A Simple Compression Scheme Based on ASCII Value Differencing

Tommy¹, Rosyidah Siregar¹, Imran Lubis¹, Andi Marwan E¹, Amir Mahmud H², and Mawaddah Harahap³

¹Department of Computer Science, Faculty of Technic and Computer Science, Universitas Harapan, Medan, 20217, Indonesia
²Department of Computer Science, Faculty of Technology and Computer Science, Universitas Prima Indonesia, Medan, 20111, Indonesia

E-mail: tommy@stth-medan.ac.id¹, rosyidahsiregar0705@gmail.com¹, imran@stth-medan.ac.id¹, andimarwanelhanafi@gmail.com¹, amirmahmud@unprimdn.ac.id², mawaddah@unprimdn.ac.id²

Abstract. ASCII characters have a different code representation where each character has a different numeric value between the characters to each other. The characters is usually used as a text message communication has the representation of a numeric code to each other or have a small difference. The value of the difference can be used as a substitution of the characters so it will generate a new message with a size that is a little more. This paper discusses the utilization value of the difference of characters ASCII in a message to a much simpler substitution by using a dynamic-sized window in order to obtain the difference from ASCII value contained on the window as the basis in determining the bit substitution on the file compression results.

1. Introduction
Compression is generally divided into two categories, namely compression lossless and lossy compression which a lossless compression is the reversible while lossy compression is irreversible. Compression of digital data generally use lossless compression due to data that has been compressed must be returned into the original data so that it can be used. There are a few exceptions in the digital multimedia files like digital images, sound, video compressed files which can still be used so that the process is not fully needed decompression which is different to other digital files such as text, documents, etc.

Most of the methods of compression using statistic frequency of occurrence characters and bytes. Methods such as Huffman Codes process symbols in ASCII encodes the digital file into a row of new bit simplier [1]. Other methods such as LZW are also using the same approach, but have slightly different operating [2]. ASCII code is the numeric value that represents the characters or command on a computer device. The character used in a text message is the characters of the alphabet and punctuation characters that has a numeric value of adjacent and have difference or difference which is quite small. The small difference that it can be used as a substitution of the sequence of characters of sources resulting in a row of bit or symbol of the new, more simple. Approach to simplification of the bits of the original symbol based on the proximity of his bit values can be found on other compression
methods such as Half-Byte Compression. The basic concept of the Half-Byte Compression was eliminated four-bit MSB on a sequence of characters that have four MSB are identical so that on the next symbol using only four bits LSB so as to allow the use of the number of bits that are much more a little though there are several conditions that must be fulfilled before the compression can be done.

The though foundation of the development of a compression scheme which was done on this paper is the approach of the difference value in byte of the symbols on the Delta Encoding [3-4]. Delta Encoding using the difference value or delta as a substitution of symbols or character input that can be reconstructed again using a reference value.

\[ f_{new} = f_{old} + f_\delta \]

\( f_{new} \) is a new file the results of the reconstruction obtained from the sum of the \( f_{old} \) and \( f_\delta \). The Compression operation on delta encoding is counting \( f_\delta \) in such a way smaller so that it can be used in reconstructing \( f_{new} \) using \( f_{old} \) as the reference value. Delta Encoding is very effective in reducing the consumption of bandwidth on the network that has been implemented in the compression on the web service and HTTP in several studies [6-7]. Utilizing the basic concept of the difference from the value of a byte from an input symbol, on paper was developed the concept of a simple compression on the difference between ASCII value that has a smaller range than with using the concept of the difference between the value of the bytes.

2. Proposed Work: ASCII Value Differencing Based Compression

Differentiation method utilizing differential value of the ASCII characters in the ASCII table. Deferential value low enough can be used as a substitution of characters in a text message. On the application of simple method is not using the table lookup or reference tables used for encoding and decoding process. The process of encoding and decoding full use of the differential value of each character in the message. However, there was a possibility of table lookup implementation to improve the effectiveness of the methods on the various types of digital files.

2.1 Basic Concept

Encoding is done by calculating the numeric value of the lowest and highest characters in the message text using the ASCII table. The next differential will be calculated from the lowest and highest value. The differential value will be divided two to find the middle value of the differential value. The central value will be the value of the reference point or reference point. Encoding process is then carried out by doing the substitution value deference obtained from each character's response to the reference point.

