GENERIC CONSTRUCTION ALGORITHMS FOR SYMMETRIC AND ASYMMETRIC RVLCS

Chia-Wei Lin, Yuh-Jue Chuang, Ja-Ling Wu, Senior Member IEEE

Communication and Multimedia Laboratory

Department of Computer Science and Information Engineering

National Taiwan University, Taipei, Taiwan, R.O.C.

E-mail:{ cwlin, chuangyj, wjl }@cmlab.csie.ntu.edu.tw

Abstract - In this paper, we propose several generic algorithms that can construct symmetric and asymmetric reversible variable length codes (RVLCs). RVLCs, adopted in emerging video coding standards such as MPEG-4 and H.263+, can be used to enhance the corresponding error resilience capability in the presence of transmission bit errors. Experimental results show that the proposed symmetric RVLC algorithm can produce codes with shorter average codeword lengths and shorter maximum codeword lengths. And the proposed asymmetric RVLC algorithms will generate codes with either shorter average codeword lengths or shorter maximum codeword lengths, as compared to the existing approaches.

Keywords - Reversible Variable Length Codes (RVLC), Huffman Code, MPEG-4, JPEG, H.263

I. INTRODUCTION

Variable length codes (VLCs) have been used as entropy coding in almost all image and video coding standards (such as JPEG [1], H.261 [2], H.263 [3], MPEG-1 [4], MPEG-2 [5] and MPEG-4 [6]). VLCs are known by their ability of achieving high compression efficiency; however, they are very sensitive to errors due to their variable codeword length nature. Even one single bit error will induce the problem of error propagations, such that the data received after the bit error position become useless and result in a serious problem for any VLC-involved applications. RVLCs, which can be decoded in either the forward or backward direction, are developed in order to lessen the effect of error propagation. The MPEG-4 video standard includes an optional RVLC that can be used for quantizing the DCT coefficients. In H.263+ Annex D and H.263++ Annex V, RVLCs are used to encode motion-vector data and header data, respectively.

There are two types of RVLCs, one is the symmetric RVLC and the other is the asymmetric one. The symmetric RVLC shares the same codeword table when decoding in both the forward and backward directions, because the codewords are symmetric. On the other hand, two codeword tables are necessary for decoding the asymmetric RVLC. Thus, the memory requirement of the symmetric RVLC is less than that of the asymmetric RVLC. However, the asymmetric RVLC always provides better efficiency (in terms of average codeword length) than the symmetric one, because the codeword selection can be more flexible.

RVLCs have been extensively studied in recent decades. Fraenkel et al. [7] presented necessary conditions for the existence of RVLCs together with an algorithm to construct a complete RVLC for a given set of codeword lengths. Takishima et al. [8] proposed an algorithm (for convenience it is called the Takishima's algorithm in short) for constructing symmetric and asymmetric RVLCs from a given Huffman code [9]. Tsai and Wu [10, 11] proposed a more efficient symmetric RVLC construction algorithm based on Takishima's algorithm. Tsai and Wu's algorithm differs from Takishima's algorithm in the codeword selection mechanism. Wen and Villasenor [12] presented an RVLC construction method that is used to obtain reversible codes with the same length distribution as the Golomb-Rice codes and exp-Golomb codes.

Recently, Tseng and Chang [13] presented a backtracking based algorithm that can construct symmetric RVLCs, effectively. The experimental results show that their algorithm can generate even better codes than those of previous methods. Tseng and Chang's algorithm provides shorter average codeword length than Tsai and Wu's algorithm on the Canterbury Corpus file set (will be addressed later), with the exception of the file "kennedy.xls". This inspires us to develop a new algorithm to generate more efficient RVLCs.

In the following, we will present one method for constructing the symmetric RVLC and two methods for asymmetric RVLC. And then some comparisons between the existing RVLC construction algorithms and the proposed ones are made. This paper is organized as follows. In Section 2, related works and details of proposed algorithms are addressed. Experimental results are shown in Section 3. Our conclusions are drawn in Section 4.

II. THE PROPOSED GENERIC ALGORITHMS FOR CONSTRUCTIING RVLCS

Since the proposed RVLC construction algorithms are

0-7803-7510-6/02/$17.00 ©2002 IEEE

derived and extended basing on *Tseng and Chang's Symmetric RVLC Algorithm*, we first review their symmetric RVLC construction approach, briefly.

