ISSN: 0976-3104
SPECIAL ISSUE: Computer Science

ARTICLE

A SIMPLE LOSSLESS COMPRESSION ALGORITHM IN WIRELESS SENSOR NETWORKS: AN APPLICATION OF SEISMIC DATA

J. Uthayakumar[1*], T. Vengattaraman[1], J. Amudhavel[2]

[1] *Department of Computer Science, Pondicherry University, Puducherry, INDIA*
[2] *Department of Computer Science and Engineering, KL University, Andhra Pradesh, INDIA*

ABSTRACT

Background: Seismic hazard is one of the natural hazards which are very difficult to predict and detect. Various seismic and seismo-acoustic monitoring systems are developed to observe the changes in rock mass processes. Some of the recent seismic dataset exceed 10Tbytes and there is a fact that present seismic studies planned with an amount of 120Tbytes. The size of the seismic data is a challenge to store, process or transmit seismic data for further investigation. ***Methods:*** Data compression (DC) techniques are commonly used to reduce the amount of data being stored or transmitted. As seismic monitoring system is a critical application, the loss of data quality is not tolerable. So, lossless data compression techniques are highly preferable. In this paper, Burrows Wheeler Transform (BWT) is used to compress the seismic data effectively. BWT is a block sorting compression which rearranges the character string into runs of identical characters. ***Results:*** *Extensive experiments were performed using real world seismic bump dataset. To ensure the effectiveness of lossless BWT coding, it is compared with 4 well-known compression algorithms namely Lempel Ziv Markov Algorithm (LZMA), Huffman coding, Lempel-Ziv-Welch (LZW) coding and Arithmetic coding (AC).* ***Conclusions:*** *By comparing the compression performance of BWT coding with existing methods, BWT achieves significantly better compression with a compression ratio of 0.167 at the bit rate of 1.342 bpc.*

KEY WORDS
Data Compression
BWT coding
Seismic data
Wireless Sensor Network

Received: 3 June 2017
Accepted: 28 July 2017
Published: 18 Sept 2017

***Corresponding Author**
Email:
uthayresearchscholar@gmail.com
Tel.: +91 9677583754

INTRODUCTION

The unexpected emission of energy from the earth crust results to earthquake. Because of the energy, seismic waves arise and the waves generate a distress effect on earth. There are various issues that influence earthquake. These factors enhance the complexity and it leads to identify location, intensity and time of the earthquake. Several studies use several information to predict earthquake like earth's crust form change, slope change, radon gas changes in well and springs, elastic variable wave velocities, groundwater level variation and seismic pulses. Few researches used seismic bumps to carry out this process. [1] developed a system to predict earthquake by investigating seismic bump data. They used k nearest neighbor algorithm from classification and obtained an accuracy of 94.11%. [2] introduced an intelligent system to predict earthquake by investigating seismic bump data with a classification accuracy of 91%. [3] performed an experiment using neuro-fuzzy system algorithm in seismometer data and they achieved an accuracy of 82%. Generally, the seismic sensors are deployed in the surface which measures the seismic signals and the signals are processed to create the picture of the subsurface. Some advanced marine seismic dataset are larger in size and it exceeds 10 Tbytes. In some cases, the volume of seismic data may exceed 120Tbytes. Since, there is a need to compress this massive amount of seismic data. Data transmission is the most energy consuming task due to the nature of strong temporal correlation in the sensed data. DC is considered as a useful approach to eliminate the redundancy in the sensed data. In WSN, the sensor node runs the compression algorithm and the compressed data will be forwarded to Base Station. The easier way to handle huge amount of data is to compress them [3]. It is applicable to compress text, images, audio or video. The data can be alphanumeric characters in a text document or numbers which represent the samples in an audio or image waveforms or series of numbers created by some processes, etc. DC can also be termed as an efficient way of representing the data in its compact manner. The compact form can be achieved by the recognition and utilization of patterns exists in the data. The evolution of DC starts with the Morse code, developed by Samuel Morse in the year 1838 [4]. Morse code is used in telegraph to compress letters. It uses smaller sequences to represent letters which occurs more frequently and thereby minimizing the message size and transmission time. This basic principle is employed in Huffman coding [5]. Nowadays, DC techniques are very essential in most of the real time applications like medical imaging, satellite imaging, Wireless Sensor Networks (WSN), etc. DC became important for the utilization of available resources effectively. Without DC, it is very difficult and sometimes impossible to store or communicate huge amount of data files.

