GENERIC CODE DESIGN ALGORITHMS FOR REVERSIBLE VARIABLE-LENGTH CODES FROM THE HUFFMAN CODE

Wook-Hyun Jeong and Yo-Sung Ho

Kwangju Institute of Science and Technology (K-JIST)
1 Oryong-dong, Buk-gu, Kwangju, 500-712, Korea
E-mail: {whjeong, hoyo}@kjist.ac.kr

ABSTRACT

Variable-length codes (VLCs) are generally employed to improve compression efficiency using data statistics. However, VLCs are weak to bit errors in noisy transmission environments such as wireless channel. Recently, reversible variable-length codes (RVLCs) have been introduced due to their capability of recovering information from corrupted compressed streams; hence, enhancing the robustness of VLCs to bit errors. However, existing RVLCs are complicated in their design and have some room for improvement in coding efficiency. In this paper, we propose generic code construction algorithms both for symmetrical RVLCs and for asymmetrical RVLCs from the optimal Huffman code table. The proposed algorithms have simpler construction processes and also demonstrate improved performances in terms of the average codeword length than existing RVLC algorithms.

1. INTRODUCTION

Coding and transmission of video information over communication networks are popularly exploited in various applications, such as interactive video over the Internet, personal video communication over wireless networks, and video broadcasting over satellite. These systems usually require some error control mechanisms; however, most of the error recovery schemes provided by networks present many challenges. Thus, error resilience schemes have received a lot of attention from many researchers in order to make compressed video data robust to transmission errors.

Compressed video streams are very sensitive to bit errors. In particular, most image and video coding standards include variable-length codes (VLCs), such as the Huffman code[1] and the arithmetic code[2]; therefore, they are so sensitive to bit errors that the decoder may lose the synchronization. In addition, loss of synchronization may lead to the loss of several video frames.

In recent days, the reversible variable-length codes (RVLCs) have been introduced in order to reduce the effect of channel errors. In RVLC with a resynchronization marker, we can decode the bitstream both in the forward and backward directions and recover uncorrupted video data as much as possible from the received bitstream. On the other hand, in VLC, we have to throw out all the received data until the next resynchronization marker even after a single bit error.

RVLC can be categorized into two different classes: symmetrical and asymmetrical RVLCs according to their bit patterns. The symmetrical RVLC employs the same decoding table both for the forward and the backward directions. Although the asymmetrical RVLC offers better coding efficiency than the symmetrical RVLC owing to the more flexible code assignment, the asymmetrical RVLC requires two separate code tables. While MPEG-4 includes an asymmetrical RVLC[3], H.263+ employs a symmetrical RVLC[4].

Fraenkel *et al.*[5] presented necessary conditions for the existence of RVLCs. Takishima *et al.*[6] proposed the first work which specifies the method for constructing symmetrical and asymmetrical RVLCs based on a given Huffman code to make their average codeword lengths close to that of the optimal Huffman code. Tsai *et al.*[7] improved this algorithm and reduced the average codeword length. Recently, Tseng *et al.*[8] introduced a non-Huffman-code-based scheme for symmetrical RVLC. This is an exhaustive algorithm, with backtracking and a bounding. Lin *et al.*[9] extended Tseng's method to the asymmetrical RVLC.

Analysis of these four RVLC algorithms, however, shows some deficiency in terms of complexity and coding efficiency. Takishima's and Tsai's algorithms that are based on the Huffman code, have to pre-calculate the number of available symmetrical codewords at each level prior to construction of RVLC and some restrictions are imposed during the construction process. Consequently, some available RVLCs can be missed, which makes the resulting RVLC inefficient. Although Tseng's and Lin's algorithms provide better performance that conventional Huffman-code-based algorithms, they are not clear in many critical factors to design RVLCs, such as the starting bit length and codeword selection mechanisms. Besides, since Tseng and Lin assume that the sum of local optimizations can lead to the global optimization, which is not always true, naturally their proposed backtracking algorithms are limited. Therefore, conventional algorithms show room for improvement to construct more efficient RVLCs.