Assume the source data has n symbols and they are represented by $(a_1, a_2, \ldots, a_n)$ in decreasing order of probability $P(a_i)$, where $P(a_i)$ is the probability of the data symbol a_i. The desired RVLC can be expressed as a codeword list with n-tuple $(C_1, C_2, \ldots, C_n)$, where C_i is the codeword of symbol a_i. Tseng and Chang's algorithm tries to minimize a bounding function (i.e. the average codeword length) $B(C_1, C_2, \ldots, C_n)$, which is defined as

$$B(C_1, C_2, \ldots, C_n) = \sum_{i=1}^{n} P(a_i) L(C_i) \qquad (1)$$

where $L(C_i)$ is the length of codeword C_i.

Definition 1— Symmetrical Children: In a full binary tree, the symmetrical children of node X are defined by all of the first symmetrical codewords on paths from mode X to leaf nodes.

Figure 1 shows an example, in which black nodes represent symmetric codewords. The symmetrical children of B('1') are D('11'), G('101'), and K('1001'). The symmetrical children of C('00') is E('000').

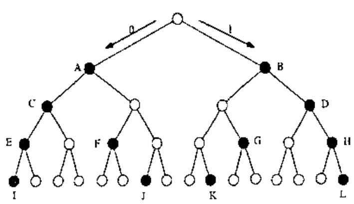

Figure 1. A sample code tree.

If we replace codeword C_i with its symmetrical children, we will obtain a new codeword list. The main idea of Tseng and Chang's algorithm is that if the bounding function of this new list is smaller than that of the original list, then the *replacement* operation is carried out and the target list is replaced with the new list.

According to Tseng and Chang's experimental results, backtracking based algorithm usually provides shorter average codeword length than that of the Tsai and Wu's. However, the not-all-win fact of Tseng and Chang's algorithm, as shown in the last row of Table 1, implies that there is some room for us to design even more efficient RVLC construction algorithms. We suggest another idea different from Tseng and Chang's in the codeword replacement. In their algorithm, when a codeword Ci is to be replaced, it is replaced with its children. (i.e. codewords that have Ci as their prefix) In our proposed idea, when a codeword Ci is to be removed from the list, in addition to Ci's children, we also traverse other parts of the code tree and add valid codewords into the target list.

Before the first algorithm is specified, we explain some related terms in the following.

Let the average codeword length of the target list before Ci is removed be L_{avg}, and let the L'_i be the average codeword length of the target list after Ci is removed.

Definition 2—Remove C_i condition: if $i = \arg\min_j L'_j$ and $L'_i < L_{avg}$ then we remove C_i from our original candidates list $(C_1, C_2, \ldots, C_m)$.

A.Symmetric RVLC Construction algorithms

Algorithm-1 – *The Greedy Symmetric RVLC Algorithm:*

Step1: The target list starts with a single codeword "1".

Step2: For each symmetric codeword at level i (i>1) :
Assign the available symmetric codewords to the target list based on the increasing order of codeword length.

Step3: If there exists a codeword in the target list satisfying the *Remove C_i* condition, defined in definition 2, remove that codeword and proceed with other valid candidates.

Step4: Repeat Step2 and Step3 until there is no codeword satisfying the *Remove C_i* condition.

B.Asymmetric RVLC Construction Algorithms

The proposed symmetric algorithms proceed with other valid candidates instead of replacing with the symmetrical children when a codeword is removed (as was done in Tseng and Chang's approach). This idea can be extended to the asymmetric case directly. A corresponding greedy algorithm for constructing the asymmetric RVLC can be derived.

Unfortunately, empirical results do not provide us good results for compressing the Canterbury Corpus file set, with the exception of the files "ptt5", "sum" and "kennedy.xls". After some investigations, we find that these three files having the following common characteristics:

$$\gamma \triangleq \frac{\max_{a_i \in S} f(a_i)}{\sum_{a_i \in S} f(a_i)} > 0.3 \qquad (2)$$

where $S = \{a_1, a_2, \ldots, a_n\}$ is the alphabet set of the file and $f(a_i)$ represents the counts of appearance of the symbol a_i. Hence, we propose a new algorithm that assigns different initial codewords to the target list based on the ratio, γ, defined in Eqn.(2). This new algorithm can be described as follows:

Algorithm-2 – *The Greedy Asymmetric RVLC Algorithm based on the ratio of the highest frequency of the symbol to the overall frequency of occurrence of the file set*

Step1: The target list starts with a single codeword "1" if $\gamma > 0.3$ or '00' if $\gamma \leq 0.3$ where γ is the ratio of the highest frequency of the symbol to the total frequency of the file set (defined in Eqn.(2)).