Basically, DC techniques falls under two categories: lossless compression and lossy compression. As the name implies, lossless compression refers that no loss of information i.e. the reconstructed data is exactly same as original data. It is used in situations where the loss of information is unacceptable. Example: Text, medical imaging, satellite imaging, etc. Lossless compression is generally used to compress text. As loss of information is intolerable in text compression, compressing text using lossless compression is mandatory. It achieves the compression ratio of 2:1 to 8:1 [6]. In some situations, the reconstructed data need not to be perfectly matched with the original data and the approximation of original data is also acceptable. In those situations, lossy compression is used which involves with some loss of information within the acceptable level. This leads to higher compression ratio when compared to lossless compression. Example: image, audio and video. Lossy compression is commonly used to compress images as the loss of image quality is tolerable. The main principle behind image compression is the correlation of pixels. The neighboring pixels of an image are high correlated. These redundant details are eliminated by computing a less correlated representation of the image. Lossy compression achieves the compression ratio of 100:1

IIOAB JOURNAL
ISSN: 0976-3104

to 200:1 based on the data to be compressed. It achieves higher compression ratio at a cost of loss in quality.

Only few researches have been made on the compression of seismic dataset. As the lossy compression techniques involve loss of quality, it cannot be used to compress seismic data which is intolerant to errors. It leads to the development of lossless compression of seismic data in the real world. Once the seismic data is compressed, it can be useful in several ways. In WSN, the sensor node executes the compression algorithm to compress the seismic data. Then, the compressed data will be transmitted to BS for further investigation. It can be easily shared; it consumes less storage with low management cost. The absence of effective lossless compression algorithm for seismic data motivated us to perform this work.

Contribution of the paper

The contribution of the paper is summarized as follows: (i) A lossless BWT compression algorithm is used to compress seismic bump dataset in WSN (ii) Two seismic dataset from NREL (Eastern and Western Dataset) is used, and (iii) BWT results are compared with 4 well-known compression algorithms namely Huffman coding, AC, LZW and LZMA interms of Compression Ratio (CR), Compression Factor (CF) and bits per character (bpc).

Organization of the paper

The rest of the paper is organized as follows: Section 2 explains the different types of classical DC techniques in detail. Section 3 presents the BWT compression algorithm for seismic dataset in WSN. Section 4 discusses the compression results obtained for the applied dataset. Section 5 concludes with the highlighted contributions, future work, and recommendations.

RELATED WORK

DC is an efficient way to reduce the amount of data being stored or transmitted. The popular coding methods are Huffman coding, Arithmetic coding, Lempel Ziv coding, Burrows-Wheeler transform (BWT), RLE, Scalar and vector quantization.

Huffman coding [7] is the most popular coding technique which effectively compresses data in almost all file formats. It is a type of optimal prefix code which is widely employed in lossless DC. It is based on two observations: (1) In an optimum code, the frequent occurrence of symbols is mapped with shorter code words when compared to symbols that appear less frequently. (2) In an optimum code, the least frequent occurrence of two symbols will have the same length. The basic idea is to allow variable length codes to input characters depending upon the frequency of occurrence. The output is the variable length code table for coding a source symbol. It is uniquely decodable and it consists of two components: Constructing Huffman tree from input sequence and traversing the tree to assign codes to characters. Huffman coding is still popular because of its simpler implementation, faster compression and lack of patent coverage. It is commonly used in text compression.

