Handling

by Budi Irmawati

Submission date: 09-Apr-2023 02:12AM (UTC-0500)
Submission ID: 2059415510
File name: 53-75-1-PB.pdf (633.35K)
Word count: 2571
Character count: 12463

Proceedings International Conference on Science and Technology (ICST)

e-ISSN: 2722-7375 pp: 233-239, Vol. 1, Juni 2020

Multiple Character Modification for Huffman Algorithm

Wilham Reyssan, Budi Irmawati*, Fitri Bimantoro

Department of Informatics Engineering, University of Mataram Lombok NTB, INDONESIA

wilhamr.reyssan@gmail.com, [budi-i, bimo]@unram.ac.id

Abstract. Data compression is a method to reduce a file size. The reduction will also speed up the data transmission between devices. Huffman algorithm is one of the lossless data compressions methods, which calculates characters' occurrence as the reference to convert the characters to the related bit strings. The more frequent the occurrence, the shorter the converted bit string obtained. In a text file, a sequence of characters (or letter) may occur frequently. These sequences consist of twin characters or sequences of vowels and consonants (eg. 'aa', 'gg', 'ny', and 'kan'), as a part of a word. If a sequence is replaced with a new symbol outside the alphabet, the vocabulary will increase. However, if the symbols frequently appear in the text, they will be coded to a shorted bit string that we believe will increase the Huffman text ratio compression.

We did experiments by converting some sequences of characters in a text file to new symbols. To find the recommended sequences, we only converted the sequences that their probability was higher than an observed threshold. Therefore, we observed several thresholds to find which sequences that potentially converted to symbols. Our test data consisted of three text files: one raw text file and two files of doc extension. To verify whether conversion of some sequence characters improved the Huffman ratio compression, the Huffman compression was implemented on the original test data and on the converted test data. The experimental results showed that the threshold of 1% is the best. Evaluation on the original test data resulted the Huffman ratio compression of 44.88% and 88.84% for the raw text file and for the doc extension file respectively. This experiment proofs that the conversion of some sequence characters to symbols benefits to the Huffman Algorithm by obtaining 45.93% and 89.09% for the raw text file and for the doc extension file respectively.

Keywords. Data Compression, Huffman Algorithm

1. Introduction

Large sized of data will require a large sized of memory allocation, which also increase the transmission cost. Compression method is an alternative to reduce the size of data. Huffman is an algorithm that gets many attentions in the data compression field because of its lossless characteristic. This type of compression returns a compressed file to its original file without any changes.

On the other hand, a document may have some sequence letter that always come together as a combination of the same characters (consonants or vowels or their combinations) such as 'aa', 'gg', 'ny', or 'ng'. The appearance of these sequences went to a question whether a conversion of a highly frequent sequence to a new character outside the original vocabulary may increase the compression ratio.

2. Literature Review and Basic Theory

2.1. Literature Review

From the literature, an experiment using Huffman compression technique reduced the number of characters from 33 to 21 and got compression ratio of 63.67% [2]. The Huffman code research for message compression occupied 85 bits storage, much smaller than the one used by ASCII coding of

*Corresponding author

216 bits out of total 27 characters. Astuti and Hidayat showed that the percentage of a compression to string ratio tested was 39.35% whether the Huffman ratio compression was more than 60% [3].

In a text compression using Huffman and MD5 algorithms on Android instant messaging, the number of text files to be compressed using Huffman was nine files with the data size in the range of 16-23.11 bit. It showed that the compression ratio was up to 75% in the file with size of 16 bits and 96 bits where each character has occurrence frequency of almost the same. The text compression ratios were not significantly improved when the text size was too small because the characters that occurred together were rare. The percentage of the smallest compression ratio was 34.3% [4].

A multimedia data hiding research used the end of file (EOF) and Huffman code method. The method was tested on four types of media files: text, image, audio, and video. When it was evaluated on five test data, which ranged 3-38 Kb, its average compression ratio was 55.07%. When it was tested on five images test data with different extensions, the average compression was 6.07%. It resulted 4.79% and 4.04% in average when it was tested on five audio data and video data respectively [5].

A comparison of text data compression using *Huffman*, *Shannon-Fano*, *Run Length Encoding*, and *Tunstall* methods, with the largest size of 12 bytes (or equivalent to 96 bits), showed that the compression ratios were 81.25%, 81.25%, 58.17%, and 79.17% respectively [6]. In the comparison of the Huffman and Shannon-Fano algorithms, it used a text file of size 357 bytes that consisted of 27 characters. It obtained 262 and 260 bytes with compression ratios of 73.59% and 73.03% respectively [7].

