*International Journal of Engineering & Technology,* 7 (4.5) (2018) 583-586

International Journal of Engineering & Technology

*Website: www.sciencepubco.com/index.php/IJET*

***Research paper***

# Effective storage of source code of the student's projects in digital libraries

**Rexline S. J[1,*], Albert William M [2]**

[1] *Department of Computer Science Loyola College, Chennai, India*
[2] *Department of Mathematics Loyola College, Chennai, India*
**Corresponding author E-mail: rexlinegerard@gmail.com*

**Abstract**

In higher educational institutions, the source code of the student's projects and their documentations should be submitted in both printed and electronic form. The electronic form of storage smooth the progress of computerized processing of the documents for purposes such as plagiarism detection and for future references. Considering the higher education system, there are several hundred theses added to the archive every year. The primary motivation for this paper was to reduce the storage requirements of the student's projects and their documentation's electronic archive in higher education institutions.

***Keywords***: *Compression Ratio; Image Compression: Source Code; Text Compression; Text Transformation.*

## 1. Introduction

Now a days, In higher educational institutions like Universities and Colleges, Students are asked to submit their projects like source code of their development, project manuals and any appen- dixes in the form of Microsoft Word compatible documents such as DOC, DOCX and PHP,HTML,XML, PS ,C, C++, C#, Java and
PDF file format too. Some of the appendixes may be multimedia, in formats such as PNG, JPEG, or MPEG.3. To store all these documents needs a lot of storage spaces in the digital library of the institutions. Every year, the data of the institutions continues to grow at an exponential rate. The increasing utilization of the source code and the documents of the students by the upcoming students are not allowing the documents to be removed from the database. Since the documents occupy a large amount of space it becomes necessary to compress and store these heterogeneous types of data. In the field of the lossless compression algorithm, lots of different types of compression algorithms are developed by the researchers to reduce the storage space required by these het- erogeneous types of data. Some of the algorithms are like Huff- man, arithmetic and LZ families of Gzip and Unix-compress, PPM compresses, LZW is used for text documents and for the purpose of image reduction the compression algorithm called Deflate used in PNG, MNG, and TIFF format files. In this paper, the use of text compression algorithms, image compression algorithms and text transformation algorithms are explored to develop a tool for storing the student's project documents and source code of their projects.
This paper is structured as follows: Section II presents the existing heterogeneous types compression algorithms; Section III proposes the proposed tool to reduce the storage cost of the student's documents in the higher educational institutions. Section IV demonstrates the viability and ability of the proposed method .and finally, Section V contains the conclusions.

## 2. Heterogeneous compression algorithms

### 2.1. Text compression algorithms

The lossless text compression algorithm can be categorized into four different methods: Basic techniques, Statistical methods, Dictionary methods and Transform based methods. Basic tech- niques are generally inefficient and cannot compete with the more advanced methods; they are sometimes incorporated into efficient compression schemes as the last stage of compression. Examples are Run Length Coding, Move-to-Front Coding etc. The statistical methods explore the probability distribution of the symbols and achieve better compression ratio. The symbol or the group will be assigned the shorter length of the code for a higher frequency of occurrences. The most famous statistical method is PPM (Prediction by Pattern Matching) and Huffman coding, Arithmetic coding etc. Dictionary methods usually encode a sequence of characters with another symbol. The dictionary contains the information in the form of characters and its corresponding codeword symbol, and the dictionary may be implemented as static, dynamic or adaptive one. The structure of the dictionary can be implemented in an array or tree of data structure for speeding up of the location of information [2]. All category of Ziv and Lempel compression algorithm belong to the dictionary-based compression algorithm. Examples are LZ77, LZ78, and LZW etc. There are some tech- niques that do not directly compress the text but preprocess the text for the actual compression. The best example is the Burrows- Wheeler block sorting algorithm. The block sorting does not com-press the data at all but actually adds an extra data, the index value. After transforming the data by expanding it, the Move-to-Front algorithm is applied to that data; it does not compress the data but makes the data more compressible for the backend compression algorithm. The Huffman [7] or arithmetic coding [14] can com- press the data by assigning shorter codes to the frequently used higher order data that compress

**Copyright © 2018 Rexline S. J, Albert William M. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.**

the higher order data with less space. Compression algorithms like Huffman or arithmetic coding have not been attaining the expected compression ratio by the researchers for the text files. In the lossless text compression re- search field, the main goal is that the reduction of storage space by removing the redundancy and achieves a better compression ratio with a good time complexity.