AC [8] is an another important coding technique to generate variable length codes. It is superior to Huffman coding in various aspects. It is highly useful in situations where the source contains small alphabets with skewed probabilities. When a string is encoded using arithmetic coding, frequent occurring symbols are coded with lesser bits than rarely occurring symbols. It converts the input data into a floating point number in the range of 0 and 1. The algorithm is implemented by separating 0 to 1 into segments and the length of each segment is based on the probability of each symbol. Then the output data is identified in the respective segments based on the symbol. It is not easier to implement when compared to other methods. There are two versions of arithmetic coding namely Adaptive Arithmetic Coding and Binary Arithmetic Coding. A benefit of arithmetic coding than Huffman coding is the capability to segregate the modeling and coding features of the compression approach. It is used in image, audio and video compression.

Dictionary based coding approaches find useful in situations where the original data to be compressed involves repeated patterns. It maintains a dictionary of frequently occurring patterns. When the pattern comes in the input sequence, they are coded with an index to the dictionary. When the pattern is not available in the dictionary, it is coded with any less efficient approaches. The Lempel-Ziv algorithm (LZ) is a dictionary-based coding algorithm commonly used in lossless file compression. This is widely used because of its adaptability to various file formats. It looks for frequently occurring patterns and replaces them by a single symbol. It maintains a dictionary of these patterns and the length of the dictionary is set to a particular value. This method is much effective for larger files and less effective for smaller files. For smaller files, the length of the dictionary will be larger than the original file. The two main versions of LZ were developed by Ziv and Lempel in two individual papers in 1977 and 1978, and they are named as LZ77 [9] and LZ78 [10]. These algorithms vary significantly in means of searching and finding matches. The LZ77 algorithm basically uses sliding window concept and searches for matches in a window within a predetermined distance back from the present position. Gzip, ZIP, and V.42bits use LZ77. The LZ78 algorithm follows a more conservative approach of appending strings to the dictionary.

IIOAB JOURNAL
ISSN: 0976-3104

LZW is an enhanced version of LZ77 and LZ78 which is developed by Terry Welch in 1984 [11]. The encoder constructs an adaptive dictionary to characterize the variable-length strings with no prior knowledge of the input. The decoder also constructs the similar dictionary as encoder based on the received code dynamically. The frequent occurrence of some symbols will be high in text data. The encoder saves these symbols and maps it to one code. Typically, an LZW code is 12-bits length (4096 codes). The starting 256 (0-255) entries represent ASCII codes, to indicate individual character. The remaining 3840 (256-4095) codes are defined by an encoder to indicate variable-length strings. UNIX compress, GIF images, PNG images and others file formats use LZW coding.

A simpler form of lossless DC coding technique is Run Length Encoding (RLE). It represents the sequence of symbols as runs and others are termed as non-runs. The run consists of two parts: data value and count instead of original run. It is effective for data with high redundancy. For example, "ABCABBBBBC" is the input sequence, then the starting 4 letters are assumed as non-run with length 4, and the next 4 letters are assumed as a run with length 5 while B is repeated 5 times. RLE identifies the run of input file, saves the symbol and length of each run. These runs are used to compress the input file and non-runs are kept uncompressed. It is not effective in cases where the redundancy is low and leads to increase in original file size. It is ineffective for less redundant data and leads to increase in compressed file size greater than original file size. Input: YYBCCECCDESSEEEEER and its Output: 2Y1B2C1E2C1D1E2S5E1R. Though RLE is simpler to implement and faster, it fails to achieve better compression when compared to other algorithms. It is used in line drawing, animation, graphic icons, fax and palette-based images like textures.

Quantization is an easiest way to represent larger set of values to a smaller set. The quantizer input can be either scalar or vector. When the input is scalar, then it is called scalar quantizer. When the input is vector, then it is called vector quantizer. Scalar quantization is the simpler and commonly used lossy compression. It is a process of mapping an input value x to a number of output values. Vector quantization (VQ) is a different type of quantization, which is typically implemented by choosing a set of representatives from the input space, and then mapping all other points in the space to the closest representative. Initially, the source output is grouped into blocks or vectors. These vectors are used as the input to quantizer. In both sides of encoder and decoder, a codebook (set of L-dimensional vectors) is available. The vectors in the codebook are termed as code-vectors and each code vector is assigned to an index value. The elements of this code vector are the quantized values of the source output. In order to report the decoder about which code vector was found to be the closest to the input vector, the index value of the code-vector is transmitted. As the decoder also has same codebook, the code vector can be retrieved from its binary index.

