

Design Considerations for Efficient Reversible Variable Length Codes

Ksenija Laković and John Villasenor

Electrical Engineering Department, University of California, Los Angeles
{ksenija, villa}@icsl.ucla.edu

Abstract

This paper considers the properties of reversible variable length codes (RVLCs) that are relevant for construction of highly efficient codes. We establish the dependency between the codeword selection process and the efficiency-limiting parameters in the Huffman-code-based RVLC construction algorithm. Using this result, we propose an RVLC construction algorithm with a new codeword selection mechanism. The proposed algorithm yields RVLCs of higher efficiency relative to the other algorithms in the literature.

1 Introduction

Reversible variable length codes (RVLCs) have been accepted as an alternative to Huffman codes in the transmission of digital signals with non-uniform occurrence probability distributions. As the name suggests, RVLCs allow instantaneous decoding in both directions, which can be exploited by a decoder to enhance robustness in the presence of transmission bit errors. One of the first publications to treat RVLCs was by Takishima *et al.* [1], who proposed Huffman-code based algorithms for RVLC construction. The research topics regarding the application of RVLCs have received extensive attention recently [2], [3], especially in the development of new video standards H.263+ and MPEG-4.

Like Huffman codes, RVLCs must satisfy the prefix condition for instantaneous forward decoding, but, unlike Huffman codes, they must additionally satisfy the suffix condition for instantaneous backward decoding. The prefix condition is that each codeword does not coincide with the prefixes of longer codewords, while the suffix condition expresses there is no coincidence with the suffixes of longer codewords. Therefore, RVLCs are a special class of prefix-free codes, and are also referred to as fix-free codes.

It is well known that prefix-free codes are optimally constructed by the Huffman algorithm. Furthermore, the conditions for the existence of Huffman codes and the bounds on their redundancy are established in the literature [4]. Several researchers have considered the conditions for the existence of fix-free codes [5-7], and the bounds on their redundancy [8]. However, an algorithm for optimal construction of fix-free codes has not yet been

proposed. The recent publication of Tsai and Wu [9] treats the strategies for construction of efficient symmetric and asymmetric RVLCs. It also proposes a suboptimal generic algorithm for RVLC construction, which yields better RVLC efficiency than the original algorithms by Takishima *et al.* [1].

Takishima's algorithm [1] constructs a reversible variable length code C for a source \mathcal{U} by starting with a Huffman code C_H for the same source. The Huffman codewords that do not violate the suffix condition are assigned as RVLC codewords, while the suffix condition violating Huffman codewords are extended by bit addition into minimal length codewords that satisfy the suffix condition. The major disadvantage of this simple method is that the efficiency of the constructed RVLC depends on the bit-alignment pattern of the starting Huffman code, i.e. that two Huffman codes of the same efficiency generally yield RVLCs of different efficiency [1], [9].

Algorithms proposed by Tsai and Wu [9], [10] overcome this problem by taking the Huffman code's bit-length vector as the only input to the RVLC construction algorithm. RVLC codeword selection is performed independently of the bit-alignment pattern of the starting Huffman code, using the proposed selection mechanisms for symmetric and asymmetric RVLCs [9], [10]. The authors conjecture that these mechanisms yield highly-efficient RVLCs, and illustrate their effectiveness by simulation results.

In this paper we formally establish the relationship between the codeword selection process and the efficiency limiting parameters in the RVLC construction algorithm. Using this result, we propose an RVLC construction algorithm with a new codeword selection mechanism, and show that it yields better results than the algorithms previously proposed in the literature [1], [9].