### 2.2. Image compression algorithms

In the student's documents Images are need to be stored as the part of their work. Since Images are very important documents nowadays they need to be compressed to reduce the spaces. There are two different algorithms that perform this compression of in different ways; the first one is lossless image compression algorithms which keeps the same resolution of the images without any loss of data. It never compromises the quality of the images like medical images, Scanned document etc. The image like medical images, the scanned document needs very high resolution. The other type of image compression algorithm is lossy compression algorithms in which images resolution can be tolerated just to get better compression performance. Even though, lot of image compression algorithms designed by the researchers, some compression algorithm will perform well for specific kinds of images and that kind of algorithm will not perform well for another kind of images
BMP (bitmap) is a bitmap file that an uncompressed format of image files. PNG (Portable Network Graphics) is a bitmap image format that comes under the category of lossless data compression. TIFF (Tagged Image File Format) is a file format for mainly storing images, including photographs and line art [9]. The LZW comes under the dictionary-based compression algorithm is used in the TIFF image compression algorithm. JPEG (Joint Photo- graphic Experts Group) is an algorithm designed to compress the images with or without loss of the quality of the images.In case, it is a lossy compression algorithm that makes the algorithm very flexible one that the compression rate can be adjusted. In this compression algorithm, the quality of the image will not be satis- fied. Since more information will be lost, but the compression ratio of the image is better in which the image occupies less spaces. JPEG 2000 (Joint Photographic Experts Group 2000) is a wavelet- based image compression algorithm.

### 2.3. Text transformation compression algorithms

As we are aware, the text transformation techniques boost the compression rates up to few percent. The text transformation is a process, which reversibly transforms a data into some intermediate form. The transformed data can be compressed with most of existing lossless data compression algorithms, like bzip2, gzip with better compression efficiency than achieved using an untrans- formed data [5]. The reverse process is that decompression using given compressor like bzip2, Gzip and a reverse preprocessing transformation.
The Burrows-Wheeler Transform (BWT) [3] is one of the best transformation schemes in the lossless text compression research area. BWT is a reversible transform that converts the data into an intermediate format that iscompression algorithmmore compressible. The Burrows-Wheeler Transform (BWT) encodes a block of data separately as a single unit. Even though BWT makes the transformed data larger than its original form, but the transformed data is more compressible than the untransformed data. David J. Wheeler developed this algorithm in 1983. It was pre- sented publicly by Michael Burrows and David J. Wheeler in 1994 as a part of the block-sorting compression algorithm. Bzip2 com- pressing algorithm compresses files using the Burrows-Wheeler block sorting text compression algorithm, and Huffman coding. Bzip2 compresses large files in blocks. The block size influences both the compression ratio attained and the amount of memory needed for compression and decompression. The internal algo- rithms used in BWT like Move-To-Front, arithmetic coder are modified to improve the performance of BWT. Chapin [4] de- scribes the method of reordering the alphabets instead of using lexicographic sorting in BWT which gives better improvements in BWT Algorithm.
R. Franceschini and A. Mukherjee [5] proposed that the design of Star Encoding algorithm is to transform the text into some in- termediate form, which can be compressed with better efficiency. The star encoding makes use of * called signature of the character to replace certain characters in a word and maintain few characters so that the word is still retrievable. This star Encoding method achieved better gain in compression rates. And also they designed different preprocessing Schemes like Length-Preserving Transform (LPT)[1], Reverse Length-Preserving Transform (RLPT), Shortened –Context Length-Preserving Transform (SCLPT) [8]. In which SCLPT outperforms other transforms and achieves better compression rates. All these transformation methods used the static dictionary both in encoding and decoding process. En- crypted word based dictionary [10] also designed and tested which produced better results compared with the un-encrypted word based dictionary.
V.K. Govindan and B.S. Shajee Mohan [6] proposed that the actual codeword consists of the length of the code concatenated with the code and the codewords are created using the ASCII characters 33 to 250. ASCII characters 251 to 254 used to repre- sent the length of the code.A flag (ASCII 255) is used to indicate the absence of a space. If the character is one of the ASCII charac- ters 251-255, the character is placed twice so as to show that it is part of the text and not a flag. Md. Ziaul Karim Zia, Dewan Md. Fayzur Rahman and Chowdhury Mofizur Rahman describe a new transformation [15] that the code words are generated using the ASCII characters (33 - 128). Spaces between words and the unused bit of ASCII character representation from each character are re- covered to save one byte per eight ASCII characters.
Weifeng Sun, Amar Mukherjee and Nan Zhang suggested StarNT method [13]by introducing ternary search tree for encoding process and hashing to speed up the decoding process and used the alphabets [a-z, A-Z] for the codeword .Grabowski extended the StarNT transformation[11] with several different algorithms like Space stuffing around the words, EOL coding, Binary filtering technique, Capital conversion, n-gram replacement to improve the preprocessing techniques. His preprocessing algorithm proposed hashing method to speed up the transformation. Grabowski [12] proposed compression algorithm for Html documents too to attain better compression

