International Journal of Computer Science and Engineering (IJCSE)
ISSN(P): 2278-9960; ISSN(E): 2278-9979
Vol. 2, Issue 5, Nov 2013, 101-110

International Academy of Science, Engineering and Technology
Connecting Researchers; Nurturing Innovations

WEIGHT BALANCED LOSSLESS DATA COMPRESSION ALGORITHM

SANTANU SANTRA & JAYASHRI DEB SINHA

Assistant Professor, Department of Computer Science and Engineering, Bengal Institute of Technology and Management Santiniketan, West Bengal, India

ABSTRACT

Data compression technique is to reduce the size of a file to minimum size as the reason the require space is to store the file is less than the original one. In this paper we propose a new technique known as Weight Balanced Data Compression Algorithm (WBDCA), which calculate the weight for each intensity value and depending upon the weight each time some code is given to each intensity value. This technique will manipulates each bit of data inside file to minimize the size without losing any information after decoding which is treated as lossless compression. This basic technique is intended to be combining with other data compression algorithm to optimize the compression ratio. The performance of this algorithm is measured by comparing combination of different data compression algorithm.

KEYWORDS: Data Compression, Intensity Value, Lossless Compression

INTRODUCTION

Recent trends in location acquisition technologies [1], such as real time video conferencing [2], Global Positioning System (GPS) [3] and wireless networks [4] have raised many novel applications like video transferring, object tracking, environmental monitoring and several location dependent techniques. These applications generate a large amount of location data, which lead to transmission and storage challenges, especially in resource constrained environments like WN. Various algorithms have been proposed for data compression and data aggregation [5, 6] to reduce the file size without losing any information when the file decoded.

Data compression is a procedure to reduce storage cost and transmission cost by reducing redundancies that occur in most files.

Data compression technique classified into two types as bellow

- **Lossy Compression:** reduced the file size by eliminating some unneeded data that won't be recognize by human visual system after decoding. These types of compression are mostly used by audio and video compression algorithm [7].
- **Lossless Compression:** calculate each bit of data inside file to minimize the file size without losing any information after decoding [8].

Most compression methods are physical and logical [9,10]. Physically because it translate the input bit-stream into another one format the only way to understand and decode of the output bit- stream is by knowing how it was encoded. Logically because it takes individual intensity vale in source bit-stream and replace common contents with short codes. Logical compression method is effective and useful for certain type of information. Lossless data compression algorithm often used to better use disk space on the storage applications or the communication bandwidth in a WN. For spreadsheets, text, executable programs lossless data compression is necessary as changing even a single bit cannot tolerated.

Data Redundancy: - Data Redundancy [10] can be defined as

$$\sum_{i=1}^{n} p\big(a(i)l(i)\big) - \sum_{i=1}^{n} -p(a(i)\log p(a(i))) \quad (1)$$

Where l (i) is the length of the encoded data representing message a(i). The expression SUM p (a(i)) l(i) represents the lengths of the encoded data weighted by their probabilities of occurrence, that is, the average encoded data length.The expression SUM [-p(a(i)) lg p(a(i))] is entropy, H. Thus, redundancy is a measure of the difference between average encoded data length and average information content. If a code has minimum average encoded data length for a given discrete probability distribution, it is said to be a minimum redundancy code.

Entropy: Entropy rate is denoted by H. The exact value of H depends on the information source, the statistical nature of the source. It is possible to compress the source, in a lossless manner, with compression rate close to H. It is mathematically impossible to do better than H [11].

Let R_n be the rate of an optimal n-th order lossless data compression code (bits/symbol) then

$$-\frac{1}{n}\sum p(B_n)\ \log_2 p(B_n) \leq R_n < -\frac{1}{n}\sum p(B_n)\ \log_2 p(B_n) + \frac{1}{n} \quad (2)$$

Since both upper and lower bounds of R_n approach the entropy rate, H, as n goes to

Infinity, we have $$\lim_{n\to\infty} R_n = H \quad (3)$$

From equation 1 & 2 we can conclude entropy rate is the rate of an optimal lossless data compression code. The limit exists as long as the source is stationary.

Prefix-Free Code: Is the bit string representing some particular symbol is never a prefix of the bit string representing any other symbol [12].

Probability Distribution: A probability distribution function [13] is a function f defined on an interval (a, b) and having the following properties.

