Burrow Wheeler's Transform for Efficient RLE Compression Applied to Audio Files

[1]Harsh Shah [2]Chinmay Sane [3]Swapnil Sheth [4]K.T.Talele

Abstract--**The aim of the project is to implement Burrow-Wheeler's Transform, a completely new approach to data compression which is the basis of some of the best compression techniques available today and show its reversibility. BWT transforms a block of data into a format that is extremely well suited for compression. The BWT algorithm takes a block of data and rearranges it using a sorting algorithm. The resulting output block contains exactly the same data elements that it started with, differing only in their ordering. The transformation is reversible; meaning the original ordering of data elements can be restored with no loss of fidelity. We have tested the proposed algorithm for different audio signals. The results show that use of BWT prior to compression has a significant impact on the size of compressed data.**

Index Terms-- Data Compression, Lossless Transform Algorithm, Redundancy, Run Length Codes

I. Introduction

COMPRESSION is an important and interesting computer application. Compression algorithms reduce the redundancy in data representation to decrease the storage required for that data. Data compression offers an attractive approach to reducing communication costs by using available bandwidth effectively. If we could somehow sort data before compressing it, we could achieve very high compression ratios indeed. For example, textual data, sorted alphabetically, character-by-character would become trivially easy to compress. Of course, the reason we do not sort data like this is that such a sort is not reversible. In other words we could never decompress our data.

However in 1994, Michael Burrows and David Wheeler, working at the DEC Research Center in Palo Alto, published a paper showing a way to reversibly sort textual data that increased the compressibility of the data [1]. The proposed reversible sort is known as the Burrows-Wheeler Transform (BWT). A block of data transformed by the BWT can be compressed using standard techniques. In this project, we investigated whether the idea of sorting data before compression is also useful for audio compression. Combined with other algorithms like - Move-to-Front encoding (MTF) - and Huffman or Run-length-encoding (RLE), the BWT provides better compression ratios than those given by these techniques alone [2]. BWT gives lossless data compression. The algorithm has received considerable attention because of its execution speed and its compression performance.

II. Technical Work Preparation

A. The Basics of BWT

BWT is a 'lossless transform algorithm' that takes a block of data and rearranges it using a sorting algorithm. The resulting output block contains exactly the same data elements that it started with, differing only in their ordering. It works as explained below-

B. Transform:

The BWT is applied on a 1-dimensional set of data. The output is generated according to the following scheme:

- All different rotations of the original data sequence are formed and arranged in a square matrix in ascending order whereby the elements of the first column are the most significant.
- Store the number of the row which contains the original sequence as the index I in order to allow the inverse transform to be successfully applied to the transformed sequence.
- Take the last column as the output of the transform.

Example:

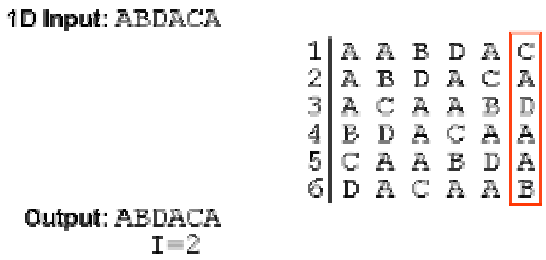

Fig. 1. Sorted text example

[1]Chinmay Sane is a student of third year of Electronics Engg., S.P.College of Engg.,Mumbai India .(email:chinmaysane@yahoo.co.in)
[2]Harsh Shah is a student of third year of Electronics Engg., S.P.College of Engg.,Mumbai India (email:hardison87@yahoo.co.in)
[3]Swapnil Sheth is a student of third year of Electronics Engg., S.P.College of Engg.,Mumbai India .(email: swap_neal4u@yahoo.co.in)
[4]K.T.Talele is a faculty in Electronics Dept., of S.P.College of Engg.,Mumbai India .(email: kttalele@yahoo.co.uk)

C. Inverse Transform:

Given the transformed vector L (i.e. the last column in the matrix of Figure 1), and the index I, we can restore the original sequence S.

We observe the following

- The output vector of the BWT contains exactly the same number of occurrences of each symbol as the original sequence S. Therefore, sorting L yields the first column of the transformation pattern as illustrated in Figure 1.
- Furthermore, all sequences starting with the same symbol are sorted according to their second symbol.

As the first column contains the symbols that precede those in the last column (imagine the pattern to be spread over the surface of a cylinder rather than a plane) and as the first column is sorted anyway those rows whose last elements are equal are (among themselves) sorted with respect to the first column. In other words, those rows which have, for instance, symbol A as their last element in common are sorted in the same order as the rows which have A as their first element in common.

We create a one-to-one mapping table that maps the row number with the k^{th} occurrence of symbol A as the last element to the row number with the k^{th} occurrence of A in the first column.

The last element of S is simply the element of L at the position of the index.

Since the elements of the first column precede those in the last column as described above, we can now recursively reconstruct S from back to front obeying the following algorithm:

- Locate the current symbol in the last column and pick the element e of the first column but in the same row.
- Use mapping table to find the symbol in the last column that corresponds to the element e of the first column – the two symbols are equal due to the definition of the mapping table.
- e is now the current symbol.

When a character string is transformed by the BWT, none of its characters change value. The transformation permutes the order of the characters. If the original string had several substrings that occurred often, then the transformed string will have several places where a single character is repeated multiple times in a row. This is useful for compression, since it tends to be easy to compress a string that has runs of repeated characters by techniques such as move-to-front transformation and run length encoding [2].