A comparative analysis of data compression with *Fixed-Length Code*, *Variable-Length Code*, and *Huffman Algorithm* used two strings of test data with each occurrence frequency was 31 and 28. Using the *Fixed-Length Code* technique, the experiment obtained 50% and 62.5% compression ratio. Using the *Variable-Length Code* technique, the compression ratio was 25% and 72%. For the test using the *Huffman algorithm* the compression ratio was 53% and 73% [8]. The experiment shows that the Huffman Algorithm outperformed the two previous methods.

The Region Based Huffman (RBH) Compression Technique with *Code Interchange* study in modifying the Huffman compression technique used RSA algorithm looked for a proper *N* values from various *N* values. The compression ratio of two raw files increased to 28.71% and 31.41% from ones without modification of 26.84% and 30.31% respectively. However, the compression ratio for two doc files increased only about 0.03% and 0.04% [9].

3. Basic Theory

3.1 Data Compression

Data compression is a process of reducing the size of data to save spaces and transfer time. Data compression is divided into two types namely lossless and lossy data compression [10].

3.2 Huffman Algorithm

Huffman code is a coding algorithm for lossless data compression. This term refers to the use of code tables that have various lengths, wherein the code table is derived in a certain way based on the occurrence probability of each value in the data source [11]. In the Huffman Code, characters that occur more often are converted to shorter bit strings than ones rarely occur in a data source. Huffman compression is the most efficient method of other similar methods because it maps each symbol from the data source into a unique string to produce a smaller output file. The method for encoding symbols or characters is used with the help of binary trees by combining the two smallest frequencies of characters to form a coded tree.

4 Research Method

4.1 System Process

This experiment had two steps: compression (shrinkage of data) and decompression (transfer back the data to their original form). Both processes consisted of several steps described in Figures 1 and 2. The multi-character modification is our proposal. It converts multi characters to a single new character outside the alphabet. The conversion is listed in Table 1.

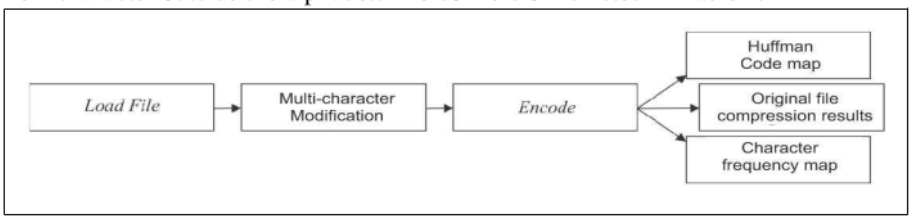

Fig. 1. The pipeline of the compression process

TABLE. 1. MODIFICATION OF SYMBOL

No	Multi characters	Introduced character	No	Multi characters	Introduced character
1	aa	Ā	16	pp	Þ
2	bb	ß	17	qq	Ō
3	cc	Ç	18	rr	Ō
4	dd	Đ	19	ss	Ō
5	ee	Ē	20	tt	Ō
6	ff	Ē	21	uu	Ū
7	gg	Đ	22	vv	Ū
8	hh	Æ	23	ww	Ū
9	ii	Ī	24	xx	Ū
10	jj	Ī	25	yy	Ȳ
11	kk	Ī	26	zz	Ā
12	ll	Ī	27	ng	ŋ
13	mm	æ	28	ny	ŝ

14	nn	N	29	sy	°
15	oo	O	30	kh	¿

The *Encode process* in Figure 2 followed the process in the original Huffman algorithm. Table 2 lists the occurrence and the probability of some characters. According to Table 2, Đ:1 and n:1are selected as the first node in the Huffman tree. Figure 3 shows the final Huffman tree according to the data in Table 2. The Huffman code is read from the root node to the leaf node by observing the label value on each edge as shown in Table 3.

TABLE 2. THE FREQUENCY AND THE PROBABILITY OF THE SAMPLED CHARACTERS FROM TABLE 1.