## 3. Proposed compression tool for the students documents

In this section, new techniques are proposed in the storage of digital libraries of the higher educational institutions. Many digital libraries seriously lack of compression capabilities in storing the heterogeneous type of data. In this proposed method, Combination of lossless text compression algorithms, both lossless and lossy based image compression algorithms and also text transformation algorithms are used to give the reduction in the storage of student's project and its related documents.
In this tool, the special option is given to use the standard gen- eral purpose algorithms like WinZip, WinRAR, Bzip2, and 7Z to store the documents. In the case of storing the images, separate options can be given to select either lossless compression algo- rithm or lossy compression algorithm according to the category of the images. In the case of source code storage like C, C++, C#, VB, JAVA, HTML, XML, PHP codes, the text transformation algorithms can be used to give better performance in compression ratio. In this text transformation algorithm, a static dictionary is used to store the frequently used words. Not only single dictionary, multiple dictionaries are used according to the input document to be compressed. In this proposed method, separate static dictionar- ies are used to store programming language keywords like C, C++, C#, VB, and another dictionary for HTML, XML, and PHP key- word and references etc. Short code words are assigned to the words pre-

sent in the dictionary to do some precompression in the preprocessing stage itself. The size of the dictionary is limited according to the number of codewords available. This Encoder method takes advantage of repetitions in the data and so it achieves better context to the existing compressors. Each codeword should be unique. In this Coder, the codeword can be formed by one up to three bytes of the ASCII table from 128 to 255, since they are never used in text files. Code length also com- bined with the codeword in this proposed method. The reason why we have used the code length combined with codeword is to re- move the space between the words. Since, from Abel J and Teahan W that using this approach, it is possible to recover 16% of space on average from any text file. There is no codeword for digits, punctuation, tab, and EOL. So that the digits, punctuation, and EOL are transferred as it is. Capital conversion is a recognized preprocessing technique. Words started with a capital letter are converted to their lowercase equivalent and full uppercase words are also converted to its lowercase form and indicated the changes with a flag. hose words that are not present in the dictionary are not converted to codeword and it is transferred as it is and indicated with a flag. This flag is used to recover the space between the unaltered words present consecutively while decoding the text. The text document formats have individual characteristics; therefore the compression ratio can be improved by adapting the transform to a particular format.

## 4. Implementation issues and performance analysis

In this section, attention is given to analyzing the performance of the proposed tool by implementing the GUI design using ASP.Net and code behind is C++ language. In this design, the entire document is categorized according to its data type and stored as the separate document. And then proper compression algorithm can be selected by the students to store their documents and finally placed in the digital library.