2 Huffman-code-based construction of RVLC

We consider the construction of a reversible variable length code C for an M -ary i.i.d. information source $\mathcal{U} = \{u^1, \dots, u^M\}$, with the probability mass function $p_U = \{p^1, \dots, p^M\}$, $p^1 \leq p^2 \leq \dots \leq p^M$. The RVLC construction algorithm involves the following steps:

1. Implement the Huffman algorithm [4], to produce a Huffman code C_H , which maps each source symbol $U \in \mathcal{U}$ into a corresponding binary codeword $c_H(U) \in \{c_H(u^1), \dots, c_H(u^M)\}$ of length:

This work was supported by the Office of Naval Research under Contract N00014-01-C-016.

$$l_H(u) \in \{l_H(u^1), \dots, l_H(u^M)\} \quad (1)$$

$$l_H(u^1) \leq l_H(u^2) \leq \dots \leq l_H(u^{M-1}) = l_H(u^M) = L_{H_max}$$

The average codeword length of this code is computed as:

$$L_{H_avr} = \sum_{j=1}^M p^j \cdot l_H(u^j) \quad (2)$$

The bit length vector of C_H is denoted by $(n_H(1), \dots, n_H(L_{H_max}))$, where $n_H(i)$ represents the number of Huffman codewords having length i , which can be written as:

$$n_H(i) = |\{c_H(u^k) | u^k \in \mathcal{U}, l_H(u^k) = i\}| \quad (3)$$

2. Initialize the number of assigned RVLC codewords as $m=0$, and the RVLC bit length vector $(n(1), \dots, n(L_{H_max}))$ as:

$$n(i) = n_H(i), \quad i=1 \dots L_{H_max} \quad (4)$$

Start the assignment of RVLC codewords at level $i=L_{H_min}$, where L_{H_min} is the minimum codeword length of the code C_H .

3. At level $i=L_{H_min}$ all length- i binary strings can be assigned as RVLC codewords. At higher levels $i > L_{H_min}$ the number of available strings is smaller than 2^i ($n_a(i) < 2^i$), because the length- i strings whose prefixes or suffixes coincide with previously assigned (shorter) codewords cannot be assigned as RVLC codewords. Therefore, prior to the codeword assignment at level i , it is necessary to identify the strings that are neither prefixed nor suffixed by the previously assigned codewords, i.e. to identify the set of available length- i binary strings $\mathcal{B}_A = \{b^1, b^2, \dots, b^{n_a(i)}\}$. The assignment of RVLC codewords at level i goes as follows:

- 3.1. If $n_a(i) \geq n(i)$, then $n(i)$ strings out of the available $n_a(i)$ ones are assigned as the RVLC codewords. The selection details are discussed in Section 4, where we propose a selection mechanism (see (27)), and compare it to selection mechanisms from the literature [9].
- 3.2. If $n_a(i) < n(i)$ all available strings are assigned as the RVLC codewords. The remaining $n(i) - n_a(i)$ codewords are set to be assigned at a higher level, which results in the increase of the average codeword length of the RVLC code relative to the average codeword length of the Huffman code. More precisely, it results in the following relationship between the RVLC codeword lengths $l(u^k)$ and the Huffman codeword lengths $l_H(u^k)$:

$$l(u^k) > l_H(u^k) \quad (5)$$

$$k = m + n_a(i) + 1, m + n_a(i) + 2, \dots, m + n(i)$$

The corresponding adjustment in the RVLC bit-length vector is:

$$n(i) = n_a(i) \quad (6)$$

$$n(i+1) = n(i+1) + n(i) - n_a(i)$$

4. After the assignment of the RVLC codewords at level i is completed, the number of the assigned RVLC codewords is updated as $m=m+n(i)$. If $m < M$, the RVLC codeword assignment (step 3) continues at level $i=i+1$. If $m=M$ the construction of the RVLC code is completed.

3 Some properties of reversible variable length codes

As shown in the previous section, the RVLC efficiency is limited by the number of available strings that are neither prefixed nor suffixed by the previously assigned codewords. In this section we establish the relationship between the number of available strings at level j , $n_a(j)$, and the structure of the previously selected codewords $\{c(u^1), c(u^2), \dots, c(u^m)\} = \{c^1, c^2, \dots, c^m\}$, of lengths $\{l(u^1), \dots, l(u^m)\} = \{i_1, \dots, i_m\}$, $i_1 \leq i_2 \leq \dots \leq i_m < j$.