Step2: For each asymmetric codeword at level i (i>1, if '1' is selected at Step1; or, i > 2, if '00' is selected at Step1):

Assign the available asymmetrical codewords to the target list based on the increasing order of codeword length.

Step3: If there exists a codeword in the target list satisfying the condition of *Remove C_i* condition, remove that codeword and proceed with other valid candidates.

Step4: Repeat Step2 and Step3 until there is no codeword satisfying the *Remove Ci* condition.

In addition to the two algorithms mentioned above, we have developed another asymmetric RVLC algorithm, which is generalizations of Tseng and Chang's method and the proposed symmetric RVLC algorithms (Algorithm-1) in the asymmetrical case. Before explaining the details of the third algorithm, we introduce some terms used in it, first.

Definition 3— Asymmetrical Children: In a full binary tree, the asymmetrical children of node X in the target list L are defined by all of the first codewords, which are affix-free with respect to the list L, on paths from node X to leaf nodes.

Figure 2 shows an example, in which black nodes represent the codewords in the target list. The asymmetrical children of A('0') are B('00'), C('010'), and D('011').

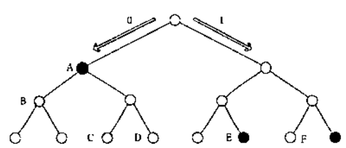

Figure 2. Another sample code tree.

Based on definitions 2 and 3, the second algorithm for constructing the asymmetric RVLC can be depicted as follows.

Algorithm-3 – *The Greedy Asymmetric RVLC Algorithm Based on Replacing Codewords with Their Children:*

Step1: The target list starts with a single codeword "1".

Step2: For each codeword at level i (i>1):

Assign the available codewords to the target list based on the increasing order of codeword length.

Step3: If there exists a codeword in the target list satisfying the *Remove C_i* condition, defined in definition 2, replace this codeword with its asymmetrical children.

Step4: Repeat Step2 and Step3 until there is no codeword satisfying the *Remove C_i* condition.

III. EXPERIMENTAL RESULTS

The proposed algorithms have been tested on the English alphabet set and the file set taken from Canterbury Corpus (available in http://corpus.canterbury.ac.nz/). The Canterbury Corpus file set was developed specifically for testing new compression algorithms. The files were selected based on their ability to provide representative performance results.

The major contribution of our work is that the proposed approaches can be applied to construct both efficient symmetric and asymmetric RVLCs. Furthermore, our algorithms, like Tsai and Wu's approaches [10, 11], can also avoid the codeword variation problem, as compared with Takishima's approach. In other words, the proposed algorithms will generate both unique symmetric and asymmetric RVLCs, for the same given source.

Table 1 respectively shows the maximum codeword lengths and the average codeword lengths of various RVLC algorithms for compressing different source files. It can be seen, from Table 1, that Algorithm-1 produces shorter maximum codeword lengths than Tsai and Wu's. And Algorithm-1 always produces symmetric RVLCs with shorter or equal average codeword length than Tseng and Chang's. The proposed asymmetric algorithm (Algorithm-2) produces shorter maximum asymmetrical codeword lengths than those of the Tsai and Wu's asymmetric approach only in 4 (out of 11) cases and shorter average codeword lengths in 8 (out of 11) cases. Algorithm-3 always produces the shortest maximum codeword lengths among the three asymmetric algorithms.

Table 2 presents the symmetric and asymmetric codewords obtained by applying different construction algorithms to the given English alphabet set, respectively. Moreover, the corresponding average codeword lengths and the maximum codeword lengths are also included. In the symmetrical case, our algorithms provide the same average codeword lengths and the maximum codeword lengths as those obtained by Tseng and Chang's algorithm, but the order of codewords is different. In the asymmetrical case, our algorithms either provide shorter average codeword lengths or shorter maximum codeword lengths than those obtained by Tsai and Wu's approach.