The following is encoding stages of differential ASCII:

1. Looking for a character with ASCII code value of the lowest and highest.
2. Calculate the differential value of the lowest and highest character:

\[ d = \text{MaxC} - \text{MinC} \]

(1)

Where:
\( D = \) Differential value
\( \text{MinC} : \) The lowest value of ASCII code of the character message.
\( \text{MaxC} : \) The highest value of ASCII code of the character message.
3. Calculate the mid value of differential value by using a rounding over

\[ \text{Mid} = d / 2 \]

(2)
4. Looking for the value of reference point:

\[ p = \text{MinC} + \text{Mid} \]

(3)
5. Calculate the differential value each character against the value of reference point.
6. Do the encoding character added with the value of reference point as an initial byte.
Decoding is done by calculating the origin of ASCII character value using the differential
between differential characters are compressed against the value of reference point. The following
stages of the process of decoding:

1. Initial the Reference Point.
2. Calculate difference between each bit character encoded with the value of reference point.
3. Conversion differential value to ASCII characters

2.2 Proposed Scheme

On differentiation the basic ASCII, use the value of a reference or references from the midpoint
between the lowest value of the character with highest value of character, so that each character will
substituting with a bit of differentiation globally so that the bigger value of the differentiation so
compression results have bad compression ratios even uncompressed altogether.

1. Take some characters that found in the area of the window.
2. Calculate the differential value of the lowest and highest character of the window

\[ d = \text{MaxC} - \text{MinC} \]  (4)

Where:
\( d \) = Differential value
\( \text{MinC} \) : The lowest value of ASCII code of the character message.
\( \text{MaxC} \) : The highest value of ASCII code of the character message.

3. If \( d \leq 15 \) so the size of window can be zoom as needed during \( d < 15 \) where the expectation
is bit encoded result no more than four bits.
4. Do the encoding against ASCII code of the character based on the value of differential against
the value of reference point.

“123” = \{49, 50, 51\}

\( \text{Min} = 49 \)

\( \text{Max} = 51 \)

\( \text{dot P} = 49 + ((51 - 49)/2) = 50 \)

Differential:
\{49, 50, 51\} = \{-1, 0, 1\}, encode = \{11, 00, 01\}

“tommy” = \{116, 111, 109, 109, 121\}

\( \text{Min} = 109 \)

\( \text{Max} = 121 \)

\( \text{Point P} = 109 + ((121 - 109)/2) = 115 \)

Differential:
\{116, 111, 109, 109, 121\} = \{1, -4, -6, -6, 6\}, encode = \{0001, 1100, 1110, 1110, 0110\}

5. Putting together bits of header compression in order to result in decompression back:

Header:
The First Bit to bit into – \( n \) = window information
Bit \( n+1 \) to end = bit encoding result

With the following lineup:

```
| Min byte window | Diff / 2 | Size window | Bit compression data |
```

**Figure 1.** Compression Scheme.
Example: From the above compression obtained two window, that window 1 \{123\} and window 2 \{tommy\}. Window 1 consists of three characters with the value min 49 and differential divided 2 = 1. So a bit of window 1 is:
00110001 001 011 11 00 10
Window 2 consists of 5 characters with the value min 109 and differential 12 divided 2 = 6, so that bit of window 2 is: 01101101 110 101 0001 1100 1110 0110
6. So, the compression results are obtained as follows:
00110001001101110010011011101101000111001110011011001101100110 = 54 bit
The number of initial bits 64-bit ratios obtained = 15.62 %

3. Performance
An experiment done with the platform specification intel core i5 2.5 GHz CPU, 8 Gb of RAM and is using an operating system WIN 10 64-bit by using a different file type. In Figure 2 is the display of application testing on the file extension .bmp, while Figure 3 with type a file: doc, the results of the test methods proposed for compression of files is shown in table 1.

| File Name | Extension | Original Size (kb) | Compressed Size (kb) | Comp. Ratio (%) |
|-----------|-----------|--------------------|----------------------|-----------------|
| File 1    | .bmp      | 260.12             | 225.7                | 13.23%          |
| File 2    | .html     | 45.56              | 45.04                | 1.14%           |
| File 3    | .doc      | 125.95             | 92.6                 | 26.48%          |
| File 4    | .doc      | 780.67             | 534.78               | 31.50%          |
| File 5    | .png      | 435.56             | 334.56               | 23.19%          |
| File 6    | .jpg      | 231.59             | 211.45               | 8.70%           |
| File 7    | .doc      | 867.23             | 672.34               | 22.47%          |
| File 8    | .html     | 49.82              | 48.79                | 2.07%           |
| File 9    | .bmp      | 987.34             | 879.34               | 10.94%          |
| File ...  | .xls      | 234.21             | 225.45               | 3.74%           |
Testing has been done gives varying compression ratio results were in a file with a uniform byte locally can give quite a good compression ratio. As for the file or files with difference bytes large enough will give you a small ratio or even give a considerable compression results given the existence of additional information for byte – byte which is not compressed and header information for the purposes of decompression. In table 1 is the partial data for test results of compression with a different file type, from the results of test the proposed method produce good compression rate of 14%, but for file with extension .doc type has higher compression lower than in the file extension .html, in Figure 4 and Figure 5 is the display of the results of compression in the form of graphs.