Definition 1: ($\mathcal{P}_j(c)$, Prefix set) Consider an RVLC codeword $c = (c_1, c_2, \dots, c_i)$ of length i . A prefix set $\mathcal{P}_j(c)$ is defined as the set of length- j binary strings, $j > i$, which are prefixed by c :

$$\mathcal{P}_j(c) = \{(b_1, b_2, \dots, b_j) | (b_1, b_2, \dots, b_i) = (c_1, c_2, \dots, c_i)\} \quad (7)$$

Definition 2: ($\mathcal{S}_j(c)$, Suffix set) Consider an RVLC codeword $c = (c_1, c_2, \dots, c_i)$ of length i . A suffix set $\mathcal{S}_j(c)$ is defined as the set of length- j binary strings, $j > i$, which are suffixed by c :

$$\mathcal{S}_j(c) = \{(b_1, b_2, \dots, b_j) | (b_{j-i+1}, b_{j-i+2}, \dots, b_j) = (c_1, c_2, \dots, c_i)\} \quad (8)$$

Definition 3: ($\mathcal{A}_j(c^1, c^2)$ Affix set) Consider two RVLC codewords $c^1 = (c_1^1, c_2^1, \dots, c_{i_1}^1)$ and $c^2 = (c_1^2, c_2^2, \dots, c_{i_2}^2)$ of lengths i_1 and i_2 . An affix set $\mathcal{A}_j(c^1, c^2)$ is defined as the set of length- j binary strings, $j > i_1, j > i_2$, which are prefixed by c^1 and suffixed by c^2 :

$$\mathcal{A}_j(c^1, c^2) = \{(b_1, b_2, \dots, b_j) | (b_1, b_2, \dots, b_{i_1}) = (c_1^1, c_2^1, \dots, c_{i_1}^1), \quad (9)$$

$$(b_{j-i_2+1}, b_{j-i_2+2}, \dots, b_j) = (c_1^2, c_2^2, \dots, c_{i_2}^2)\}$$

Denote the cardinality of the prefix set $\mathcal{P}_j(c)$ with $P_j(c)$ and the cardinality of the suffix set $\mathcal{S}_j(c)$ with $S_j(c)$. Assuming c is an RVLC codeword of length i , it is obvious, with regard to (7) and (8), that:

$$P_j(c) = S_j(c) = 2^{j-i} \quad (10)$$

Theorem 1: Let $A_j(c^1, c^2)$ denote the cardinality of the affix set $\mathcal{A}_j(c^1, c^2)$, i.e. $A_j(c^1, c^2) = |\mathcal{A}_j(c^1, c^2)|$, where c^1 and c^2 are RVLC codewords of lengths i_1 and i_2 respectively.

If $j \geq i_1 + i_2$, then

$$A_j(c^1, c^2) = 2^{j-i_1-i_2} \quad (11)$$

If $i_1, i_2 < j < i_1 + i_2$, then

$$A_j(c^1, c^2) = \begin{cases} 1, & \text{if } (c_{j-i_2+1}^1, c_{j-i_2+2}^1, \dots, c_{i_1}^1) = (c_1^2, c_2^2, \dots, c_{i_1+i_2-j}^2) \\ 0, & \text{if } (c_{j-i_2+1}^1, c_{j-i_2+2}^1, \dots, c_{i_1}^1) \neq (c_1^2, c_2^2, \dots, c_{i_1+i_2-j}^2) \end{cases} \quad (12)$$

Proof: Consider a length- j binary string $b = (b_1, b_2, \dots, b_j)$ satisfying $b \in \mathcal{A}_j(c^1, c^2)$.

Case 1: $j \geq i_1 + i_2$

According to (9) the first i_1 bits of b must be the same as c^1 and the last i_2 bits of b must be the same as c^2 , while each of its $j - i_1 - i_2$ intermediate bits is either "0" or "1". Therefore, there are $2^{j-i_1-i_2}$ strings of length j that are both prefixed by c^1 and suffixed by c^2 .