Character	Frequency	Probability
Đ	1	1/6
t	1	1/6
a	2	2/6
n	1	1/6

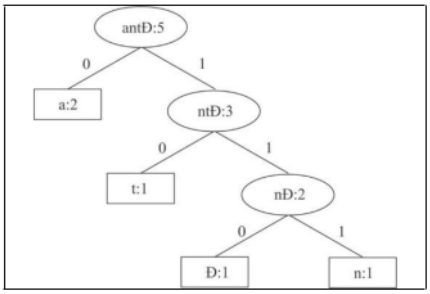

Fig. 3. The final result of the Huffman tree

TABLE 3 THE PROPOSED HUFFMAN MODIFICATION ALGORITHM RESULTS.

Character	*Huffman code*
Đ	110
A	0
N	111
T	10

In the final step, the compression results consist of three files:

4.1.1 Original Compression File, is the result of original Huffman code. It is created for easy comparison with the compression given by our proposed method.

4.1.2 Huffman Code Map file, is a map file resulted by the encoding process.

4.1.3 Character Map file, is the character map file.

4.2 Evaluation Scenario

To examine the performance of our proposed Huffman modification algorithm, we did some compression ratio testing. Compression ratio is the level of data reduction achieved as a result of compression follows Equation (1). The evaluation was done using several raw text files and doc files with different length. To find out the optimum compression, we used different numbers of multi characters minimum frequency. Then, we looked for the minimum frequency that resulted the highest compression for each file type.

$$P = \left(\frac{original\ file\ size - compressed\ file\ size}{original\ file\ size}\right) x\ 100\% \qquad (1)$$

5. Result and Discussions

Based on the results of data test in Table 4 we used several thresholds as a parameter in converting multi characters according to the new symbols in Table 1. We carried out five texts as the test data from two different file extensions, at the minimum probability of 1% threshold as the highest result of compression. We obtained the average percentage of the compression ratio of Huffman modification of 45.93% for raw files, higher than the average percentage ratio of the original Huffman compression ratio without modification which is 44.88%. For doc files with also minimum 1% threshold as the highest compression result. The average percentage of the modified Huffman compression ratio is 89.10%, higher than the average percentage ratio of the original Huffman compression ratio without modification that is 88.83%. These results indicate the higher the threshold used as a parameter to convert Huffman modifications, the higher the percentage of compression ratio produced. This can be interpreted, the higher the probability of multi characters appearing, the better it is to convert to another symbol with a modification of Huffman.

TABLE I. EXPERIMENTAL RESULTS OF SEVERAL TEST FILE WITH SEVERAL THRESHOLD

Text file	*Original text size (bytes)*	*Proposed Huffman Modification*				*Original Huffman Method*		
		Threshold	*compression (bytes)*	*Percentage (%)*	*decompression (bytes)*	*compression (bytes)*	*Percentage (%)*	*decompression (bytes)*
novel.txt	35770	T = 0%	19,333	45.95	35,770	19,678	44.99	35,770
		T = 0,01%	19,333	45.95	35,770			
		T = 0,1%	19,331	45.96	35,770			
		T = 1%	19,300	46.04	35,770			
sejarah.txt	31424	T = 0%	17,390	44.66	31,424	17,631	43.89	31,424
		T = 0,01%	17,390	44.66	31,424			
		T = 0,1%	17,388	44.67	31,424			
		T = 1%	17,376	44.70	31,424			
cerita_raky at.txt	29591	T = 0%	15,704	46.93	29,591	16,054	45.75	29,591
		T = 0,01%	15,704	46.93	29,591			
		T = 0,1%	15,703	46.93	29,591			
		T = 1%	15,670	47.04	29,591			
fabel_kancil.doc	38400	T = 0%	4,660	87.86	8,684	4,751	87.63	8,684
		T = 0,01%	4,660	87.86	8,684			
		T = 0,1%	4,660	87.86	8,684			
		T = 1%	4,649	87.89	8,684			
hikayat_buah_kemuning.doc	34304	T = 0%	3,330	90.29	6,387	3,416	90.04	6,387
		T = 0,01%	3,330	90.29	6,387			
		T = 0,1%	3,330	90.29	6,387			
		T = 1%	3,326	90.30	6,387			

6. Conclusion

Based on the testing and analysis of the system that has been done, the following conclusions can be drawn:

1. Minimum probability testing with a threshold of 1% obtained the smallest compression output with the highest percentage of compression
2. The higher the threshold used as a parameter to convert Huffman modifications, the higher the percentage of compression ratio produced.
3. The higher the probability of multiple characters appearing, the better it is to convert to another symbol with Huffman's modification.