Based on the test results statistics, it can be seen that the development differential ASCII can be applied not only on files with text characters but also can be applied to digital files in general although the necessary existence of a development further away from the side of efficiency header information that many spend space compression results.
4. Conclusion
In this paper has discussed a simple compression scheme utilizing difference value from ASCII symbols to produce a simple bit value. Compression ratio of the scheme proposed in this paper is good enough, but relies on a variant of the ASCII symbols of files or messages. There is a weakness in the delta encoding that requires a constant variant symbol or stable so it can be retrieved the optimal compression ratio. The application of window size parameters and limit values for the differential can be adjusted so as to give the different compression results, so more research is needed to be able to see the influence of the use of window size and limit differential value are different.

References
[1] Canovas, R., & Moffat, A. (2013). Practical Compression for Multi-Alignment Genomic Files. Proceedings or the Thirty-Sixth Australasian Computer Science Conference.
[2] Chakraborty, D., Ghosh, D., & Ganguly, P. (2015). A Dictionary based Efficient Text Compression Technique using Replacement Strategy. International Journal of Computer Applications (0975 – 8887).
[3] Sharma, M. (2010). Compression Using Huffman Coding. IJCSNS International Journal of Computer Science and Network Security, VOL.10 No.5.
[4] Schlosberg, A., Lam, B. Y., Yeo, G. S., & Clifton, R. J. (2013). DEFLATE Compression Algorithm Corrects for Overestimation of Phylogenetic Diversity by Grantham Approach to Single-Nucleotide Polymorphism Classification. International Journal of Molecular Sciences, 15, 8491-8508.
[5] Zu, Y., & Hua, B. (2015). Parallelizing the Deflate Compression Algorithm on GPU. Journal of Computational Information Systems 11:17.
[6] Sulochana, V., & Verma. (2012). Design and Implementation of LZW Data Compression Algorithm. International Journal of Information Sciences and Techniques (IJIST) Vol.2, No.4.
[7] Sidhu, A. S., & Garg, E. M. (2014). An Advanced Text Encryption & Compression System Based on ASCII Values & Arithmetic Encoding to Improve Data Security. International Journal of Computer Science and Mobile Computing, Vol. 3, Issue 10.
[8] Singh, U., & Garg, U. (2013). An ASCII Value Based Text Data Encryption System. International Journal of Scientific and Research Publications, Vol. 3, Issue 11.
[9] Klein, S. T., & Shapira, D. (2007). Compressed Delta Encoding for LZSS Encoded Files. Data Compression Conference (DCC ’07). ISSN : 1068-0314.
[10] Mogul, J. C., Douglas, F., Feldmann, A., & Krishnamurthy, B. (1997). Potential benefits of delta encoding and data compression for HTTP (Corrected version). WRL Research Report 97/4a. Western Research Laboratory.
[11] Klein, S. T., Serebro, T. C., & Shapira, D. (2006). Modeling Delta Encoding of Compressed Files. Data Compression Conference.
[12] Suel, T., & Memon, N. (2002). Algorithms for Delta Compression and Remote File Synchronization. Academic Press.
[13] Psounis, K. (2002). Class-based Delta-encoding: A Scalable Scheme for Caching Dynamic Web Content. Proceedings of the 22 nd International Conference on Distributed Computing Systems Workshops (ICDCSW’02).
[14] Mogul, J., Krishnamurthy, B., Douglas, F., Feldmann, A., Goland, Y., van Hoff, A., et al. (2002, January). Delta encoding in HTTP. Network Working Group.
[15] Tommy. (2016). Pemanfaatan Diferensiasi ASCII Pada Implementasi Kompresi Teks. Jurnal Logika, Vol. IX, No. 18, ISSN : 2087-3018.

Acknowledgments
This research was supported by Universitas Prima Indonesia and Universitas Harapan Medan, Indonesia.