In this paper, we propose generic algorithms to construct symmetrical RVLCs and asymmetrical RVLCs based on the optimal Huffman code. The proposed algorithms search more efficient symmetrical and asymmetrical RVLCs to any given source than existing methods and also simplify construction processes using essential information from the Huffman code and the property of average length function.

2. THE SYMMETRICAL RVLC

2.1 Searching for Symmetrical RVLCs

VLCs, composed of only symmetrical codewords, can be decoded in two directions and they are called symmetrical RVLCs. This symmetric bit-pattern property can provide a useful advantage to the construction process. If all the bits of a symmetrical codeword are inversed, we can obtain another symmetrical codeword with the same bit length.

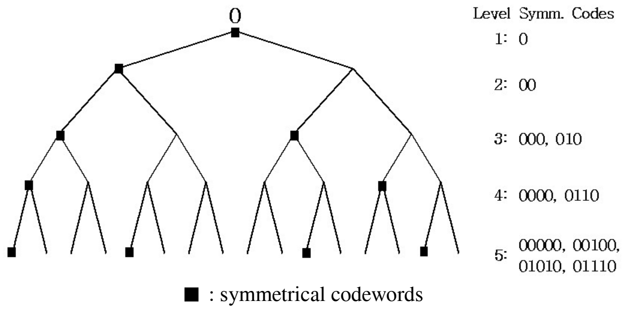


Fig 1. Distribution of symmetrical codewords

In order to exploit this property, we search symmetrical codewords in the half region, especially assigned to '0', of the binary Huffman tree; thus, the half region is the set that contains all Huffman codewords which start from '0' bit, as depicted in Fig 1. In this set, available candidates of the symmetrical RVLC are selected up to $\lceil S/2 \rceil$, not S, where S is the number of total symbols and $\lceil x \rceil$ is the smallest integer larger than or equal to x. Finally, we combine the previously chosen symmetrical codewords in the half region and their bit-inversed versions in the other region, allocated to '1', and we can obtain a target symmetrical RVLC for S, as shown in Fig 2[10].

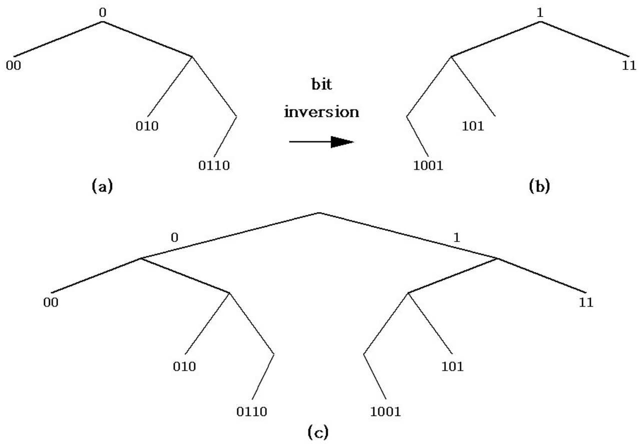


Fig 2. Construction process with bit-inversion

In Fig 2, we note that the symmetrical bit pattern makes the prefix condition guarantee the suffix condition and the bit-inversion operation cannot effect on the instantaneous decodability. The Kraft inequality for a Huffman binary tree is given by :

$$\sum_{i=1}^{S} 2^{-l_i} \le 1, (l_1 \le \ldots \le l_S) \tag{1}$$

where l_i is the encoded words of lengths. However, the choice of codewords is accomplished in one half region, that is the first sub-tree of a full binary tree, so that the Kraft inequality is revised from Eq. (1).

$$\sum_{i=1}^{S} 2^{-l_i} \le \frac{1}{2}, (l_1 \le \ldots \le l_{\lceil S/2 \rceil}) \tag{2}$$

When the selected symmetrical codewords in the half region set are prefix-free in Fig 2(a), reciprocal codewords with bit-inversion can be instantaneously decoded only in the other region in Fig 2(b). In this case, these codewords properly satisfy Eq. (2). A target symmetrical RVLC is presented with bit-inversed symmetrical codewords whose bit lengths are the same ones of already chosen codewords, as shown in Fig 2(c). Thus, Eq. (2) is extended to Eq. (3).

