

Short Paper

Faster Implementation of Canonical Minimum Redundancy Prefix Codes^{*}

KUO-LIANG CHUNG AND JUNG-GEN WU⁺

*Department of Information Management
and Institute of Information Engineering
National Taiwan University of Science and Technology
Taipei, Taiwan 106, R.O.C.
E-mail: klchung@cs.ntust.edu.tw*

⁺*Department of Information and Computer Education
National Taiwan Normal University
Taipei, Taiwan 106, R.O.C.*

This note modifies the data structure developed by Moffat and Turpin [3]. Besides reducing the memory requirement, the decoding algorithm is further improved. Experimental results demonstrate both the memory and decoding-time advantages of the modified method.

Keywords: Huffman decoding, compression, algorithm, data structure, canonical Huffman tree

1. INTRODUCTION

Huffman coding [2] has been widely used in data, image, and video compression. Based on the single-side growing Huffman tree (SHT), Moffat and Turpin [3] recently presented an efficient data structure for representing the SHT. It requires $(n + 2d)w$ bits, where n denotes the number of source symbols; d denotes the depth of the SHT; and w bits are required to save one source symbol or one integer. Their decoding method, called canonical minimum redundancy prefix codes (CMRPC), takes $O(\log d)$ time. In addition, the one-shift method and table-lookup method were employed to reduce the decoding time, but the memory requirement increased. Experimental results reveal that both the memory requirement and the decoding time in their three implementations are superior to the results obtained by Hashemian [1].

Suppose the topmost $d' + 1$ levels of the SHT form a complete subtree. This note modifies the CMRPC method [3]. As a result, the memory requirement can be reduced to $(n + 2d - 2d')w$ bits, and the decoding time can be reduced to $O(d - d')$. Experimental results reveal that the memory requirement can be improved by about 1.1% to 4.1%; the

Received August 29, 1998; revised September 30 & December 2, 1999; accepted January 18, 2000.

Communicated by Yung-Nien Sun.

^{*}This research was supported by NSC88-2213-E011-006.

decoding time can be improved by from approximately 16.2% to 25.5%, which is superior to the one-shift method and table-lookup method.

2. THE MODIFIED CMRPC

Given a set of symbols $s_1, s_2, \dots,$ and s_{10} with frequencies 1, 2, ..., and 10, respectively, the constructed SHT is that shown in Fig. 1.

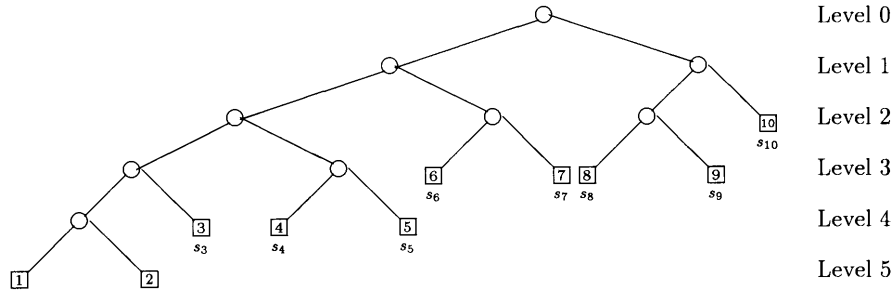


Fig. 1. The SHT.

Moffat and Turpin [3] use three tables for representing the SHT. The symbol table (see Fig. 2(a)), *symbol* [0..9], stores the source symbols in the SHT in a left-to-right and bottom-up manner. The offset table stores the index of the first codeword among those of the same length (w.r.t. the SHT) in the symbol table. For example, the index of the first codeword with length 5 is 0. The corresponding offset table is shown in Fig. 2(b). The offset of codewords with length is 10, which is greater than any index in the table. This means that there is no codeword with length 1. The base table stores the bases of codewords for all the codeword lengths. The base of a set of codewords is the smallest value of the codewords. For example, the set of codewords with length 3 has four elements, i.e., $010_2, 011_2, 100_2$ and 101_2 , and the smallest value is 2 (010_2); the set of codewords with length 4 has three elements, i.e., $0001_2, 0010_2,$ and 0011_2 , and the smallest value is 1 (0001_2). The base of the codewords with length is calculated as $base[l] = \left\lceil \frac{\sum_{k=l+1}^d m_k 2^{d-k}}{2^{d-l}} \right\rceil$, where m_k is the number of codewords of length k . Return to Fig. 1.

We have $base[5] = 0, base[4] = 1, base[3] = 2, base[2] = 3,$ and $base[1] = 2,$ as shown in Fig. 2(c).

0	1	2	3	4	5	6	7	8
s_1	s_2	s_2	s_3	s_4	s_5	s_6	s_7	s_8

(a) The symbol table.

Length	1	2	3	4	5
	10	9	5	2	0

(b) The offset table.

Length	1	2	3	4	5
	2	3	2	1	0

(c) The base table.

Fig. 2. The original data structure.

