AN ALGORITHM FOR CONSTRUCTION OF ERROR-CORRECTING SYMMETRICAL REVERSIBLE VARIABLE LENGTH CODES

Chia-Wei Lin, Ja-Ling Wu and Jun-Cheng Chen

Communication and Multimedia Laboratory

Department of Computer Science and Information Engineering

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

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

Abstract

In this paper, we present a novel algorithm for constructing efficient symmetrical reversible variable length codes (RVLCs) with good error-correcting capabilities. Free distance is evaluated as the metric of error-correcting capabilities of the proposed RVLCs. The free distance of the proposed RVLCs is always greater than one, which results in certain improvement in symbol error rate relative to VLCs with free distance one. The proposed algorithm can apply to any VLC-involved applications to enhance the error-correcting capability, especially when the applications are critical time-constrained applications, such as real-time video conferencing over wireless channels. Experimental results show that t he proposed algorithm yields more efficient symmetrical RVLCs with error-correcting capabilities, in terms of the average codeword length, than most of existing methods in the literature.

1. Introduction

This work was partially supported by the National Science Council and the Ministry of Education of ROC under the contract No. NSC92-2622-E-002-002, NSC92-2213-E-002-023 and 89E-FA06-2-4-8.

RVLCs are accepted as an alternative to Huffman codes [1] in emerging video coding standards such as MPEG-4 [2] and H.263+ [3], to enhance the corresponding error resilience capability. RVLCs can be decoded both in the forward and backward directions, which is exploited by a decoder to enhance robustness in the presence of transmission bit errors. In 1990, Fraenkel and Klein [4] presented necessary conditions for the existence of RVLCs. Generic algorithms for the construction of RVLCs were first studied in [5] and [6]. These construction algorithms generally use a Huffman code as an input and then construct RVLCs by selecting the shortest RVLC codewords to have the same length as that of the shortest Huffman codewords. Longer RVLC codewords are selected to have the same lengths as those of the Huffman codewords, provided there are codewords of the corresponding length simultaneously satisfying both the prefix and suffix conditions. Fraenkel and Klein [4] pointed out that in searching for fix-free codes one should not be restricted to the set of Huffman codes, because there are optimum codes which cannot be obtained by the Huffman algorithm. In other words, there might be other efficient RVLC construction schemes that are not Huffman code based. Tseng and Chang [7] proposed a backtracking based algorithm that can construct symmetrical RVLCs. Its basic idea is to build up an initial solution first, and then use a bounding function and a refinement scheme to iteratively refine the solution. Lin and Wu [8] presented an efficient algorithm for constructing RVLCs based on [7], in which a greedy search heuristic has been used in the solution refinement scheme. Recent researches show that soft decoding of VLCs is advantageous over traditional VLC decoding [9]-[11]. Due to their inherent redundancy, RVLCs yield good results with soft decoding. As shown in [10], the symbol error rate performance is strongly determined by the metric called *free distance* (d_{free}). Laković and Villasenor [11] proposed an algorithm for constructing RVLCs with good error-correcting capabilities. [11] is one of the first publications that addresses the design of RVLCs with good distance properties. The so-obtained RVLCs exhibit improved performance in symbol error rate.

In wireless communications applications, where the environment possesses the following characteristics: high error rate and low bandwidth. RVLCs with good error-correcting capabilities can be applied to enhance the error-resilience capabilities. In this paper, we present a novel algorithm for constructing efficient symmetrical RVLCs with good distance properties. This paper is organized as follows. Section 2 introduces the required notations and preliminaries. In section 3, related works are presented. Section 4 presents the proposed algorithm for constructing symmetrical RVLCs with $d_{free}\geq 2$. Performance comparisons of the proposed algorithm with other known construction algorithms are given in Section 5. Section 6 concludes this paper.

2. Notations and preliminaries

Assume the information source S consists of n symbols and they are represented by (a_1, a_2,…, a_n) in decreasing order of probabilities $P(a_i)$, where $P(a_i)$ denotes the probability of the data symbol a_i.