$$2 \times \sum_{i=1}^{\lceil S/2 \rceil} 2^{-l_i} \le 2 \times \frac{1}{2}, (l_1 \le \ldots \le l_{\lceil S/2 \rceil}) \tag{3}$$

Since l_i can be different from or equal to each other, the index i is extended to S. We obtain the complete Kraft inequality from Eq. (3).

$$\therefore \sum_{i=1}^{S} 2^{-l_i} \le 1, (l_1 \le \ldots \le l_S) \tag{4}$$

Therefore, the proposed search mechanism for symmetrical codewords with bit-inversion offers instantaneous decodable symmetrical RVLCs from Eq. (1) ~ Eq. (4).

2.2 $\mathbf{Z_L}$ Adaptation for Symmetrical RVLCs

There are symmetrical codewords that consist of all '0' bits at each level in the half region set, as shown in Fig 1. These codewords are defined as $\mathbf{Z_L}$s at level L. For $\mathbf{Z_L}$s, we can consider the following situations.

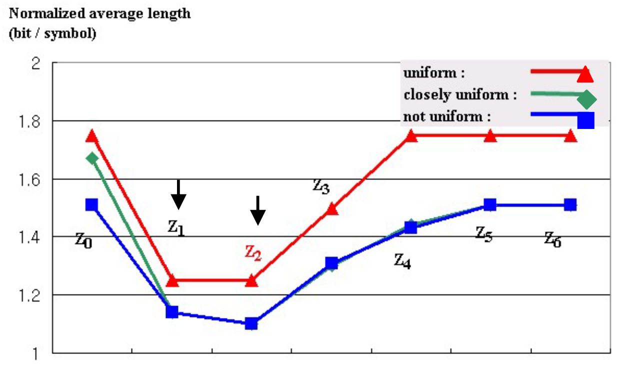
(1) If we select $\mathbf{Z_L}$, the number of '0' bits in both ends of each chosen symmetrical codeword in the half region set, except $\mathbf{Z_L}$, is restricted to (L—1).
(2) Since the bit-inversion operation is conducted, we can get the codeword composed of L '1' bits from $\mathbf{Z_L}$. Thus, these two codewords are added to level L. If $\mathbf{Z_L}$ is chosen carefully at a proper level L, the average length can be reduced successfully.

In order to construct an efficient symmetrical RVLC from the optimal Huffman code, the bit length L of $\mathbf{Z_L}$ is determined to be L_{min} that is the bit length of the shortest codeword in the given Huffman code. This process is called $\mathbf{Z_L}$ adaptation[10].

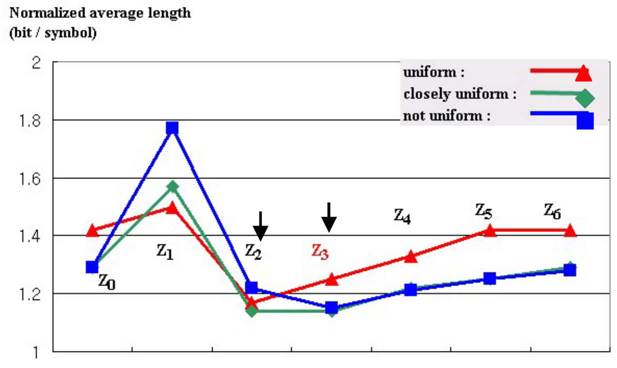
Fig 3 depicts the comparison of normalized average lengths of symmetrical RVLCs for several probability distributions. The normalized average length $\dot{N}$ is

$$\dot{N} = \frac{\overline{L}}{H(X)} \tag{5}$$

where $H(X)$ is the entropy of the given probability set and $\overline{L}$ denotes the average length of the symmetrical RVLC. We notice that average length of the symmetrical RVLC designed by the proposed $\mathbf{Z_L}$ adaptation is less than those of other symmetrical RVLCs. In case of the uniform distribution or nearly uniform distribution, however, the result of $\mathbf{Z}_{L-1}$ rather than $\mathbf{Z}_L$ has the better performance in terms of the average length. Thus, more efficient symmetrical RVLC is located on $\mathbf{Z}_{L-1}$ adaptation or $\mathbf{Z}_L$ adaptation empirically, as shown Fig3.