- f(x) \geq 0 for every x
- \displaystyle\int_a^b f(xb)\dx=1

In the above example a and b both to be infinite. This would make the integral in (b) an improper one statically it says is an integral in which (A) one or both of the limits of integration is infinite, or (B) in which the integrand becomes infinite somewhere in the range of integration.

Compression Ratio: Compression ratio is defined as the ratio of an original data and compressed.

Compression Ratio = Original Image Size / Compressed Image Size

Entropy: The entropy measure conveys the information contained in the image and any compression scheme should retain the entropy measure while reducing the redundant information from the image.

Source Entropy: The entropy of original image.

Destination Entropy: The entropy of reconstructed image

This paper is divided into various sections. Where 1st section describes the introductory part of the research work. Section 2 discuss several data compression algorithms. Section 3 gives the clear idea over the technique that is proposed in

this research work with the help of several proposed algorithms. Section 4 describes the algorithm analysis of the proposed technique. Section 5 describes comparative study of varies lossless data compression algorithms and finally we finish this research article with concluding remarks.

RELATED WORK

Data compression [13] helps reduce resources usage, such as data storage space or transmission capacity. Compressed data must be decompressed to use, this extra processing imposes computational or other costs through decompression; Data compression is subject to a space-time complexity trade-off. The design of data compression schemes involves trade-offs among various factors, including the degree of compression, the amount of distortion introduced in lossy compression and the computational resources required to compress and uncompressed the data.

Approaches for lossless data compression

Run-Length Encoding (RLE)

The idea behind the technique is if a data item x appears n consecutive times in the input bit stream, replace the n occurrences with the single pair xn. This is most useful on data that contains many such occurrences: for example, simple graphical images such as icons, BMP, TGA and TIFF images. The RLE [14] algorithm is not suitable with file that don't have many weight or runs as it could greatly increase the file Burrows-wheeler transform

Dictionary Coders

The idea behind the technique is searching for matches between the information to be compressed and a set of strings contained in a data structure or dictionary maintained by the encoder. After finding match replacements a reference to the string's position in the dictionary [15].

In static dictionary a full set of strings is coded before compression begins and the code is fixed during the compression process. The approach is used when a file or set of to be encoded is fixed and large. A personal digital assistant (PDA) generally builds a static dictionary from a concordance of the text and then uses that dictionary to compress the file. In dynamic dictionary some predefined code for data is present but the code may change during the encoding process. This technique is followed by LZ77 [16] and LZ78 [17] method.

In LZ77 a data structure or dictionary called "sliding window" is used to hold the last N bytes of data processed, effectively storing every substring that appeared in the past N bytes as dictionary entries. In this approach to identifying a dictionary entry two fields are needed the length field, represent the length of the matched information and the offset represent the match is identifying in the sliding window starting offset bytes before the current information.

Burrows-Wheeler Transform (BWT)

The BWT method for compress data or information converts a normal text file into a file where series of the same letter occur near each other many times [18].

Prediction by Partial Matching (PPM)

The main idea behind the method is based on context modeling and prediction. Context modeling is the boundary between the symbol or information and its environment and predicted the partial matching with the help of a set of previous symbols in the uncompressed symbol stream to predict the next symbol in the stream and reduce the symbol ranking [19].

If no prediction can be find based on all n context symbols a prediction is attempted with n-1 symbols. The process is repeated until a match is found or there is no more symbol remains in context modeling. In this situation DMC (Dynamic Markov Chain) compression algorithms represent a fixed prediction consist of zero-order model.

Context Mixing (CM)

In this approach the next-symbol predictions of two or more statistical models are combined to yield a prediction that is often more accurate than any of the individual predictions is one simple method is to average the probabilities assigned by each symbol. Combining models is an active area of research in machine learning [20]. PAQ series of data compression programs use context mixing to assign probabilities to individual symbols of the input.

Huffman Coding

In this entropy encoding method the term refers to the use of a variable length code table for encoding an input symbol where the variable-length code table has been derived in a particular way based on the estimated probability of occurrence for each possible value of the input symbol [12].