Case 2: $i_1, i_2 < j < i_1 + i_2$

Assume $(c_{j-i_2+1}^1, c_{j-i_2+2}^1, \dots, c_{i_1}^1) = (c_1^2, c_2^2, \dots, c_{i_1+i_2-j}^2)$. In this case, there is exactly one codeword:

$$b = (c_1^1, c_2^1, \dots, c_{j-i_1}^1, c_1^2, c_2^2, \dots, c_{i_2}^2) \quad (13)$$

that satisfies both $(b_1, b_2, \dots, b_{i_1}) = (c_1^1, c_2^1, \dots, c_{i_1}^1)$ and $(b_{j-i_2+1}, b_{j-i_2+2}, \dots, b_j) = (c_1^2, c_2^2, \dots, c_{i_2}^2)$; thus $A_j(c^1, c^2) = 1$. Otherwise, no string of length j satisfying these two conditions exists, and $A_j(c^1, c^2) = 0$.

Theorem 2: Consider a set of RVLC codewords $C^{(m)} = \{c^1, c^2, \dots, c^m\}$, of lengths $i_1 \leq i_2 \leq \dots \leq i_m < j$. The number of length- j strings that are neither prefixed nor suffixed by any of the codewords $\{c^1, c^2, \dots, c^m\}$ is equal to:

$$n_a(j) = 2^j - 2 \sum_{k=1}^m 2^{j-i_k} + \sum_{k=1}^m \sum_{l=1}^m A_j(c^k, c^l) \quad (14)$$

Proof: Because the total number of length- j strings is 2^j , the number of those that are neither prefixed nor suffixed by any of the codewords $\{c^1, c^2, \dots, c^m\}$ is:

$$\begin{aligned} n_a(j) &= 2^j - \left| \left(\bigcup_{k=1}^m \mathcal{P}_j(c^k) \right) \cup \left(\bigcup_{l=1}^m \mathcal{S}_j(c^l) \right) \right| \\ &= 2^j - \left| \bigcup_{k=1}^m \mathcal{P}_j(c^k) \right| - \left| \bigcup_{l=1}^m \mathcal{S}_j(c^l) \right| + \left| \left(\bigcup_{k=1}^m \mathcal{P}_j(c^k) \right) \cap \left(\bigcup_{l=1}^m \mathcal{S}_j(c^l) \right) \right| \\ &= 2^j - \left| \bigcup_{k=1}^m \mathcal{P}_j(c^k) \right| - \left| \bigcup_{l=1}^m \mathcal{S}_j(c^l) \right| + \left| \bigcup_{k=1}^m \bigcup_{l=1}^m (\mathcal{P}_j(c^k) \cap \mathcal{S}_j(c^l)) \right| \end{aligned} \quad (15)$$

Recall that no RVLC codeword is a prefix or a suffix of another codeword. This implies that all the prefix sets, as well as all the suffix sets, are disjoint, i.e.

$$\begin{aligned} |\mathcal{P}_j(c^k) \cap \mathcal{P}_j(c^l)| &= 0, \quad k, l \in \{1 \dots N\}, k \neq l \\ |\mathcal{S}_j(c^k) \cap \mathcal{S}_j(c^l)| &= 0, \quad k, l \in \{1 \dots N\}, k \neq l \end{aligned} \quad (16)$$

Therefore:

$$n_a(j) = 2^j - \sum_{k=1}^m |\mathcal{P}_j(c^k)| - \sum_{l=1}^m |\mathcal{S}_j(c^l)| + \sum_{k=1}^m \sum_{l=1}^m |\mathcal{P}_j(c^k) \cap \mathcal{S}_j(c^l)| \quad (17)$$

Substituting (10) into (17) yields:

$$n_a(j) = 2^j - \sum_{k=1}^m 2^{j-i_k} - \sum_{l=1}^m 2^{j-i_l} + \sum_{k=1}^m \sum_{l=1}^m |\mathcal{P}_j(c^k) \cap \mathcal{S}_j(c^l)| \quad (18)$$

Finally, notice that $\mathcal{A}_j(c^k, c^l) = \mathcal{P}_j(c^k) \cap \mathcal{S}_j(c^l)$, which completes the proof.