(a) L_{min} = 2



(b) L_{min} = 3

Fig 3. Normalized average length of symmetrical RVLCs

The proposed construction procedure for the symmetrical RVLC is presented as follows.

(1) In the half region, $\mathbf{Z}_L$ is determined. For example, if a given Huffman code starts from 3-bit codewords, that is, L_{min} is 3, select '000'.
(2) All available symmetrical candidates are searched from the root until the number of chosen codewords is $\lceil S/2 \rceil$. After each selection, remove codewords that violate the prefix condition.
(3) Construct a symmetrical RVLC by combining the already chosen codewords and their bit-inversed codewords.
(4) Repeat (2) and (3) with $\mathbf{Z}_{L-1}$ adaptation. If L_{min} is 3, we choose '00'.
(5) Compare the average lengths of two symmetrical RVLCs and select a more efficient RVLC.

3. THE ASYMMETRICAL RVLC

Asymmetrical RVLCs are determined by codewords selected at the starting level and the number of these codewords. Since asymmetrical RLVCs do not concern bit patterns of codewords, they are close to the Huffman code. Thus, we adopt a critical description, L_{min} of a given Huffman code, as the starting level of an asymmetrical RVLC.

Let the bit length vector $n(i)$ denote the number of codewords with the bit length i, and $n_{Huff}(i)$ and $n_{RVLC}(i)$ be bit length vectors of the Huffman code and the RVLC, respectively. In order to design more efficient RVLCs, we find codewords to be selected at L_{min} and $n_{RVLC}(L_{min})$ with an exhaustive search; however, we take a fast approach to the optimal RVLC using $n_{Huff}(L_{min})$, $\mathbf{Z}_L$, and the minimum repetition gap (MRG) proposed by Tsai[7].

In addition, the property of the average length function is useful to the construction procedure. We use the average length, which is a convex function, as the measurement of coding performance. $n_{RVLC}(L_{min})$ ranges from 1 to 2^{Lmin} and the average length function has the minimum value in this interval. The most optimal RVLC exists at this minimum point, which should be the global optimal point, not the local optimal point. During the design procedure, we should be careful to avoid the local minimum point. Thus, we determine $n_{RVLC}(L_{min})$ by checking the average lengths of typical RVLCs based on MRG, which makes the proposed design process pass the locally minimum values.

We present a construction algorithm for asymmetrical RVLCs below, and Fig 4 illustrates an example of this procedure.

(1) Determine the starting point of $n_{test}(L_{min})$ where $n_{test}(L_{min})$ is the bit length vector of each typical asymmetrical RVLC. If $n_{Huff}(L_{min}) \leq 2^{Lmin-1}$, $n_{test}(L_{min})$ is increased from 1; otherwise, $n_{test}(L_{min})$ is decreased from 2^{Lmin}, as shown in Fig 4.
(2) Construct each typical RVLC involving $n_{test}(L_{min})$ codewords prioritized based on MRG.. The priority of codewords is in the ascending order in MRG. For instance, if L_{min} is 3, '000', '111', '010', '101', '001', '011', '100', and '110' are arranged. As shown in Fig 4, if the average length of the current typical RVLC is larger than that of the previous one, stop constructing typical RVLCs. $n_{RVLC}(L_{min})$ is the previous $n_{test}(L_{min})$ with minimum value.
(3) Select $\mathbf{Z}_L$ as one of $n_{RVLC}(L_{min})$ codewords.
(4) In order to find the other $(n_{RVLC}(L_{min})-1)$ codewords, creat $\binom{2^{Lmin}-1}{n_{RVLC}(L_{min})-1}$ asymmetrical RVLCs with $\mathbf{Z}_L$ where $\binom{a}{b}$ is the number of combinations of size b from a set of size a. Compare the average length of each RVLC and select the most efficient one.