The statistical compression algorithm used for comparison purposes is the Huffman Coding and Arithmetic Coding. The general-purpose compression algorithms involved in this performance analysis are WinZip, WinRAR, and Bzip2. Transformation algo- rithms used for compression algorithms are Burrow Wheeler Transform, Word Replacement Transform. The image compres- sion algorithms involved to analyze the data are JPEG compres- sion algorithm, wavelet-based image compression algorithm, DE- FLATE algorithm, LZW lossless compression algorithm.To ana- lyze the performance of this tool, few documents with heterogene- ous data are taken and categorized the documents according to its type and the appropriate compression algorithm is used. Instead of using the same compression algorithm for heterogeneous data, it is better to decide the compression algorithm, which gives better compression according to the data type.

In this section, the source file with collections of images, source code, and text contents are placed as a single document and compressed with standard general purpose compression algorithm. And then the contents are separated based on its type and stored as separate documents and then compressed with its appropriate compression algorithms. The results are then compared with the compressed document consists of all content together. The pro- posed method gives better compression performance. Table 1 shows the various image compression algorithm results. Table 2 shows the results of text transformation algorithm results. Table 3 and 4 shows the comparative results of text documents using dif- ferent general purpose compression algorithms and statistical compression algorithms respectively. The end user can select and store their content according to their data. The compression ratio is measured by the unit Bits per Character for text compression algorithms.

**Table 1:** Comparative Results of Images Using Different Image Compression Algorithms

| Files | uncompressed image Size[BMP] | Tagged Image File Format[TIFF] | Portable Network Graphics[PNG] | Joint Photographic Experts Group[JPEG] |
|---|---|---|---|---|
| Image1 | 2.27 MB | 87.4KB | 41.6KB | 35.2KB |
| Image2[lossy] | 768KB | no compression | no compression | 334KB[100%]<br>49.5KB[50% loss] |
| Image3[Lossless] | 13.1MB | 8.28MB | 6.93MB | 1.47MB |
| Image4[Lossless] | 696KB | 378KB | 258KB | 43.3KB |
| Image5[Lossless] | 1.01MB | 578KB | 745KB | 91KB |

**Table 2:** Comparative Results of Text Documents Using Different Text Transformation Algorithms

| Files | File size(Bytes) | BWT(BPC) | Bzip2 (BPC) | Bzip2+ WRT(BPC) |
|---|---|---|---|---|
| File1.txt | 111261 | 2.11 | 1.98 | 1.721 |
| File2.txt | 768771 | 2.85 | 2.42 | 2.131 |
| File3.txt | 610856 | 2.43 | 2.06 | 1.872 |
| File4.txt | 377109 | 2.83 | 2.52 | 2.330 |
| File5.txt | 53161 | 2.65 | 2.49 | 2.104 |
| File6.txt | 82199 | 2.61 | 2.44 | 2.069 |

**Table 3:** Comparative Results of Text Documents Using Different General Purpose Compression Algorithms

| Files | File Size (Bytes) | WinRAR (Bytes) | WinRAR (BPB) | WinZip (Bytes) | WinZip (BPB) | Bzip2 (Bytes) | Bzip2 (BPB) |
|---|---|---|---|---|---|---|---|
| File1.txt | 121024 | 34022 | 2.248 | 34775 | 2.298 | 32099 | 2.122 |
| File2.txt | 73308 | 20306 | 2.215 | 20780 | 2.267 | 19684 | 2.148 |
| File3.txt | 58864 | 16110 | 2.189 | 16624 | 2.259 | 15215 | 2.067 |
| File4.txt | 56737 | 15804 | 2.228 | 16228 | 2.288 | 14854 | 2.094 |
| File5.txt | 229354 | 66233 | 2.310 | 67757 | 2.362 | 62169 | 2.168 |
| File6.txt | 191737 | 53188 | 2.219 | 54561 | 2.276 | 50209 | 2.095 |

**Table 4:** Comparative Results of Text Documents Using Different Statistical Compression Algorithms

| Files | File Size (Bytes) | Arithmetic Coding (Bytes) | Arithmetic Coding (BPB) | Huffman Coding (Bytes) | Huffman Coding (BPB) |
|---|---|---|---|---|---|
| File1.txt | 121024 | 28776 | 1.902 | 29234 | 1.932 |
| File2.txt | 73308 | 18463 | 2.015 | 18354 | 2.003 |
| File3.txt | 58864 | 15077 | 2.049 | 14743 | 2.004 |
| File4.txt | 56737 | 14362 | 2.025 | 14212 | 2.004 |
| File5.txt | 229354 | 57858 | 2.018 | 57366 | 2.000 |
| File6.txt | 191737 | 46889 | 1.956 | 47962 | 2.001 |