Let C be the corresponding source code for S with n codewords $\{\boldsymbol{c}_1, \boldsymbol{c}_2, \ldots, \boldsymbol{c}_n\}$. Let l_i denote the length of codeword $\boldsymbol{c}_i$. A codeword $\boldsymbol{c}$ with k bits is represented as $c_1c_2\ldots c_k$.
Let $H(x,y)$ be the Hamming distance between two equal length binary words x and y. Let f_i be a codeword sequence constructed from C. Further, let σ denote the number of different codeword lengths in C and let the corresponding lengths be $l_1, l_2, \ldots, l_\sigma$, where $l_1 < l_2 < \ldots < l_\sigma$. Let $|f_i|$ be the bit length of the codeword sequence. The set $F_N = \{ f_i : |f_i| = N \}$ collects all possible codeword sequences of bit length N. The free distance, d_{free}, of C is defined to be the smallest possible Hamming distance between two different code sequences. That is,

$$d_{free} = \min\{H(f_i,f_j) : f_i, f_j \in F_N, i \neq j, N = 1,2,\ldots,\infty\} \qquad (1)$$

The direct evaluation of d_{free} is relatively complicated. [9] and [11] derived simple bounds by considering of other simpler VLC distance metrics called the minimum diverging distance (d_{min}), the minimum converging distance (c_{min}) and the minimum block distance (d_b). The definitions of d_{min}, c_{min} and d_b are given in [9]

It is shown in [9] that the free distance of a VLC C is bounded by

$$d_{free} \geq \min(d_b, d_{min} + c_{min}) \qquad (2)$$

If a VLC C does not have equal-length codewords, then its minimum block distance is undefined. It is shown in [11] that in the case of RVLCs

$$d_{free} \geq \begin{cases} 2 & \text{, if } d_b \text{ is undefined} \\ \min(2, d_b) & \text{, if } d_b \text{ is defined} \end{cases} \qquad (3)$$

From (3), it is clear that all RVLCs that do not have equal-length codewords have $d_{free} \geq 2$. Furthermore, if an RVLC contains codewords of equal length, $d_b \geq 2$ is the sufficient condition for $d_{free} \geq 2$.

In a full binary tree, the symmetrical children [7] of a codeword $\boldsymbol{c}$, *children*($\boldsymbol{c}$), are defined by all of the first symmetrical codewords on paths from the codeword $\boldsymbol{c}$ to leaf codewords. Let CL be a codeword list and CL_d is also a codeword list in which the constraint $d_b \geq d$ holds for all its codewords. Define *children*($\boldsymbol{c}$, d) to be a subset of the symmetrical children of a symmetrical codeword $\boldsymbol{c}$ in which the constraint $d_b \geq d$ holds for all the children in the subset. Define *children*($\boldsymbol{c}$, CL_d, d) to be a subset of symmetrical children of $\boldsymbol{c}$ that the union of the subset and CL_d is still a codeword list in which the constraint $d_b \geq d$ holds for all its codewords.

3. Related works

Since the proposed RVLC construction algorithm is derived on the basis of [6] and [8], we first review these RVLC construction approaches, briefly.

3.1.Tsai and Wu's Symmetrical RVLC Algorithm [6]

This algorithm uses a new codeword selection mechanism to select candidate codewords in each level. We explain some terms in relation to their construction algorithm, first.

Let *CAND*(l) denote the set of symmetrical candidate codewords of length l on a full binary tree. Then, we have

$$|CAND(l)| = 2^{\lfloor (l+1)/2 \rfloor} \qquad (4)$$

Secondly, let $p(l)$ denote the total number of symmetrical codewords, located at level l, which violating the affix condition owing to some codewords positioned, in the path from the root to the codewords, at the l-th level that have been selected as target codewords. By taking the difference of these two numbers, the total number of available candidate codewords, $avail(l)$, at level l is

$$avail(i) = |CAND(l)| - p(l) \quad (5)$$

The codeword selection mechanism, for symmetrical RVLCs, in each level is based on the ordering of candidate codewords in accordance with their maximum symmetrical suffix lengths (MSSLs) [6]. These codeword arrangements always produce more usable candidate codewords for subsequent levels. Thus, MSSL can be treated as an indication of the amount of usable candidate codewords for subsequent levels. Tsai and Wu's algorithm can be summarized as follows.