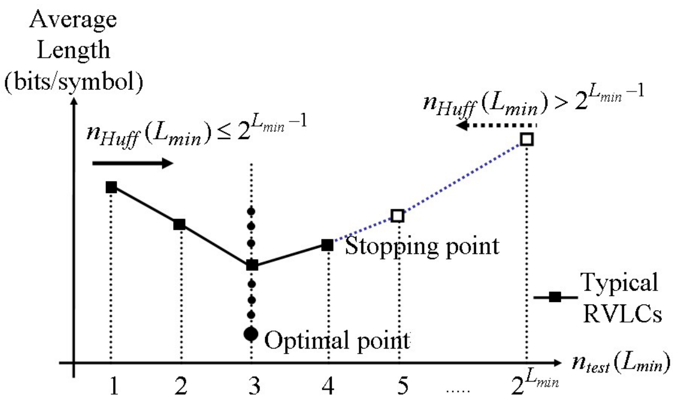


Fig 4. Construction procedure for Asymmetrical RVLCs

4. EXPERIMENTAL RESULTS

Table I and Table II list codeword assignments for the English alphabet with symmetrical RVLCs and asymmetrical RVLCs designed by the existing algorithms and the proposed algorithms, and compare their coding performances in terms of the average codeword length.

In Table I, S1, S2, and S3 are generated by the Huffman, Takishima's[6], and Tsai's[7] algorithms, respectively. Results of S2 and S3 show that their algorithms miss some available symmetrical RVLCs at Level 3, Level 8, and Level 9 in spite of the existence of other available candidates at those levels. At those levels, bit length vectors of symmetrical RVLCs should be restricted to those of the given Huffman code, which impose some redundancies to a target symmetrical RVLC. In addition, these schemes construct RVLCs on the Huffman binary tree as many as *S*. After the pre-computation of the number of available codewords at each level, they repeat the adaptation and search process, whose Big-Oh notation of complexity is $O(N^2 \cdot (\sqrt{2})^N)|_{N=S}$.

In Table I, S5 is designed by the proposed algorithm. Experimental results indicate that the average codeword length of S5 is about 5.2% and 3.2% shorter than those of S2 and S3, respectively. In addition, S5 is composed of pairs of bit inversion, $\mathbf{Z}_3$ and '111' are allocated to the most probable symbols by $\mathbf{Z}_\mathrm{L}$ adaptation, and effective symmetrical candidates are not omitted at any level. The complexity of the proposed RVLC scheme is $O(N^2 \cdot (\sqrt[4]{2})^N)|_{N=S}$. The proposed algorithm for symmetrical RVLC reduces the number of symbols to be handled and the search range by half. This reduction makes a difference of complexity, $O((\sqrt[4]{2})^N)|_{N=S}$ between conventional Huffman-code-based schemes and the proposed schemes.

S4, in Table I, is generated by Tseng's[8] algorithm. Tseng's method seems to overcome existing Huffman based methods in the viewpoint of efficiency. However, in the worst case, there exists a significant difference of complexity between Tseng's algorithm and the new construction algorithm, which is about $O(N^3)|_{N=S}$.

In Table II, A1, A2, A3, A4, and A5 are constructed by the Huffman, Takishima's[6], Tsai's[7], Lin's[9], and the proposed algorithms, respectively. In case of A2 and A3, still, some restrictions happen at Level 3 in A2, and at Level 3, Level 9, and Level 10 in A3.

Although Lin's algorithm reduces the average length sufficiently, some room for improvement exists. For asymmetrical RVLCs, conventional algorithms, either Huffman-code-based or non-Huffman-code-based algorithms overlook that each asymmetrical codeword is determined by only specific codewords at L_{min}.

The proposed approach to the more optimal asymmetrical RVLC reduces the average length considerably, as shown in Table II. The average length of A5 is about 5.7%, 3.1%, and 0.34% shorter than those of A2, A3, and A4, respectively. Since Lin's algorithm is derived from Tseng's algorithm, the difference of complexity $O(N^3)|_{N=S}$ is fixed in the worst case.