Huffman coding uses a specific method for representation for each symbol, resulting in a prefix-free code that expresses the most common characters using shorter strings of bits rather than are used for less common input symbols. The method is most efficient compression method of the lossless data compression. No other mapping of individual input symbols to unique strings of bits will produce a smaller average output size when the actual symbol frequencies agree with those. Used to create the code through Huffman coding procedure optimal code length for a symbol-by-symbol coding with a known input probability distribution, its optimality can sometimes accidentally be over-stated and the method is not use full when input probabilities are not precisely calculated.

Adaptive Huffman Coding

The method is based on Huffman coding to produce the encoded code as the symbols are being transmitted, having no initial knowledge of source distribution that allows one-pass encoding and adaptation to changing conditions in symbols [21]. The benefit of one-pass procedure is that the source can be encoded real-time, though it becomes more sensitive to transmission errors, since just a single loss ruins the whole code.

Arithmetic Coding

The method is a form of entropy encoding for lossless data compression, but where other entropy encoding techniques separate the input symbols into its component symbols and replace each symbol with a code word, arithmetic coding encodes the entire message into a single number, a fraction n where ($0.0 = n < 1.0$) [22].

PROPOSED WEIGHT BALANCED DATA COMPRESSION ALGORITHM

The proposed algorithm for lossless data compression take a set of symbols or string and calculate the weight, if a symbol is scanned first time the weight will be 1 and assign a code in the form of weight tree(WT) and depending upon the weight it should choose its proper place to get encoded code in descending order . The process will be continuing until all encoded codes for symbol is generated. After finding the encoded code each symbol with in the string will be replaced by the encoded code.

Weight Tree: Here we proposed a tree which can be constructed using the weight (number of occurrence) of the symbol. Weight for a symbol will be increased by 1 if symbol is scanned previously. All symbols will arrange such a way that all

symbols along with highest weight will be stored in the lowest level then the symbols along with low weight If weight for a symbol is high then it will be interchanged the position of symbols. The proposed WBDC algorithm consist of a rule for describing strings of symbols from a finite set of symbol A into substrings or words may consist of one symbol and the lengths do not exceed a predefined integer Ls and the coding technique which create maps for these substrings sequentially into uniquely decipherable code words of fixed length Lc over the same of alphabet. The word-length bounds Ls and Lc allowed for bounded delay encoding and decoding and they are related by

$$L_c = 1 + [\log (n - L_s)] + [\log L_s] \quad (1)$$

Where logarithm base is the cardinality α of the set of alphabet A, and n is the length of buffer used at the encoding end of the process to store the least n symbols emitted by the source. The relation between n and Ls can be represented as

$$n \cong L_s \, \alpha^{hL_s} \quad (2)$$

Whereas 0<h<1. For on-line coding, a buffer of similar lengths has to be employed at the decoding end.

Theorem: Let d denotes the number of symbol in a message . Then the height h of weight tree WT is O (log d)

Proof of Theorem: Let L (h) denote the minimum number of symbols of a balanced binary search tree having height h. then D (0) =1, D (1) =2, D (3) =3 and for d>=2,

$$D(h) = 1 + D(h-1) + D(h-2) \quad (3)$$

Thus D (h) has exponential growth. Hence, there exists a>1 and C>0 such that

$$D(h) \geq Ca^h \quad (4)$$

For sufficiently large h, from equation (4) we get

$$h \leq \log_a D(h) - \log_a C \leq \log_a d - \log_a C = O(\log n) \quad (5)$$

Proposed Algorithm for Weight Balance Tree

In our proposed method, Weight Tree (WT) is required to compress a string or a message to produce the encrypted data and to store the encrypted data for each symbol. Algorithm 1 is used to generate the Weight Tree (WT) for the proposed method. Here A is represent symbols find in string or message. W is weight for the corresponding symbol in A.COUNT indicates the number of symbols present in the string or message. S is the next input symbols find in the string or message. W (T) is the weight for the previous symbol present in A.

Algorithm 1

- Initialize A,W and COUNT.
- S is next Input Symbol.
- Search A for S
    - S Not found then store S in next location of A and assign the weight as 1 in the same position of W and increase COUNT by 1 . goto Step 2.
    - S found then increase same position value of W by 1

- If (W(S) > W(T)) then
    - Interchange the position of both the symbol and weight in A and W.
- If S= NULL then goto Step 6.