D. Use of BWT for a wave file:

In this project we have used BWT on a audio file. For this, we need 2 convert the audio (wave) file samples into a string so that we can apply the transform. The original wave file samples can be observed in the figure below.

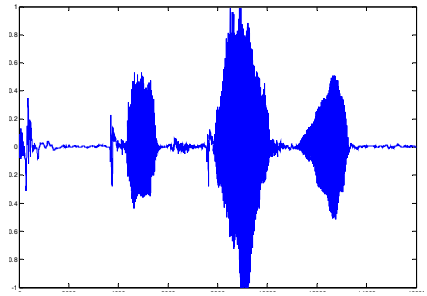

Fig. 2. Original Wave File

E. Encrypting with BWT:

- These sample values are stored in an array of 1 column and number of rows same as the number of samples.(Here the number is 16,000).
- The sample values are in the range -1 to 1. To make them fit for BWT, all the values are made positive by 'uint'function in MATLAB and normalized by a multiplying factor.
- Now we consider this modified array as a string and apply BWT on it. The output of the transform contains the same number of samples as that of the original signal; but these samples are now encrypted. The encrypted output samples if plotted appears as follows

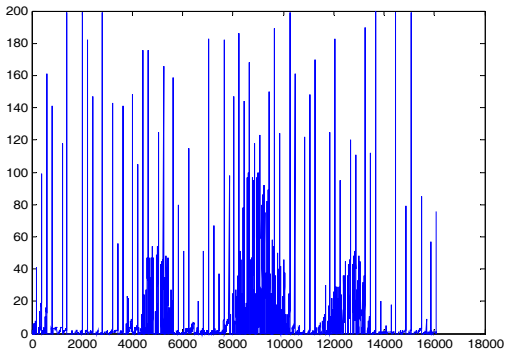

Fig. 3. Effect of BWT on the wave file

F. Decrypting with BWT:

The encrypted file is the array which contains the transformed samples of the original file. The decryption is done by applying inverse BWT to this array. The output so obtained is same as the original wave file. This can be seen in the figure below.

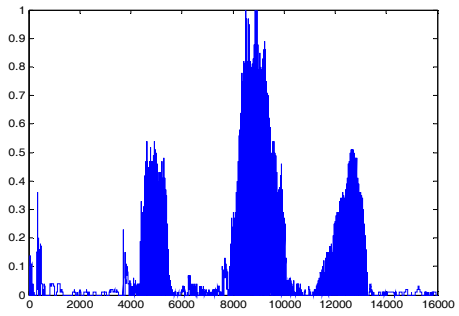

Fig. 4. Reconstructed wave file

G. RLE (Run Length algorithm) applied on the transformed array:

- To compress the daa by reducing the number of samples, we can apply RLE to the transformed array.
- Compression with RLE works very well for a BW transformed signal because the output of transform contains same characters appearing consecutively [3].
- As we can see in the figure the number of samples is reduced from 16000 to 10684,

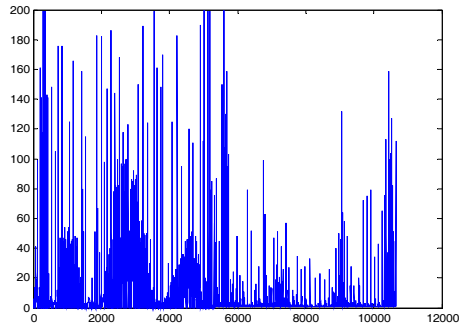

Fig. 5. Effect of RLE on the transformed signal

III. CONCLUSION

In this project we successfully implemented the lossless Burrow-Wheeler's Transform and verified its reversibility. We demonstrated the use of BWT for audio signals. Subsequent use of compression techniques on the transformed audio file gives better compression than use of these techniques directly on the audio file.

IV. REFERENCES

Books:

[1] Burrows, M. and Wheeler, D.J. (1994) "A Block-sorting Lossless Data Compression Algorithm", Digital Systems Research Center Research Report 124.

[2] Nelson, Mark and Gailly, Jean-Loup, (1995) The Data Compression Book, second edition, M&TBooks, New York.

[3] H. Kruse and A. Mukherjee, "Improving Text Compression Ratios with the Burrows-Wheeler Transform," Proceedings of the IEEE Data Compression Conference 1999, Snowbird, p. 536.

[4] B. Chapin and S. Tate, "Higher Compression from the Burrows-Wheeler Transform by Modified Sorting," Proceedings of the IEEE Data Compression Conference 1998, Snowbird, p. 532.

V. BIOGRAPHIES

Harsh Shah is a student of final year of Electronics Engg., S.P.College of Engg., Mumbai, India. His fields of interest include robotics, DSP and programming.
(hardison87@yahoo.co.in)

Chinmay Sane is a student of final year of Electronics Engg., S.P.College of Engg., Mumbai, India. His fields of interest include microprocessors and computer organization.
(chinmaysane@yahoo.co.in)

Swapnil Sheth is a student of final year of Electronics Engg., S.P.College of Engg., Mumbai, India. His fields of interest include microprocessors and telecommunications.
(swap_neal4u@yahoo.co.in)

K.T.Talele is Assistant Professor in Electronics Engg Dept, S. P. College of Engineering, Mumbai. He is a member of IEEE. His area of interest includes DSP, Image Processing and Multimedia Comm. He has published twenty papers in National Conferences and four papers in International conference.
(kttalele@yahoo.co.uk)