References

[1] H. Sinaga, P. Sihombing, and H. Handrizal, "Perbandingan Algoritma Huffman Dan Run Length Encoding Untuk Kompresi File Audio," *Talent. Conf. Ser. Sci. Technol.*, vol. 1, no. 1, pp. 010–015, 2018.

[2] A. Putera and U. Siahaan, "JURNAL INFORMATIKA Vol. 10, No. 2, Jul 2016 IMPLEMENTASI TEKNIK KOMPRESI TEKS HUFFMAN," *J. Inform.*, vol. 10, no. 2, pp. 1251–1261, 2016.

[3] E. Z. Astuti and E. Y. Hidayat, "Kode Huffman untuk Kompresi Pesan," *Techno.COM*, vol. 12, no. 2, pp. 117–126, 2013.

[4] M. T. Chulkamdi, S. H. Pramono, and E. Yudaningtyas, "Kompresi Teks Menggunakan Algoritma Huffman dan Md5 pada Instant Messaging Smartphone Android," *J. EECCIS*, vol. 9, no. 1, pp. 103–108, 2015.

[5] Y. R. Nasution, A. Johar, and F. F. Coastera, "Aplikasi Penyembunyian Multimedia," *J. Rekursif*, vol. 5, no. 1, pp. 86–106, 2017.

[6] D. A. Rachesti, "Comparison of Text Data Compression Using Huffman , Shannon-Fano , Run Length Encoding , and Tunstall Methods," *Int.*

[7] *J. Appl. Eng. Res.*, vol. 12, no. 23, pp. 13618–13622, 2017.

[8] A. M. Pratama, N. A. Hasibuan, and E. Buulolo, "Penerapan algoritma huffman dan shannon-fano dalam pemampatan file teks," *Inf. dan Teknol. Ilm.*, vol. 12, no. September, pp. 312–317, 2017.

[9] J. Jamaluddin and U. M. Indonesia, "Analisis Perbandingan Kompresi Data dengan Fixed-Length Code , Variable- Length Code dan Algoritma Huffman Analisis Perbandingan Kompresi Data dengan Fixed-Length Code , Variable-Length Code dan Algoritma Huffman," *Maj. Ilm. Methoda*, vol. 3, no. 2, pp. 41–47, 2018.

[10] U. Nandi, J. K. Mandal, W. Bengal, and W. Bengal, "REGION BASED HUFFMAN (RBH) COMPRESSION TECHNIQUE WITH," *Malaysian J. Comput. Sci.*, vol. 23, no. 2, pp. 111–120, 2010.

[11] T. Y. Septianto *et al.*, "Pemampatan Tata Teks Berbahasa Indonesia Panjang Simbol Bervariasi," *J.*

Mhs. TEUB, vol. 3, no. 1, 2015.

[12]D. A. Yansyah and I. Pendahuluan, "Perbandingan Metode Punctured Elias Code," *J. Ris. Komput.*, vol. 2, no. 6, pp. 33–36, 2015.

Handling

ORIGINALITY REPORT

8%	5%	5%	5%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to Universitas Mataram Student Paper	3%
2	Kuo-Kun Tseng, Jun Min Jiang, Jeng-Shyang Pan, Ling Ling Tang, Chih-Yu Hsu, Chih-Cheng Chen. "Enhanced Huffman Coding with Encryption for Wireless Data Broadcasting System", 2012 International Symposium on Computer, Consumer and Control, 2012 Publication	1%
3	www.ripublication.com Internet Source	1%
4	www.ijcaonline.org Internet Source	1%
5	e-journal.uajy.ac.id Internet Source	<1%
6	Anatoly M. Strel'chuk, Viktor V. Zelenin, Alexei N. Kuznetsov, Joseph Tringe, Albert V. Davydov, Alexander A. Lebedev. "Anomalous Scatter of Forward Current-Voltage Characteristics of He^{+}-Irradiated Ni/4H-SiC	<1%

Schottky Diodes", Materials Science Forum, 2016
Publication

7	N. Khellaf, K. Kebiche. "Geometric and Material Nonlinear Analysis of Square-Based Tensegrity Ring Structures", Arabian Journal for Science and Engineering, 2014 Publication	<1 %
8	vdoc.pub Internet Source	<1 %
9	eprints.unram.ac.id Internet Source	<1 %

Exclude quotes	On	Exclude matches	< 3 words
Exclude bibliography	On		