Lossless compression techniques are widely used in text compression as loss of information is not tolerable while compressing text. A novel data compression technique is proposed to compress general data using a logical truth [Table 1]. Here, the two bits of the data are represented by only one bit. Using this method, the data bits can be represented by its half bits only. i.e. 128 bits can be represented by 64 bits; 64 bits can be represented by 32 bits; 1GB data can be represented by 512MB and so on. The logic truth table with two inputs and four combinations are given below.

IF A=0 and B=0 then Z=0
IF A=0 and B=1 then Z=Ô
IF A=1 and B=1 then Z=1
IF A=1 and B=0 then Z=î

First, input data is tested to identify whether it is odd or even. If it is odd, extra bit is added at the last (0 is added when the last bit is 0 and vice versa). Then, the data is split into two bits and truth [Table 1] is applied to compress the data. Traditionally, there are two levels to represent digital data but proposed method uses four levels. The proposed method compresses the 20 bit data into 10 bits. This method uses Compression Ratio and Compression Factor as performance metrics. The author fails to compare the proposed method with existing method. It can be employed in wired and wireless scenarios.

Although some of the common compression techniques can be applicable to some preferred applications, the concentration has been on the technique rather than on the application. However, there are certain techniques for which it is impossible to separate the technique from the application. This is because that several techniques depend on the properties or characteristics of the application. Therefore, some compression techniques were developed with focus on particular applications.

A new database compression method is proposed to compress large databases using association rule mining. Association rule mining from a database can identify frequent item sets in the database relations. The extracted association rules represent the repeated item sets in the database and they are compressed to decrease the file size. There are five steps in database compression: i. An effective association rule mining method using a tree called Class Inheritance Tree (CIT) is employed, ii. Collection of compression rules from strong association rules, iii. Compressed space for each compression rule is calculated, iv. A heuristic method is employed to resolve the conflicts of compression rules and v. Compression ratio is computed.

To reduce the Inter Track Interference (ITI) read latency of Shingled Magnetic Recording, two lossless compression mechanisms are used. It is a known fact that most of the files stored in disks are lossless

IIOAB JOURNAL
ISSN: 0976-3104

compressible. When a sector of user data can be losslessly compressed to a particular level, free storage space will become available to store ECC redundancy. This extra space can be used for storing a stronger ECC than normal ECC for the current sector. This leads to decrease the probability of reading one or multiple neighboring tracks for explicit ITI compensation. Lempel-Ziv-Stores-Symanski (LZSS) algorithm is used for compressing several file types stored in hard disks. This method uses intra-sector data compression method to minimize the read latency is shingled recording. Here, lossless data compression technique is employed on multiple physically consecutive sectors on the same track to increase the compression efficiency. It results to reduction in read latency overhead by enabling a stronger ECC. This is termed as virtual sector compression. Because of overlapping writing in shingled recording, update-in-place feature fails to work. This makes the virtual sector compression practically possible. Two scenarios (fixed size and context aware virtual sector) are used to investigate the results of virtual sector compression. Intra-sector lossless compression results to the decrease in disk read latency.

Phasor measurement units (PMU) are used to monitor and regulate the power grid where the devices records globally synchronized measurements of voltage and current signals. It provides the respective phasor magnitude and angles at typical rate of 30 frames/s. To compress the phasor angle data, a new pre-processor is proposed. An adaptive high performance entropy encoder called golomb rice coding. This method involves two stages: In the first stage, preprocessing of data takes place. A new preprocessor based on frequency compensated differential encoding is proposed. In the second stage, golomb rice encoding is used. It is a lossless compression technique where the data transmission from space vessels takes place and higher throughput is achieved with limited memory and computational resources. Golomb encoding is designed to encode non-negative, integer valued signals in which probability value n decreases exponentially. It is very effective than existing coding techniques like Huffman coding and Arithmetic coding. Golomb rice coding is simpler to implement in hardware because it uses only few operators namely bitwise left, bitwise right, bitwise AND, OR, ++ and --. Next, mapping of values to GR codes does not require any lookup tables or trees/ GR compress 8 million phasor angles/second shows that it can be very useful for applications requires less delay. Data rate, compression rate and Error Propagation Rate (EPR) are used to evaluate the performance of this technique.