Theorem 2 establishes the relationship between the number of available codewords at any level of the RVLC construction algorithm and the structure of the previously selected codewords. This result will be used in the discussion on RLVC codeword selection mechanisms in the following section. To simplify the interpretation of Theorem 2 and the related discussion we will introduce the following notation.

Definition 4: ($A_j(C^{(m)})$ **Affix index**) Consider a set of codewords $C^{(m)} = \{c^1, c^2, \dots, c^m\}$. The affix index is defined as:

$$A_j(C^{(m)}) = \sum_{k=1}^m \sum_{l=1}^m A_j(c^k, c^l) \quad (19)$$

where $A_j(c^k, c^l)$ denotes the cardinality of the affix set $\mathcal{A}_j(c^k, c^l)$, $c^k, c^l \in C^{(m)}$.

4 On Construction of Efficient RVLCs

In this section we discuss methods of improving an RVLC code efficiency by optimizing the codeword selection process in the RLVC construction algorithm described in Section 2. We start by considering the selection process at level $i = L_{H, \min}$. At this level $n(i)$ strings are to be selected out of the available $n_a(i) = 2^i$ ones. Denote the set of available strings as $\mathcal{B}_A = \{b^1, b^2, \dots, b^{n_a(i)}\}$. Since $n_a(i) \geq n(i)$, $n(i)$ strings from \mathcal{B}_A should be assigned as the RVLC codewords. There are obviously:

$$S = \begin{pmatrix} n_a(i) \\ n(i) \end{pmatrix} \quad (20)$$

possible selections. Denote one candidate selection with $\mathcal{B}^{(n(i))}$, and the set of candidate selections with:

$$\mathcal{B} = \{ \mathcal{B}^{(n(i))} \mid \mathcal{B}^{(n(i))} \subseteq \mathcal{B}_A, \ |\mathcal{B}^{(n(i))}| = n(i) \} \quad (21)$$

Assuming that the strings from the set $\mathcal{B}^{(n(i))} \in \mathcal{B}$ are assigned as the RVLC codewords at level i , the number of available strings at levels $n_a(j)$, $j > i$, is given by Theorem 2:

$$n_a(j) = 2^j - n(i) \cdot 2^{j-i+1} + A_j(\mathcal{B}^{(n(i))}) \quad (22)$$

Furthermore, because of (11) and (12):

$$\begin{aligned} A_j(\mathcal{B}_i^{(n(i))}) &= n(i)^2 \cdot 2^{j-2i}, & j \geq 2i \\ 0 \leq A_j(\mathcal{B}_i^{(n(i))}) &\leq n(i)^2, & i < j < 2i \end{aligned} \quad (23)$$

It can be easily noticed from (22) and (23) that the number of available strings $n_a(j)$ at levels $j \geq 2i$ does not depend on the codeword selection at level i . On the other hand, for $i < j < 2i$, the best-case selection at level i can yield up to an $n(i)^2$ increase in $n_a(j)$, relative to the worst-case selection at level i .

Obviously, the highest $n_a(j)$, $i < j < 2i$, is achieved by the selection of the set with the maximal value of $A_j(\mathcal{B}_i^{(n(i))})$, denoted

$$\mathcal{B}_{i^*}^{(n(i))}(j) = \arg \max_{\mathcal{B}_i \in \mathcal{B}} A_j(\mathcal{B}_i^{(n(i))}) \quad (24)$$