Step1: Initialize the bit length vector of the target symmetrical RVLC,
$n_{rev}(i) = n(i)$,
where $n(i)$ is the bit length vector of a list of given Huffman codes.

Step2: Calculate $avail(i)$ and $MSSL$ for each candidate codeword, and arrange these codewords based on
the increasing order of $MSSL$s.
If $n_{rev}(i) \leq avail(i)$, $n_{rev}(i)$ is not changed. Otherwise,
$n_{rev}(i+1) = n_{rev}(i+1) + n_{rev}(i) - avail(i)$,
$n_{rev}(i) = avail(i)$.
Assign the candidate codewords as target codewords in sequence.

Step3: Repeat Step2 until every codeword has been assigned a symmetrical RVLC codeword.

3.2 Lin and Wu's Generic Symmetrical RVLC algorithm [8]

Suppose we have selected *n* symmetrical codewords and they are represented by a codeword list *CL*. If we replace codeword $\boldsymbol{c}_i$ with its symmetrical children, we will obtain a new codeword list *CL'*. The main idea of [8] is that if the average codeword length of this new list *CL'* is smaller than that of the original list *CL*, then a *replacement* operation is carried out and the target list is updated with the new list. Lin's algorithm can be summarized as follows.

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

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

Step3: If any codeword in the target list satisfies the condition of *replacement* [7], replace the codeword with one of its symmetrical children which results in the smallest average codeword length.

Step4: Repeat Step2 and Step3 until the size of the target list is greater than or equal to the total number of codewords.

4. The proposed algorithm

In this algorithm, a greedy search heuristic is adopted as in [7] and a new codeword replacement scheme with the constraint that inhibiting $d_b = 1$ is added. Before explaining the details of the proposed algorithm, we first define some terms employed in our algorithm.

Let *d* be the desired minimum block distance of the resultant symmetrical RVLCs. Then, *children*($\boldsymbol{c}$,*d*), and *children*($\boldsymbol{c}$, CL_d, *d*) can be found as follows..

children($\boldsymbol{c}$,*d*): Sort *children*($\boldsymbol{c}$) based on the increasing order of codeword length and MSSL. Scan from the beginning of *children*($\boldsymbol{c}$). Reserve the codeword with smaller MSSL when there are two codewords possess relative Hamming distance less than *d*, and discard the codeword with larger MSSL.

children($\boldsymbol{c}$, CL_d, *d*): Sort *children*($\boldsymbol{c}$,*d*) based on the increasing order of codeword length and MSSL. Scan from the beginning of *children*($\boldsymbol{c}$,*d*). Reserve the codeword in *children*($\boldsymbol{c}$,*d*) if that codeword can be added into *CL* without violating the constraint $d_b \geq d$ in *CL*.

Define $R(\boldsymbol{c}, CL_d, d)$ to be the replacement of a codeword $\boldsymbol{c}$ in the codeword list CL_d. $R(\boldsymbol{c}, CL_d, d)$ can be found as follows: Let $R(\boldsymbol{c}, CL_d, d) = children(\boldsymbol{c}, CL_d, d)$, first. Then, for each symmetrical codeword at level i (i>1): Add the available symmetrical codewords to $R(\boldsymbol{c}, CL_d, d)$ if both the affix-free condition and the $d_b \geq d$ condition are satisfied.

THE PROPOSDED ALGORITHM:

Step1: Assign the initial codeword list ("1","00","010","0110",...) to the target list, *T-List*.

Step2: If any codeword $\boldsymbol{c}$ in *T-List* satisfies the condition of replacement, replace the codeword $\boldsymbol{c}$ with $R(\boldsymbol{c}$, *T-List*, 2) that results in the smallest average codeword length. If the number of codewords in *T-List* is more than n, the number of source symbols, keep the first n codewords in *T-List* and discard the others.

Step3: Repeat Step2 until there is no codeword in *T-List* satisfying the condition of replacement.