5. CONCLUSIONS

In this paper, we have proposed new generic algorithms to construct symmetrical RVLCs and asymmetrical RVLCs based on the optimal Huffman code. For symmetrical RVLC, we search symmetrical candidates in the half region set with $\mathbf{Z}_\mathrm{L}$ adaptation and we generate more efficient symmetrical RVLC by using bit-inversion. In case of asymmetrical RVLC, we accomplish an exhaustive search. However, by using critical factors from the given Huffman code and the property of average length function, we find fast approach to more efficient asymmetrical RVCL. Our experimental results have demonstrated that the proposed generic algorithms simplify the construction processes and improve coding performance over existing RVLC algorithms.

ACKNOWLEDGEMENT

This work was supported in part by K-JIST, in part by the Ministry of Information and Communication (MIC) through the Realistic Broadcasting Research Center at K-JIST, and in part by the Ministry of Education (MOE) through the Brain Korea 21 (BK21) project.

REFERENCES

[1] D. Huffman, "A method for the construction of minimum redundancy codes," *Proc. Inst. Radio. Engr.,* vol. 40, pp. 1098-1101, Sept. 1952.

[2] J.J. Rissanen and G.G. Langdon. Jr., "Arithmetic coding," *IBM J. Res. Develop.*, 23, pp. 149-162, 1979.

[3] ISO/IEC 14496-2, "Information technology – coding of audio/video objects," *Final Draft Int. Std.*, Part 2 : Visual, Oct. 1998.

[4] ITU-T Rec. H.263, "Video coding for low bit communications," Annex V, 2000.

[5] A.S. Fraenkel and S.T. Klein, "Bidirectional Huffman coding," *Comp. J.*, vol. 33, no. 4, 1990.

[6] Y. Takishima, M. Wada and H. Murakami, "Reversible variable length codes," *IEEE Trans. Comm.*, vol. 43, pp. 158-162, Feb. 1995.

[7] C.W. Tsai and J.L. Wu, "On constructing the Huffman-code-based reversible variable-length codes," *IEEE Trans. Comm.*, vol 49, pp. 1506-1509, Sept. 2001.

[8] H.W. Tseng and C.C. Chang, "Construction of symmetrical reversible variable length codes using backtracking," *Comp. J.*, vol. 46, no. 1, 2003.

[9] C.W. Lin, Y.J. Chuang, and J.L. Wu, "Generic construction algorithms for symmetric and asymmetric RVLCs," *Proc. IEEE Int. Conf. Communication Systems*, vol. 2, pp. 968-972, Nov. 2002.

[10] W.H. Jeong and Y.S. Ho, "Design of symmetrical reversible variable-length codes from the Huffman code," *Picture Coding Symposium*, pp. 135-138, April 2003.

Table I. Coding performances of symmetrical RVLCs for the English alphabet

Occurrence Probability		**Huffman : S1**		**Takishima's algorithm : S2**		**Tsai's algorithm: S3**		**Tseng's algorithm : S4**		**Proposed algorithm : S5**	
		L	**codeword**	**L**	**codeword**	**L**	**codeword**	**L**	**codeword**	**L**	**codeword**
E	0.14878570	3	001	3	000	3	010	3	000	3	000 (**Z_3**)
T	0.09354149	3	110	3	111	3	101	3	111	3	111
A	0.08833733	4	0000	4	0110	4	0110	3	010	3	010
O	0.07245796	4	0100	4	1001	4	1001	3	101	3	101
R	0.06872164	4	0101	5	00100	4	0000	4	0110	4	0110
N	0.06498532	4	0110	5	11011	4	1111	4	1001	4	1001
H	0.05831331	4	1000	5	01010	5	01110	5	00100	5	00100
I	0.05644515	4	1001	5	10101	5	10001	5	11011	5	11011
S	0.05537763	4	1010	5	01110	5	00100	5	01110	5	01110
D	0.04376834	5	00010	5	10001	5	11011	5	10001	5	10001
L	0.04123298	5	00011	6	001100	6	011110	6	001100	6	001100
U	0.02762209	5	10110	6	110011	6	100001	6	110011	6	110011
P	0.02575393	5	10111	6	010010	6	001100	6	011110	6	011110
F	0.02455297	5	11100	6	101101	6	110011	6	100001	6	100001
M	0.02361889	5	11110	6	011110	7	0111110	7	0010100	7	0010100
C	0.02081665	5	11111	6	100001	7	1000001	7	1101011	7	1101011
W	0.01868161	6	011100	7	0010100	7	0010100	7	0011100	7	0011100
G	0.01521216	6	011101	7	1101011	7	1101011	7	1100011	7	1100011
Y	0.01521216	6	011110	7	0011100	7	0011100	7	0111110	7	0111110
B	0.01267680	6	011111	7	1100011	7	1100011	7	1000001	7	1000001
V	0.01160928	6	111011	7	0100010	7	0001000	8	00111100	8	00111100
K	0.00867360	7	1110100	7	1011101	7	1110111	8	11000011	8	11000011
X	0.00146784	8	11101011	8	00111100	8	01111110	8	01111110	8	01111110
J	0.00080064	9	111010101	9	001010100	9	011111110	8	10000001	8	10000001
Q	0.00080064	10	1110101000	10	0010110100	10	0111111110	9	011111110	9	011111110
Z	0.00053376	10	1110101001	10	1101001011	10	1000000001	9	100000001	9	100000001
Average length		4.15572392		4.69655649		4.60728507		4.46463681		**4.46463681**	