Unfortunately, as can be inferred from (12), the set $\mathcal{B}_i^{(n(i))}$ that maximizes $n_a(j)$ does not necessarily maximize $n_a(j_2)$, i.e. it generally holds that $\mathcal{B}_{i^*}^{(n(i))}(j_1) \neq \mathcal{B}_{i^*}^{(n(i))}(j_2)$, $i < j_1 < j_2 < 2i$. This effect is illustrated with the example presented in Table I. For the considered selection of $n(i)=2$ strings at level $i=3$, the highest value of $A_4(\mathcal{B}_3^{(2)})$ is achieved with the sets $\{0,1\}$, $\{0,4\}$, $\{0,7\}$, $\{2,5\}$ (and their bit-inverse counterparts). However, none of these sets are associated with the highest value of $A_5(\mathcal{B}_3^{(2)})$, because it is achieved with the set $\{0,2\}$, i.e. $\mathcal{B}_{3^*}^{(2)}(3) = \{0,2\}$.

This discussion can be easily extended to the selection of RVLC codewords at a level $i > L_{H_min}$. The difference is that, in this case, it is necessary to consider the RVLC codewords that have been assigned at earlier levels. Assuming that the codewords $C^{(m)} = \{c^1, c^2, \dots, c^m\}$, are assigned at levels $i_1 \leq i_2 \leq \dots \leq i_m < i$ the number of available length- i strings, $n_a(i)$, is computed using (14). For $n_a(i) \geq n(i)$, the selection process can be formulated using the same framework as above (see (21)). In this case, however, the selection $\mathcal{B}_i^{(n(i))} \in \mathcal{B}$ at level i , yields the number of available strings $n_a(j)$, $j > i$, equal to:

$$n_a(j) = 2^j - 2 \sum_{k=1}^m 2^{j-i_k} - n(i) \cdot 2^{j-i+1} + A_j(\mathcal{B}_i^{(n(i))} \cup C^{(m)}) \quad (25)$$

Consequently, the highest $n_a(j)$, is achieved by the selection of the set with the maximal value of $A_j(\mathcal{B}_i^{(n(i))} \cup C^{(m)})$, denoted:

$$\mathcal{B}_{i^*}^{(n(i))}(j) = \arg \max_{\mathcal{B}_i \in \mathcal{B}} A_j(\mathcal{B}_i^{(n(i))} \cup C^{(m)}) \quad (26)$$

This expression identifies the codeword selection at level i of the RVLC construction algorithm, $\mathcal{B}_{i^*}^{(n(i))}(j)$, which maximizes the number of available strings at level j . With regard to this result, and the previously discussed property that $\mathcal{B}_{i^*}^{(n(i))}(j_1) \neq \mathcal{B}_{i^*}^{(n(i))}(j_2)$, $j_1 \neq j_2$, we propose the following selection mechanism in the RVLC construction algorithm:

At level i , with $n_a(i) \geq n(i)$, assuming that the previously assigned codewords are $C^{(m)} = \{c^1, c^2, \dots, c^m\}$, and that the

available strings are $\mathcal{B}_A = \{b^1, b^2, \dots, b^{n(i)}\}$, assign as RVLC codewords the strings from the set:

$$\mathcal{B}_{i^*}^{(n(i))}(i+1) = \arg \max_{\mathcal{B} \in \mathcal{B}} A_{i+1}(\mathcal{B}_i^{(n(i))} \cup C^{(m)}) \quad (27)$$

where \mathcal{B} is the set of all the candidate selections at level i $\mathcal{B} = \{\mathcal{B}^{(n(i))} \mid \mathcal{B}^{(n(i))} \subseteq \mathcal{B}_A, \|\mathcal{B}^{(n(i))}\| = n(i)\}$ and $A_{i+1}(\mathcal{B}_i^{(n(i))} \cup C^{(m)})$ is the affix index of the set $\mathcal{B}_i^{(n(i))} \cup C^{(m)}$.