We illustrate the algorithm by an example. Six symbols with probabilities (0.286, 0.214, 0.143, 0.143, 0.071) are given. The height of the full binary tree for finding the symmetrical children of this example is 6. Table 1 shows the process of the example. In the first iteration, *T-List* = ("1","00","010","0110","01110","011110"). The average codeword length is 2.856. The respective average codeword lengths after each codeword in *T-List* is replaced are 2.714, 3.142, 3.285, 2.856, 2.856 and 2.856. Then codeword "1" is replaced since "1" satisfies the condition of replacement and the replacement of "1" results in the smallest average codeword length in this iteration. Table 2 lists some temporary results obtained in the first iteration. In the second iteration, *T-List* = ("00","11","010","101","0110","1001"). Finally, there are no other codewords in *T-List* satisfying the condition of replacement and the algorithm ends. The resultant code is ("00","11","010","101","0110","1001").

Table 1: An example of the proposed algorithm, where $P_U(U)$ denotes the probability of the source symbol U.

U	$P_U(U)$	Iteration #1	Iteration #2
a1	0.286	1	00
a2	0.214	00	11
a3	0.143	010	010
a4	0.143	0110	101
a5	0.143	01110	0110
a6	0.071	011110	1001
Average codeword length		2.856	2.714

The proposed algorithm produces symmetrical RVLCs with d_{free}≥2 by introducing the constraint that d_b = 1 is not allowed. It should be pointed out that symmetrical RVLCs with d_{free}≥3 cannot be surely obtained by the proposed algorithm.

Table 2: Some temporary results of the proposed algorithm.

Iteration #1				
c	**children(*c*)**	**children(*c, 2*)**	**children(*c*, *T-List*, 2)**	**R(*c*,*T-List*, 2)**
"1"	**"11", "101", "1001", "10001", "100001"**	**"11", "101", "1001", "10001", "100001"**	**"11", "101", "1001", "10001", "100001"**	**"11", "101", "1001", "10001", "100001"**
"00"	**"000", "00100", "001100"**	**"000", "00100", "001100"**	**"00100", "001100"**	**"0000", "00100", "001100"**

5. Experimental results

The proposed algorithm has 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. In the following, the heights of full binary trees for finding the symmetrical children (ranged from 12 to 20) depend on the number of symbols of the information source. In the following comparisons, the results of [6], [8], [11] and the proposed algorithm are obtained as follows: Given a probability distribution, [6] and [11] should construct a Huffman code first and then begins code construction from the Huffman code. In contrast to [6] and [11], [8] and the proposed algorithm take the same probability distribution and construct symmetric RVLCs. In other words, the code construction of [11] is based on [6] while the proposed algorithm is based on [7] and [8]. In order to make a more fair comparison, instead of using Huffman codes, we take the symmetrical RVLCs generated from [8] as the initial codes of [11]. In this way, both [11] and the proposed algorithm start their code constructions from the same initial symmetrical RVLC, which is ("1","00","010","0110",...).

Table 3 respectively lists various symmetrical RVLCs constructed by [6], [8], [11] and the proposed algorithm for the English alphabet set. [11] starts code construction from a given initial Huffman code and the symmetric RVLC generated by [8]. It is clear that the proposed algorithm produces more efficient symmetrical RVLCs with $d_{free}\geq 2$. It is interesting to note that the code construction of [11] with initial codes taken from [8] performs better than its original setting, which suggests that Huffman codes are not the best initial codes for [11].

Table 3: Symmetrical RVLCs for the English alphabet set: symmetrical RVLC from [6], symmetrical RVLC from [8], symmetrical RVLC with $d_{free} \geq 2$ from [11], symmetrical RVLC with $d_{free} \geq 2$ from [11] with the initial code generated by [8] and symmetrical RVLC with $d_{free} \geq 2$ from the proposed algorithm.