BWT COMPRESSION ALGORITHM ON SEISMIC DATA

The BWT [12] is a reversible block-sorting transform works on a series of data symbols to attain a permuted data sequence of similar symbols and an integer in {1, 2,..., n}. Let

$$BWT_n: X^n \rightarrow X^n \times \{1,2..,n\}$$

represents the n-dimensional BWT function and inverse of BWT is represented by

$$BWT_n^{(-1)}{}_n: X^n \times \{1,2..,n\} \rightarrow X^n$$

As the length of the series is identified from the source argument, the functionally transcript is removed and it gives

$$(y^n, u) = BWT(x_n)\ and\ BWT^{(-1)}((y^n, u) = x^n$$

where BWT_X represents the character and BWT_N represents the integer part of BWT. The forward BWT progresses by creating the all cyclic shifts of the uncompressed data string and arranging those cyclic shifts lexicographically. The BWT output has two parts. The first part is a length- string which gives the last character of each of the cyclic shifts. The next part is an integer which describes the position of the uncompressed data in the ordered list. For instance, BWT of the word “bananas” is given in [Fig. 1]. $BWT(banana) = bnnsaaaa, 4)$. While performing inverse BWT, reversible sequence transformation, the table should be reconstructed by the use of last column in the [Table 1], i.e. BWT output. Naturally, the reconstruction process of the table will be done in a column by column manner. Using the table construction, first column of the table is an ordered copy of the last column of the table. Hence, the *first* column can be reconstructed by replacing the alphabets on the list found in the last column. For reconstructing the second column, it is clearly shown that every row is a cyclic shift of every another row, and so last and first columns together gives a list of all successive pairs of symbols. Sorting this list of pairs produces the first and *second* column of the table. This process is repeated for third, fourth column and so on till all columns in the original table is reconstructed. The transform index represents the preferred row of the entire table. Inverse BWT is shown in [Fig. 2], $BWT^{(-1)}(bnnsaaa, 4) = banana.$

The steps involved in BWT are given below.

Step 1: Identify every cyclic sequence of the given data series
Step 2: Arrange the cyclic shift (rows) lexicographically
Step 3: The last column of the given array is given as output and row index of the input sequence.

The steps involved in inverse BWT are given below. For i=1, 2,.., n-1,

Step 1: Place column *n* in front of columns 1, 2,..., i-1
Step 2: Arrange the resulting length strings lexicographically
Step 3: Place the ordered list in the first I columns of the table.

IIOAB JOURNAL
ISSN: 0976-3104

	Step 1						
1	**b**	**a**	**n**	**a**	**n**	**a**	**s**
2	s	b	a	n	a	n	a
3	a	s	b	a	n	a	n
4	n	a	s	b	a	n	a
5	a	n	a	s	b	a	n
6	n	a	n	a	s	b	a
7	a	n	a	n	a	s	b

	Step 2							Step 3
1	a	n	a	n	a	s	b	b
2	a	n	a	s	b	a	n	n
3	a	s	b	a	n	a	n	n
4	**b**	**a**	**n**	**a**	**n**	**a**	**s**	s
5	n	a	n	a	s	b	a	a
6	n	a	s	b	a	n	a	a
7	s	b	a	n	a	n	a	a

Fig. 1: BWT of a sequence "banana".