The proposed selection mechanism considers all the possible combinations of candidate codewords at each level of the RVLC construction algorithm, and selects the candidate combination that maximizes the number of available RVLC codewords at the next level. Note that it is possible to have multiple candidate selections $\mathcal{B}_{i^*}^{(n(i))}$, $1 \leq i \leq S$ with the same $A_{i+1}(\mathcal{B}_{i^*}^{(n(i))} \cup C^{(m)}) = \mathcal{B}_{i^*}^{(n(i))}(i+1)$. In that case one solution is to select any of them and proceed. However, a better solution is to compute $\mathcal{B}_{i^*}^{(n(i+1))}(i+2)$ for each of the candidate selections $\mathcal{B}_{i^*}^{(n(i))}$, and select the one with the highest corresponding $\mathcal{B}_{i^*}^{(n(i+1))}(i+2)$.

The previously proposed algorithm by Tsai and Wu [9] involves a simpler codeword selection mechanism. More precisely, their method is based on codeword by codeword selection at each level i , with respect to the metric called minimum repetition gap. Using the notation introduced in this paper the minimum repetition gap can be interpreted in the following manner. The minimum repetition gap of a codeword c of length i , denoted as $g(c)$, is the minimal non-negative integer such that $A_{i+g}(c, c) > 0$, where $A_{i+g}(c, c)$ denotes the cardinality of the affix set $\mathcal{A}_{i+g}(c, c)$.

TABLE I
Codeword selection at level $i=3$ $n(i)=2$. Length-3 codewords are presented in the octal format

\mathcal{B}_3	$A_4(\mathcal{B}_3)$	$n_a(4)$	$A_5(\mathcal{B}_3)$	$n_a(5)$
$\{0,1\}, \{7,6\}$	2	10	2	18
$\{0,2\}, \{7,5\}$	1	9	4	20
$\{0,3\}, \{7,4\}$	1	9	2	18
$\{0,4\}, \{7,3\}$	2	10	2	18
$\{0,5\}, \{7,2\}$	1	9	2	18
$\{0,6\}, \{7,1\}$	1	9	2	18
$\{0,7\}$	2	10	2	18
$\{1,2\}, \{6,5\}$	1	9	2	18
$\{1,3\}, \{6,4\}$	1	9	0	16
$\{1,4\}, \{6,3\}$	1	9	2	18
$\{1,5\}, \{6,2\}$	0	8	2	18
$\{1,6\}$	0	8	2	18
$\{2,3\}, \{5,4\}$	0	8	2	18
$\{2,4\}, \{5,3\}$	1	9	2	18
$\{2,5\}$	2	10	2	18
$\{3,4\}$	0	8	2	18

TABLE II
Variable length codes for English alphabet: Huffman code, RVLCs from [1], [9], and the RVLC constructed using the proposed algorithm

u	$p_U(u)$	Huffman code	Takishima's RVLC	Tsai's RVLC	Proposed RVLC
E	0.14878	001	001	000	000
T	0.09351	110	110	111	001
A	0.08833	0000	0000	0101	0100
O	0.07245	0100	0100	1010	0101
R	0.06872	0110	1000	0010	0110
N	0.06498	1000	1010	1101	1010
H	0.05831	1010	0101	0100	1011
I	0.05644	1110	1100	1011	1100
S	0.05537	0101	01100	0110	1101
D	0.04376	00010	00010	11001	01110
L	0.04124	10110	10010	10011	01111
U	0.02762	10010	01111	01110	10010
P	0.02575	11110	10111	10001	10011
F	0.02455	01111	11111	001100	11110
M	0.02361	10111	111101	011110	11111
C	0.02081	11111	101101	100001	100010
W	0.01868	000111	000111	1001001	100011
G	0.01521	011100	011101	0011100	1000010
Y	0.01521	100110	100111	1100011	1000011
B	0.01267	011101	1001101	0111110	1110111
V	0.01160	100111	01110011	1000001	1000010
K	0.00867	0001100	00011011	00111100	10000011
X	0.00146	00011011	000110011	11000011	11100111
J	0.00080	000110101	0001101011	100101001	100000010
Q	0.00080	0001101001	00011010011	0011101001	100000010
Z	0.00053	0001101000	000110100011	1001011100	1000000111
Avg. length		4.15572	4.36068	4.30678	4.25145
Efficiency		0.99161	0.94500	0.95682	0.96928