Table II. Coding performances of asymmetrical RVLCs for the English alphabet

Occurrence Probability		**Huffman : A1**		**Takishima's algorithm : A2**		**Tsai's algorithm: A3**		**Lin's algorithm : A4**		**Proposed algorithm : A5**	
		L	**codeword**	**L**	**codeword**	**L**	**codeword**	**L**	**codeword**	**L**	**codeword**
E	0.14878570	3	001	3	001	3	000	3	000	3	000 (**Z_3**)
T	0.09354149	3	110	3	110	3	111	3	100	3	101
A	0.08833733	4	0000	4	0000	4	0101	3	101	3	110
O	0.07245796	4	0100	4	0100	4	1010	4	0010	4	0010
R	0.06872164	4	0101	4	0101	4	0010	4	0011	4	0011
N	0.06498532	4	0110	4	1000	4	1101	4	0110	4	0100
H	0.05831331	4	1000	4	1010	4	0100	4	0111	4	0111
I	0.05644515	4	1001	5	10010	4	1011	4	1110	4	1001
S	0.05537763	4	1010	5	01100	4	0110	4	1111	4	1111
D	0.04376834	5	00010	5	00010	5	11001	5	01001	5	01010
L	0.04123298	5	00011	5	00011	5	10011	5	01010	5	01011
U	0.02762209	5	10110	5	10111	5	01110	5	01011	5	01100
P	0.02575393	5	10111	5	11100	5	10001	5	11001	5	10001
F	0.02455297	5	11100	5	11111	6	001100	5	11010	5	11100
M	0.02361889	5	11110	6	111101	6	011110	5	11011	6	011010
C	0.02081665	5	11111	6	101101	6	100001	6	010001	6	011011
W	0.01868161	6	011100	6	011101	7	1001001	6	110001	6	100001
G	0.01521216	6	011101	6	111011	7	0011100	7	0100001	6	111010
Y	0.01521216	6	011110	8	01110011	7	1100011	7	1100001	6	111011
B	0.01267680	6	011111	8	11101011	7	0111110	8	01000001	7	1000001
V	0.01160928	6	111011	9	111010011	7	1000001	8	11000001	8	10000001
K	0.00867360	7	1110100	9	011110011	8	00111100	9	010000001	9	100000001
X	0.00146784	8	11101011	10	0111110011	8	11000011	9	110000001	10	1000000001
J	0.00080064	9	111010101	10	1110101011	9	100101001	10	0100000001	11	10000000001
Q	0.00080064	10	1110101000	11	11101010011	10	0011101001	10	1100000001	12	100000000001
Z	0.00053376	10	1110101001	13	1110101000111	10	1001011100	11	01000000001	13	1000000000001
Average length		4.15572392		4.42607344		4.30677804		4.18734808		**4.172804**	