	$(i=1)$					$(i=2)$				
	1	2	3			4	5	6		
1	b	a	a	⋯	b	ba	an	an	⋯	b
2	n	a	a	⋯	n	na	an	an	⋯	n
3	n	a	a	⋯	n	na	as	as	⋯	n
4	s	b	b	⋯	s	sb	ba	ba	⋯	s
5	a	n	n	⋯	a	an	na	na	⋯	a
6	a	n	n	⋯	a	an	na	na	⋯	a
7	a	s	s	⋯	a	as	sb	sb	⋯	a

	$(i=3)$			$(i=4)$		
	7	8	9	10	11	12
1	ban	ana	ana⋯b	bana	anan	anan⋯b
2	nan	ana	ana⋯n	nana	anas	anas⋯n
3	nas	asb	asb⋯n	nasb	asba	asba⋯n
4	sba	ban	ban⋯s	sban	bana	bana⋯s
5	ana	nan	nan⋯a	anan	nana	nana⋯a
6	ana	nas	nas⋯a	anas	nasb	nasb⋯a
7	asb	sba	sba⋯a	asba	sban	sban⋯a

	$(i=6, cont.)$
	18
1	ananasb
2	anasban
3	asbanan
4	**bananas**
5	nanasba
6	nasbana
7	sbanana

Fig. 2: IBWT of a sequence "(bnnsaaaa, 4)".

PERFORMANCE EVALUATION

To ensure the effectiveness of the BWT algorithm, its lossless compression performance is compared with 4 different, well-known compression algorithms namely Huffman coding, AC, LZW and LZMA.

Metrics

In the section, various metrics used to analyze the compression performance are discussed. The performance metrics are listed below: CR, CF, and BPC.

Compression Ratio (CR)
CR is defined as the ratio of the number of bits in the compressed data to the number of bits in the uncompressed data and is given in Eq. (1). A value of CR 0.62 indicates that the data consumes 62% of its original size after compression. The value of CR greater than 1 result to negative compression, i.e. compressed data size is higher than the original data size.

$$CR = \frac{\text{No. of bits in compressed data}}{\text{No. of bits in original data}} \quad (1)$$

Compression Factor (CF)
CF is a performance metric which is the inverse of compression ratio. A higher value of CF indicates effective compression and lower value of CF indicates expansion.

$$CF = \frac{\text{No. of bits in original data}}{\text{No. of bits in compressed data}} \quad (2)$$

bits per character (bpc)
bpc is used to calculate the total number of bits required, on average, to compress one character in the input data. It is defined as the ratio of the number of bits in the output sequence to the total number of characters in the input sequence.

$$bpc = \frac{\text{No. of bits in compressed data}}{\text{No. characters in the original data}} \quad (3)$$

Dataset description

For experimentation, a dataset which is used to identify seismic bumps larger than 104 J is chosen [13]. These data were collected from the mines in Poland with the time interval of 8 hours. The dataset consist of 2584 samples with two different classes 0 and 1. The value 0 indicates no earthquake hazard in the succeeding time interval and 1 indicates the occurrence of the earthquake hazard.

IIOAB JOURNAL
ISSN: 0976-3104

RESULTS AND DISCUSSION

To highlight the good characteristics of BWT based lossless compression algorithm for seismic dataset, it is compared with 4 states of art approaches namely Huffman coding, AC, LZW and LZMA. A direct comparison is made with the results of existing methods using the same dataset. Table 1 summarizes the experiment results of compression algorithms based on three compression metrics such as CR, CF and bpc. As evident from Table 1, the overall compression performance of BWT coding algorithm is significantly better than other algorithms on applied seismic bump dataset. It is observed that BWT coding achieves higher compression than existing methods. It is noted that Huffman coding and arithmetic coding produces poor results than BWT, LZW and LZMA.

The compression performance on applied dataset files reveals some interesting facts that the compression algorithms perform extremely different based on the nature of applied dataset. The existing LZW and LZMA method achieves better compression performance than Huffman coding and AC, but it fails to give performs well than BWT algorithm. Likewise, Huffman and Arithmetic coding also produce appropriately equal compression performance. This is due to the fact that the efficiency of the arithmetic coding is always better or at least identical to a Huffman code. Similar to Huffman coding, Arithmetic coding also tries to calculate the probability of occurrence of particular symbols and to optimize the length of the necessary code. It achieves an optimum code which exactly corresponds to the theoretical specifications of the information theory. A minor degradation results from inaccuracies, because of the correction operations for the interval division.