  Otherwise goto Step 2.
- END

The proposed method (algorithm 2) use algorithm1 to generate the symbol table (A), weight table (W) for each symbol and numbers of symbols (COUNT) present in A. X is the input string and the proposed method read each symbol until A is NULL and calculate the number of symbols it compare (CN) and according to the CN the symbol will be encoded. C is the next symbol in the input string X. Y is a binary array to store compressed data and variable k is used to represent the location in Y. proposed method is used for data compression, each input symbol is mapped into encoded form.

Algorithm 2

- Initialize K=0;
- Read next symbol C from X.
- if C=NULL goto step 9
- CN:=0,
- Search C in A
    - If C≠A then increase the value of CN by 1
- CN1:=CN
- Until CN =0 do
    - Y[K]=1
    - K=K+1

  End
- If (CN=0 and CN1%2=0)
    - Y[K] :=1
- If (CN=0 and CN1%2 ≠ 0)
    - Y[K]:=0;
- If C≠NULL
    - Goto Step 2.
- OUT PUT = Y
- Stop

Let n number of different symbols present in a string and the height (h) of WT then

There is a relation in between h and n

$$h = \begin{cases} n-1 & if\ n > 0 \\ 1 & if\ n = 0 \end{cases}$$

The first step of the proposed approach is to create symbol table and weight table for different symbols with the help of weight tree in descending order. Second step is to create a series of source reductions by mapped into fixed sequences of code symbols which treated as optimal codes.

EXPERIMENTAL RESULTS

The proposed method is best illustrated by an example. Consider S is a string of symbols given bellow

S= {ababacdcdadbcdbbecbad}

- Algorithm 1 generate the symbol table(ST) and the weight table(W) which contain weight for relevant symbol

  ST= {b,a,d,c,e} and corresponding weight table for relevant symbol is

  W = {6,5,5,4,1}

- Algorithm 2 scan each input symbol and store the corresponding optimal encoded code in the output file.

- The optimal code for each symbol b is 0;a is 10;d is 110; c is 1110 and d is 1111

  Y={10010010110011011101000110000111111100101110}

The total number of bits required to represent the string is number of symbols x bits require to represent symbols. Most of the system used ASCII character to represent a symbol and the bit length for each symbol is 8 bits. The entire string size is 23 x 8=168 bits after compression by the proposed method the size decrease and the size is 43 bits. The resulting compression and corresponding relative redundancy are

The compression ratio is $CR = \text{actual data size}/\text{Encoded data size}$

$$CR = \frac{21\ X\ 8}{43} = \frac{168}{43} \cong 3.91$$

The relative redundancy is $R = 1 - \frac{1}{CR} = 1 - \frac{1}{3.91} = 0.744$

Statically 74.4 % of the data in the input string is redundant. To decompress the file algorithm first read each bit of compress file until it get 0. Upto 0 it will be treated as a encoded code for a string. After getting the encoded code it search the ST and

COMPARISON

The data set S is tested using several data compression algorithms and the following table 1 shows the time complexity for several algorithms. The proposed algorithm generates the encoded data for each symbol by the first iteration and can encode the entire data in the next pass to generate the encoded file or data set.

Table 1: Summary of Several Lossless Data Compression Algorithms

Algorithm	Type	Time Complexity	Compression
Run-Length Encoding (RLE)	Statistical	O(n)	Average
dictionary coders	Dictionary	Context dependent	Good
Burrows-Wheeler transform (BWT)	Dictionary	$O(n \log n)$	Average
prediction by partial matching (PPM)	Pattern	Context dependent	Good
context mixing (CM)	pattern	$O(n^2)$	Good
Huffman coding	Statistical	$N[n + \log(2n-1)] + S_n$	Average
Adaptive Huffman coding	Statistical	$O(n * \log \lvert\Sigma\rvert)$	Average
Arithmetic coding	Statistical	$N[\log(n) + a] + S_n$	Average
Weight Balanced Data Compression Algorithm	Statistical	O(n)	Good

CONCLUSIONS

Weight Balanced Data Compression Algorithm (WBDCA), which calculates the weight for each intensity value and depending upon the weight each time some code is given to each intensity value. This technique will manipulates each bit of data inside file to minimize the size without losing any information after decoding which is treated as lossless compression. This basic technique is intended to be combining with other data compression algorithm to optimize the compression ratio.