IV. CONCLUSION

Several generic and efficient construction algorithms for symmetric and asymmetric RVLCs are proposed in this paper. The major advantage of the proposed algorithms is their better coding efficiency (in terms of average codeword length). Besides, the proposed algorithms do not have the codeword variation problem.

According to the experimental results, the proposed algorithms can provide either shorter average codeword lengths or shorter maximum codeword lengths. Under certain circumstances, such as limited hardware capacity [14], the maximum codeword length often plays a more important role than the average codeword length. There is still a broad range of strategies available in deciding which codewords to be removed, and further improvement of average codeword length or maximum codeword length may be possible. We will continue investigating into the design of more efficient RVLC construction algorithms.

ACKNOWLEDGEMENT

The authors would like to thank Prof. Chien-Wu Tsai who brought reference [13] to our attention. This project is sponsored by the Program for Promoting Academic Excellence of Universities, Ministry of Education, R.O.C., under the grant number 89E-FA06-2-4-8.

REFERENCES

[1]. ISO/IEC, "Digital Compression and Coding of Continuous-Tone Still Images," International Standard 10918-1, 1991.

[2]. ITU-T, "Video Codec for Audiovisual Services at p*64 kbits/sec," Recommendation H. 261, 1993.

[3]. ITU-T, "Video Coding for Low Bit Rate Communication," Recommendation H. 263, 1997.

[4]. ISO/IEC, "Coding of Moving Pictures and Associated Audio for Digital Storage Media at up to about 1.5 Mbits/s," International Standard 11172, 1991.

[5]. ISO/IEC, "Information Technology - Generic Coding of Moving Pictures and Associated Audio," Draft International Standard 13818, 1994.

[6]. ISO/IEC, "Coding of Audio-Visual Objects: Visual," Final Draft International Standard 14496-2, 1998.

[7]. A. S. Fraenkel and S. T. Klein, "Bidirectional Huffman coding," *Computer Journal*, vol. 33, no. 4, 1990.

[8]. Y. Takishima, M. Wada, and H. Murakami, "Reversible Variable Length Codes," *IEEE Transactions on Communications*, vol. 43, pp. 158–162, 1995.

[9]. D. A. Huffman, "A Method for the Construction of Minimum Redundancy Codes," *Proceeding of the IRE 40*, pp. 1098-1101, 1952.

[10]. C. -W. Tsai and J. -L. Wu "Modified Symmetrical Reversible Variable-Length Code and Its Theoretical Bounds," *IEEE Transactions on Information Theory*, vol. 47, pp. 2543-2548, 2001.

[11]. C. -W. Tsai and J. -L. Wu "On Constructing the Huffman-Code-Based Reversible Variable-Length Codes," *IEEE Transactions On Communications*, vol. 49, pp. 1506-1509, 2001.

[12]. J. Wen and J. D. Villasenor, "A Class of Reversible Variable Length Codes for Robust Image and Video Coding," *Proceedings of the 1997 IEEE International Conference on Image Processing*, Santa Barbara, vol. 2, pp. 65–68, 1997.

[13]. H. -W. Tseng and C. -C. Chang "Construction of Symmetrical Reversible Variable Length Codes Using Backtracking," to appear in the *Journal of Computers*.

[14]. H. Murakami, S. Matsumoto and H. Yamamoto "Algorithm for Construction of Variable Length Code with Limited Maximum Word Length," *IEEE Transactions On Communications*, vol. com-32, pp. 1157-1159, 1984

Table 1. Comparisons of the maximum codeword lengths and the average codeword lengths of different RVLC construction algorithms for compressing different source files.