## 5. Conclusion

Information Technology has changed tremendously the current century libraries when compared with conventional libraries. Tra- ditionally, libraries have been collecting a variety of sources and storing them in readiness for use by users. Information in digital form is stored electronically and accessed, where access to the digital library has no boundary or particular restrictions. It helps the students to increase the access of all types of information availabe both offline and online and also helps to preserve the original documents and manuscripts. It also improves the effective utilization of qualitative and quantitative resource sharing among students.

## References


[1] F. Awan and A. Mukherjee, "*LIPT: A Lossless Text Transform to Improve Compression*," Proceedings of International Conference on Information and Theory:Coding and Computing, IEEE Comput- er Society, pp. 452-460, April 2001.

[2] Abel,J, Teahan,W, "Universal Text Preprocessing for Data Compression",IEEE Trans.Computers,54(5)pp :497-507,2005.

[3] M. Burrows and D.J. Wheeler, "A Block-Sorting Lossless Data Compression Algorithm", SRC Research Report 124, Digital Systems Research Center, Palo Alto, CA, 1994.

[4] Chapin B, Tate SR."Higher Compression from the Burrows–Wheeler Transform by Modified Sorting", In Storer JA, Cohn M, editors, Proceedings of the 1998 IEEE Data Compression Confer- ence, IEEE Computer Society Press, Los Alamitos, California,pp.532,1998.

[5] R. Franceschini, H. Kruse, N. Zhang, R. Iqbal, and A. Mukherjee, "Lossless, Reversible Transformations that Improve Text Compression Ratio," Project paper, University of Central Florida, USA. 2000.

[6] V.K. Govindan, B.S. Shajee mohan, "IDBE – An Intelligent Dictionary Based Encoding Algorithm for Text Data Compression for High Speed Data Transmission Over Internet", Proceeding of the International Conference on Intelligent Signal Processing and Ro- botics IIIT Allahabad February 2004.

[7] Huffman, D.A.," A method for the construction of minimum- redundancy codes". Proc. Inst. Radio Eng., 40: pp: 1098- 1101.1952.

[8] H. Kruse and A. Mukherjee, "Preprocessing Text to Improve Compression Ratios", Proceedings of Data Compression Conference, IEEE Computer Society, Snowbird Utah, pp. 556, 1998.

[9] Paula Aguilera, "Comparison of different image compression formats", Project Report.

[10] Robert Franceschini, Amar Mukherjee, " Data Compression Using Encrypted Text ",proceedings of the third forum on Research and Technology, Advances on Digital Libraries,ADL 96,pp .130-138, May 1996.

[11] P. Skibiński, Sz. Grabowski and S. Deorowicz. "Revisiting dictionary-based compression". Software–Practice and Experience, pp.1455-1476, 2005.

[12] P. Skibiński, "Improving HTML Compression", Informatica 33 (2009) 363–373, 2009.

[13] Sun W, Mukherjee A, Zhang N. "A Dictionary-based Multi- Corpora Text Compression System". In Storer JA, Cohn M, edi- tors, Proceedings of the 2003 IEEE Data Compression Conference, IEEE Computer Society Press, Los Alamitos, California, pp .448 2003.

[14] Witten, I.H., R.M. Neal and J.G. Cleary, "Arithmetic coding for data compression",Commun.ACM, 30: pp : 520-540.,1987.

[15] Md. Ziaul Karim Zia, Dewan Md. Fayzur Rahman, and Chowdhury Mofizur Rahman, "Two-Level Dictionary-Based Text Compression Scheme", Proceedings of 11th International Conference on Computer and Information Technology, Khulna, Bangladesh.,pp.25-27 December, 2008.



**Copyright © 2018 Rexline S. J, Albert William M. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.**