By selecting the codewords c with the lowest $g(c)$ Tsai's algorithm does contribute to the increase of $n_a(i+g)$, with the lowest g first, because $n_a(i+g)$ directly depends on $A_{i+g}(c, c)$, as in (14). However, this algorithm does not consider the effects of codeword groups, i.e. the elements $A_{i+g}(c^k, c^l)$, $c^k \neq c^l$. As can be noticed from (14), these elements have a significant impact on the RVLC efficiency; therefore the efficiency can be improved by their consideration.

Table II compares the RVLCs for the English alphabet, constructed using the proposed algorithm, and the algorithms published in [1], [9]. Due to the group-optimized codeword selection mechanism, the proposed algorithm yields an RVLC of significantly higher efficiency.

5 Conclusions

This paper considers the properties of reversible variable length codes (RVLCs) that are relevant for construction of highly efficient codes. RVLC construction algorithms generally use a Huffman code as an input. Since the codeword length

distribution of Huffman codes is the optimal prefix code distribution, RVLC construction starts by selecting the shortest RVLC codewords that have same length as the shortest Huffman codewords. The longer RVLC codewords are selected to have the same length as the Huffman codewords, provided there exist codewords of that length simultaneously satisfying both the prefix and suffix conditions. In the case that such codewords do not exist, RVLC codeword length must be increased, which causes the decrease in the RVLC code efficiency.

The number of prefix and suffix condition satisfying codewords of a certain length depends on the structure of the previously assigned (shorter) RVLC codewords. In this paper we have shown how to perform a selection of RVLC codewords of length i that maximizes the number of available (prefix and suffix condition satisfying) codewords of any length $j > i$. This selection method is utilized in the proposed RVLC construction algorithm, which, starting from the shortest codewords, assigns RVLC codewords of any length i with respect to maximizing the number of available codewords of length $i+1$. We have shown that the proposed algorithm constructs RVLCs of higher efficiency, relative to other construction algorithms in the literature.

References

- [1] Y. Takishima, M. Wada, H. Murakami, "Reversible variable length codes," *IEEE Trans. on Communications*, vol. 43, (no. 2-4), Feb.-April 1995.
- [2] J. Wen, J. Villasenor, "Reversible variable length codes for efficient and robust image and video coding," *Proc. IEEE Data Compression Conf.*, March 1998.
- [3] J. Wen, J. Villasenor, "A Class of Reversible Variable Length Codes for Robust Image and Video Coding," *Proc. IEEE International Conference on Image Processing*, vol. 2, pp. 65-68, Oct 1997.
- [4] T. M. Cover, J. A. Thomas, *Elements of Information Theory*, Wiley Series in Communications, USA, 1991.
- [5] Z. Kukorelly, K. Zeger, "New binary fix-free codes with Kraft sum $\frac{1}{2}$," *Proc. IEEE International Symposium on Information Theory*, p. 178, June 2002.
- [6] S. Yekhanin, "Sufficient conditions of existence of fix-free codes," *Proc. IEEE International Symposium on Information Theory*, p. 284, June 2001.
- [7] C. Ye, R.W. Yeung, "On fix-free codes," *Proc. IEEE International Symposium on Information Theory*, p. 428, June 2000.
- [8] C. Ye, R.W. Yeung, "Some basic properties of fix-free codes," *IEEE Transactions on Information Theory*, vol. 47, pp. 72-87, Jan. 2001.
- [9] C.W. Tsai, J.L. Wu, "On constructing the Huffman-code based reversible variable length codes," *IEEE Transactions on Communications*, vol. 49, (no. 9), Sept. 2001.
- [10] C.W. Tsai, J.L. Wu, "Modified symmetrical reversible variable-length code and its theoretical bounds," *IEEE Transactions on Information Theory*, vol. 47, pp. 2543-2548, Sept. 2001.