REFERENCES

1. Zhou Hongxia, Zhang Dezheng , Shan Ping, "Research on Dynamic Knowledge Acquisition Technology Based on Man-Computer Interaction" IEEE conference ISBN 978-1-4244-6977-2 27-28 Nov. 2010 .
2. *J. Park, K. Seshadrinathan, S. Lee and A.C. Bovik*, "VQ Pooling: Video quality pooling adaptive to perceptual distortion severity" IEEE Transactions on Image Processing, Vol: 22 No: 2, February 2013 Page(s): 610-620.
3. Xu, S., Zhang, H., Yang, Z., "GPS Measuring Principle and Application", 3rd edn., pp. 1–10. Wuhan University of Technology Press, Wuhan (2008)
4. T. F. La Porta, K.K. Sabnani, and R.D. Gitlin, "Challenges for Nomadic Computing: Mobility Management and Wireless Communications," Mobile Networks and Applications, Vol. 1, 1996, pp. 3-16.
5. K. Dasgupta, K. Kalpakis, and P. Namjoshi, "An Efficient Clustering-based Heuristic for Data Gathering and Aggregation in Sensor Networks", Wireless Communications and Networking, 2003. WCNC 2003. 2003 IEEE Vo.3
6. K. Dasgupta et al., "Maximum Lifetime Data Gathering and Aggregation in Wireless Sensor Networks", In *Proc. of IEEE* Networks'02 Conference, 2002.
7. Francisco E. Eraso, Mostafa Analoui, "Impact of lossy compression on diagnostic accuracy of radiographs for periapical lesions", Scienc direct, Oral Surgery, Oral Medicine, Oral Pathology, Oral Radiology, and Endodontology, Volume 93, Issue 5, May 2002, Pages 621–625
8. Giridhar Mandyam, Nasir Ahmed, Neeraj Magotra, "Lossless Image Compression Using Predictive Codebooks", Sciencedirect- Digital Signal Processing Volume 7, Issue 3, July 1997, Pages 147–152.
9. Salomon, D. 2004. Data Compression the Complete References Third Edition. Springer-Verlag New York, Inc.

10. Matthew J. Zukoski, Terrance Boult, "A novel approach to medical image compression" Int. J. Bioinformatics Research and Applications, Vol. 2, No. 1, 2006, pp 89-103.
11. Borda, Monica, "*Fundamentals in Information Theory and Coding"*. Springer (2011). ISBN 978-3-642-20346-6, pp-11.
12. Thomas H. Cormen, Charles E. Leiserso, "*Introduction to Algorithms"*, Second Edition. MIT Press and McGraw-Hill (2001), ISBN 0-262-03293-7pp. 385–392.
13. Corder, G.W., Foreman, D. I., "*onparametric Statistics for Non-Statisticians: A Step-by-Step Approach*" Wiley 2009, ISBN 978-0-470-45461-9.
14. M. Banikazemi, "LZB: Data Compression with Bounded", Proceedings of the Data Compression IEEE Computer Socity Conference, 2009 .
15. Red Comet. "Dual-Tile Encoding: NES/Famicom Implementation". Dual-Tile Encoding.
16. J. Pike, "Text compression using a 4 bit coding scheme", The Computer Journal, Vol. 24 No. 4 1981 pp.324-330.
17. Michael Dipperstein," Lempel-Ziv-Welch (LZW) Encoding: Discussion and Implementation"
18. Salson M, Lecroq T, Léonard M and Mouchard L . "A Four-Stage Algorithm for Updating a Burrows–Wheeler Transform". *Theoretical Computer Science* 410 (43): 4350 (2009)
19. Cleary, J. Witten, I (April 1984). "Data Compression Using Adaptive Coding and Partial String Matching". *IEEE Trans. Commun.* 1984 Vol.32 No. 4 pp. 396–402.
20. Mahoney, M., "Adaptive Weighing of Context Models for Lossless Data Compression", *Florida Tech. Technical Report 2005*.
21. J. S. Vitter, "Design and Analysis of Dynamic Huffman Codes", Journal of the ACM, vol. 34 No.4, 1987, pp 825–845.
22. Rissanen J and Langdon G., "Arithmetic coding", IBM Journal of Research and Development, 1979 Vol. 23, No. 2, pp 149-162.