File	Number of Codewords	Huffman Code		Symmetric RVLC						Asymmetric RVLC					
				Tsai and Wu's		Tseng and Chang's		Algo-1		Tsai and Wu's		Algo-2		Algo-3	
		avg	max	avg	max	avg	max	avg	max	avg	max	avg	max	avg	max
asyoulik.txt	68	4.84465	15	5.27886	15	5.21025	12	**5.21025**	**12**	5.01142	15	**5.00954**	15	5.13624	**11**
alice29.txt	74	4.61244	16	5.01398	16	4.93155	12	**4.93155**	**12**	4.80326	17	**4.68871**	18	4.86762	**11**
xargs.1	74	4.92382	12	5.39863	12	5.33996	12	**5.33996**	**12**	5.07334	13	**5.16087**	15	5.27537	**11**
grammar.lsp	76	4.66434	12	5.04757	12	5.01774	12	**5.01774**	**12**	4.85461	12	**4.78581**	17	4.9613	**11**
plrabn12.txt	81	4.57534	19	4.94473	19	4.89527	14	**4.89527**	**14**	4.80659	19	**4.64910**	17	4.84043	**11**
lcet10.txt	84	4.69712	16	5.12232	16	5.01682	13	**5.01682**	**13**	4.87868	16	**4.74177**	17	4.93372	**11**
cp.html	86	5.26716	14	5.85839	14	5.81173	12	**5.81173**	**12**	5.37113	14	**5.77080**	16	5.7458	**11**
fields.c	90	5.04090	13	5.47596	13	5.46332	12	**5.46332**	**12**	5.26987	13	**5.20278**	13	5.36233	**11**
ptt5	159	1.66091	17	1.77735	17	1.75992	16	**1.75992**	**16**	1.71814	17	**1.70401**	15	1.72843	**13**
Sum	255	5.36504	14	6.10683	15	6.03917	15	**6.03917**	**15**	5.49767	13	**6.01870**	15	5.78572	**13**
kennedy.xls	256	3.59337	12	4.25681	17	4.27209	17	**4.21058**	**17**	3.89401	13	**3.85384**	14	3.86296	**14**

Table 2. Comparisons of the maximum codeword lengths and the average codeword lengths of different RVLC construction algorithms for compressing the English alphabet set.

Alphabet	Occurrence Probability	Huffman Code	Symmetric RVLC			Asymmetric RVLC		
			Tsai and Wu's	Tseng and Chang's	Alg.1	Tsai and Wu's	Alog. 2	Alog. 3
E	0.14878570	001	010	000	010	000	000	010
T	0.09354149	110	101	010	101	111	100	101
A	0.08833733	0101	0110	101	000	0101	101	000
O	0.07245769	0110	1001	111	111	1010	0010	111
R	0.06872164	0111	0000	0110	0110	0010	0011	0110
N	0.06498532	1001	1111	1001	1001	1101	0110	1001
H	0.05831331	1010	01110	00100	01110	0100	0111	01110
I	0.05644515	1011	10001	01110	10001	1011	1110	10001
S	0.05537763	1111	00100	10001	00100	0110	1111	00100
D	0.04376834	00000	11011	11011	11011	11001	01001	11011
L	0.04123298	00001	011110	001100	011110	10011	01010	011110
U	0.02762209	00011	100001	011110	100001	01110	01011	100001
P	0.02575393	01000	001100	100001	001100	10001	11001	001100
F	0.02455297	01001	110011	110011	110011	001100	11010	110011
M	0.02361889	11100	0111110	0010100	0111110	011110	11011	0111110
C	0.02081665	11101	1000001	0011100	1000001	100001	010001	1000001
W	0.01868161	000100	0010100	0111110	0010100	1001001	110001	0010100
G	0.01521216	100000	1101011	1000001	0011100	0011100	0100001	0011100
Y	0.01521216	100001	0011100	1100011	1100011	1100011	1100001	1100011
B	0.01267680	100010	1100011	1101011	1101011	0111110	01000001	1101011
V	0.01160928	100011	0001000	00111100	01111110	1000001	11000001	01111110
K	0.00867360	0001011	1110111	01111110	10000001	00111100	010000001	10000001
X	0.00146784	00010100	01111110	10000001	00111100	11000011	110000001	00101100
J	0.00080064	000101010	011111110	11000011	11000011	100101001	0100000001	00110100
Q	0.00080064	0001010110	0111111110	011111110	011111110	0011101001	1100000001	00111100
Z	0.00053376	0001010111	1000000001	100000001	100000001	1001011100	01000000001	11000011
Average Codeword Length		4.15572284	4.60728399	4.46463681	**4.4633**	4.30677804	**4.18734808**	4.4633
Maximum Codeword Length		10	10	9	**8**	10	11	**8**