Table 1: Performance comparison of various compression methods on seismic data

Dataset	Compression techniques	Compression Ratio (CR)	Compression Factor (CF)	bits per character (bpc)
Seismic dataset	BWT	0.167	5.962	1.342
	LZMA	0.193	5.174	1.546
	Huffman	0.389	2.568	3.114
	LZW	0.285	3.512	2.277
	Arithmetic	0.383	2.608	3.066

On the other side, Huffman coding generates rounding errors because of its code length and is limited to multiples of a bit. The variation from the theoretical value is more than the inaccuracy of arithmetic coding. In overall, BWT coding results in effective compression than existing methods. Generally, dictionary based coding approaches find useful in situations where the original data to be compressed involves repeated patterns. LZMA extends LZW with the range encoding technique which enables to produce significantly higher compression than LZW. Interestingly, LZMA achieves significantly better compression with the compression ratio of 0.193 at the bit rate of 1.546 bpc. But, BWT achieves higher compression with the compression ratio of 0.167 at the bit rate of 1.342bpc.

CONCLUSION

This paper employs a Burrows Wheeler Transform (BWT) coding to compress seismic data. BWT is a block sorting compression which rearranges the character string into runs of identical characters. Extensive experiments were performed using real world seismic bump dataset. To ensure the effectiveness of lossless BWT coding, it is compared with 4 well-known compression algorithms namely Lempel Ziv Markov Algorithm (LZMA), Huffman coding, Lempel-Ziv-Welch (LZW) coding and Arithmetic coding (AC). By comparing the compression performance of BWT coding with existing methods, BWT achieves significantly better compression with a compression ratio of 0.167 at the bit rate of 1.342 bpc.

CONFLICT OF INTEREST
There is no conflict of interest.

ACKNOWLEDGEMENTS
None

FINANCIAL DISCLOSURE
None

ISSN: 0976-3104

REFERENCES

[1] M Bilen, AH Işık, T Yiğit. [2015] Seismic hazard prediction with classification of seismic pulses, International Burdur Earthquake & Environment Symposium (IBEES2015) Burdur, Turkey. 41-48.

[2] E Celik, M Atalay,H Bayer. [2014] Earthquake prediction using seismic bumps with Artificial Neural Networks and Support Vector Machines, Signal Processing and Communications Applications Conference, Trabzon, Turkey. 730-733.

[3] RW Hamming. [1986] Coding and information theory.

[4] Huffman DA [1952]. A Method for the Construction of Minimum-Redundancu Codes. 1098–1102.

[5] Vitter Jeffrey S. [1987] Design and Analysis of Dynamic Huffman Coding. 34: 825–845SW Drost, N Bourbakis. [2001] A Hybrid system for real-time lossless image compression. Microprocess. Microsyst. 25: 19–31. doi:10.1016/S0141-9331(00)00102-2

[6] SW Drost, N Bourbakis. [2001] A Hybrid system for real-time lossless image compression. Microprocess. Microsyst. 25: 19–31. doi:10.1016/S0141-9331(00)00102-2

[7] DA Huffman. [1952] A Method for the Construction of Minimum-Redundancu Codes. 1098–1102.

[8] H Witten Ian, M Neal Radford, G Cleary John. [1987] Arithmetic coding for DC. *Commun. ACM.* 30(6): 520–540.

[9] J Ziv. A Lempel. [1977] A Universal Algorithm for DC. *IEEE Trans. Inf. Theory*, 23(3): 337–343.

[10] J Ziv, A Lempel. [1978] Iz78.pdf. IEEE. 530–536.

[11] TA Welch. [1984] A technique for high-Performance DC. *IEEE*. 8–19.

[12] M Burrows, D Wheeler. [1994] A block-sorting lossless DC algorithm. *DC*. 124:18.

[13] https://archive.ics.uci.edu/ml/datasets/seismic-bumps