Suppose the topmost $d' + 1$ levels of the SHT form a complete subtree. In Fig. 1, $d' = 1$. We can discard the first d' entries in the offset table and the base table. Thus, the modified offset table, $moffset[1..4]$, and the modified base table, $mbase[1..4]$ (see Fig. 3(a) and Fig. 3(b)) are obtained. The modified data structure need $(n + 2d - 2d')w$ bits.

Length	2	3	4	5
	9	5	2	0

(a) The modified offset table.

Length	2	3	4	5
	3	2	1	0

(b) The modified base table.

Fig. 3. The modified data structure.

Based on the modified data structure, the decoding algorithm is listed below:

1. $code \leftarrow$ read the first $d' + 1$ bits from the input and $length \leftarrow d' + 1$.
2. While $code < mbase[length-d']$ do
 - (a) shift $code$ one bit position to the left and add the next bit from the input to the rightmost bit of $code$.
 - (b) $length \leftarrow length + 1$.
3. $output \leftarrow symbol[moffset[length-d'] + (code - mbase[length-d'])]$.

3. EXPERIMENTAL RESULTS

Four real images, namely Lena, Mandrill, F16, and Peppers, were used to evaluate the performance. Each image had 512×512 pixels, and each pixel has 256 greyscales. That is, eight bits ($w = 8$) were used to represent the greyscale of each pixel (source symbol). The four images were directly Huffman coded without any pre-processing. The depths of the corresponding four images were 18, 18, 17, and 17; the number of topmost levels ($d' + 1$) that formed a complete subtree for each of the four images was 7, 7, 4, and 5, respectively. The machine used was an IBM compatible personal computer a 133 MHz Pentium microprocessor and the Windows 95 operating system.

Table 1 and Table 2 show the memory and decoding time performance, where: MMRPC denotes the modified CMRPC method; OMRPC denotes the one-shift method [3], where at each time, the code with the longest codeword length in the SHT was processed; TMRPC denotes the table-lookup method [3], where the variable x (see [3]) was set to 8.

The experimental results show that the improvement both in memory utilization and in decoding time was proportional to d' . Hence, for pre-processed data, such as images compressed using the JPEG method, which have very small d' (1 or 2), the performance improvement will be minor. However, for ordinary data, the experimental results reveal that the memory utilization for CMRPC can be improved from 1.1% ($= (2168 - 2144)$ bits/2168 bits) to 4.1% ($= (2168 - 2068)$ bits/2168 bits); the decoding time can be improved from approximately 16.2% ($= (1.91 - 1.60)$ sec/1.91 sec) to 25.5% ($= (2.12 - 1.58)$ sec/2.12 sec). It is also observed that the proposed modified method is superior to the one-shift method and the table-lookup method [3].

Table 1. Memory requirement (in bits) comparison among CMRPC, OMRPC, TMRPC, and MMRPC.

image	CMRPC	OMRPC	TMRPC	MMRPC
Lena	2168 1	2348 108%	4216 194%	2080 95.9%
Mandrill	2160 1	2340 108%	4208 195%	2072 95.9%
F16	2144 1	2297 107%	4192 196%	2104 98.1%
Peppers	2168 1	2321 107%	4216 194%	2144 98.9%

Table 2. Decoding time (in seconds) comparison among CMRPC, OMRPC, TMRPC, and MMRPC.

image	CMRPC	OMRPC	TMRPC	MMRPC
Lena	2.09 1	1.69 80.9%	1.66 79.4%	1.59 76.1%
Mandrill	2.12 1	1.76 83%	1.67 78.8%	1.58 74.5%
F16	1.91 1	1.68 88%	1.65 86.4%	1.60 83.8%
Peppers	2.10 1	1.71 81.4%	1.70 81%	1.67 79.5%

ACKNOWLEDGMENT

The authors are indebted to the three reviewers, Prof. W. H. Tsai, and Prof. W. L. Hsu for their valuable suggestions and corrections that led to the improved version of the paper.

REFERENCES

1. R. Hashemian, "Memory efficient and high-speed search Huffman coding," *IEEE Transaction on Communications*, Vol. 43, No. 10, 1995, pp. 2576-2581.
2. D. A. Huffman, "A method for the construction of minimum redundancy codes," in *Proceedings of IRE*, Vol. 40, 1952, pp. 1098-1101.
3. A. Moffat and A. Turpin, "On the implementation of minimum redundancy prefix codes," *IEEE Transaction on Communications*, Vol. 45, No. 10, 1997, pp. 1200-1207.

Kuo-Liang Chung (鍾國亮) received the B.S., M.S., and Ph.D. degrees in Computer Science and Information Engineering from National Taiwan University, R.O.C. since 1995, he has been a professor in the Department of Information Management at the National Taiwan Institute of Technology. From 1984 to 1986, he fulfilled his military obligation. His current research interests include image processing, computer graphics, applied computational geometry, and data compression. He is a member of ACM, IAPR, and IEEE.

Jung-Gen Wu (吳榮根) received the B.S., M.S., and Ph.D. degrees in Electrical Engineering from National Taiwan University, R.O.C. He is now a professor in the Department of Computer and Information Education at National Taiwan Normal University. His current research interests include image processing, spatial data structures, computer architecture, and parallel processing. He is a member of IEEE and the Computer Society.