation allows the data to be compressed in a higher rate comparing to Huffman Coding. In terms of memory overhead PBR is costly as it would require more memory space to save the floating points and check each of the probabilities in the PBR tree. In fact, even though PBR is costly in terms of memory overhead, the algorithm provides stronger compression rate compared to Huffman Coding Algorithm.

REFERENCES

1. S.A. Campos, *The Introduction to Data Compression*, http://www.arturocampos.com/cp_ch1.html (2000)
2. J. Gilchrist, *Parallel Data Compression with BZIP2*, Elytra Enterprise Inc., Ottawa.
3. Klinc, Demijan, C. Harvay and T. Rabin, *On Compression of Data Encrypted with Block Ciphers*, 2009.
4. S. Lipschultz and M. Lipson, *Schaum's Outline of Theory and Problems of Probability*, 2nd ed, New York, McGraw-Hill, 2000.
5. A.J Hayter, *Probability and Statistics for Engineers and Scientists*, 2nd ed, Great Britain, Pacific Grove.
6. M. Lefebvre, *Applied Probability and Statistics*, New York, Springer, 2006.
7. R. Howell, *Huffman Trees*, http://people.cis.ksu.edu/~rhowell/viewer/huffman.html (2010)
8. (n.d.). *Informatik: Lossless Compression* http://pi4.informatik.unimannheim.de/pi4.data/content/animations/losslesscompression/index.html
9. D. Salomon, *Data Compression: The Complete Reference*, Dordrecht, Springer, 2007.