U	$p_U(U)$	[6]	[8]	[11]	[11] from [8]	The Proposed RVLC
E	**0.14878**	**010**	**010**	**010**	**010**	**11**
T	**0.09351**	**101**	**101**	**101**	**101**	**010**
A	**0.08833**	**0110**	**000**	**0110**	**0110**	**101**
O	**0.07245**	**1001**	**111**	**1001**	**1001**	**0110**
R	**0.06872**	**0000**	**0110**	**0000**	**0000**	**1001**
N	**0.06498**	**1111**	**1001**	**1111**	**1111**	**0000**
H	**0.05831**	**01110**	**01110**	**01110**	**01110**	**01110**
I	**0.05644**	**10001**	**10001**	**10001**	**10001**	**10001**
S	**0.05537**	**00100**	**00100**	**00100**	**00100**	**00100**
D	**0.04376**	**11011**	**11011**	**11011**	**11011**	**011110**
L	**0.04124**	**011110**	**011110**	**011110**	**011110**	**100001**
U	**0.02762**	**100001**	**100001**	**100001**	**100001**	**001100**
P	**0.02575**	**001100**	**001100**	**001100**	**001100**	**0111110**
F	**0.02455**	**110011**	**110011**	**110011**	**110011**	**1000001**
M	**0.02361**	**0111110**	**0111110**	**0111110**	**0111110**	**0010100**
C	**0.02081**	**1000001**	**1000001**	**1000001**	**1000001**	**0001000**
W	**0.01868**	**0010100**	**0010100**	**0011100**	**0010100**	**01111110**
G	**0.01521**	**1101011**	**0011100**	**1100011**	**1100011**	**10000001**
Y	**0.01521**	**0011100**	**1100011**	**0001000**	**0001000**	**00111100**
B	**0.01267**	**1100011**	**1101011**	**1110111**	**1110111**	**00011000**
V	**0.01160**	**0001000**	**01111110**	**01111110**	**01111110**	**011111110**
K	**0.00867**	**1110111**	**10000001**	**10000001**	**10000001**	**100000001**
X	**0.00146**	**01111110**	**00111100**	**00111100**	**00111100**	**001010100**
J	**0.00080**	**011111110**	**11000011**	**011111110**	**11000011**	**001101100**
Q	**0.00080**	**0111111110**	**011111110**	**0111111110**	**011111110**	**000101000**
Z	**0.00053**	**1000000001**	**100000001**	**1000000001**	**100000001**	**0111111110**
Avg. length		**4.60728**	**4.46463**	**4.62757**	**4.62543**	**4.56725**

Table 4 lists the results of the Canterbury Corpus file set compressed by using [6], [8], [11] and the proposed algorithm. The corresponding Huffman codes and symmetric RVLCs [8] are obtained based on the probability distributions. (The probability distribution is obtained by calculating the frequencies of occurrence of source symbols of the source file.) Likewise, [11] starts code construction from both the given initial Huffman codes and the symmetric RVLCs generated by [8]. Once again, the proposed algorithm produces more efficient symmetrical RVLCs with $d_{free} \geq 2$ than the other approaches. Once again, [11] with modified initial codes performs better than its original version in 7 out of total 11 cases. Due to its iterative and data symbol probability dependent nature, the complexity analysis of the proposed algorithm is still under developing. To show its performance in practical usage, Table 4 also lists the number of iterations required to generate the codes.

Table 4: The average codeword lengths of various symmetrical RVLCs for the Canterbury Corpus file set: symmetrical RVLC from [6], symmetrical RVLC from [8], symmetrical RVLC with $d_{free} \geq 2$ from [11], symmetrical RVLC with $d_{free} \geq 2$ from [11] with an initial code generated by [8] and symmetrical RVLC with $d_{free} \geq 2$ from the proposed algorithm.

filename	Number of codewords	[6]	[8]	[11]	[11] from [8]	The Proposed RVLC	Number of iterations
asyoulik.txt	**68**	**5.27886**	**5.21025**	**5.41179**	**5.38806**	**5.31472**	**4**
alice29.txt	**74**	**5.01398**	**4.93155**	**5.21075**	**5.1012**	**5.02553**	**4**
xargs.l	**74**	**5.39863**	**5.33995**	**5.52851**	**5.51597**	**5.44334**	**4**
grammar.lsp	**76**	**5.04757**	**5.01773**	**5.24805**	**5.24966**	**5.13222**	**4**
plrabn12.txt	**81**	**4.94473**	**4.89526**	**4.98592**	**5.0117**	**4.98392**	**4**
lcet10.txt	**84**	**5.12232**	**5.01682**	**5.22562**	**5.10798**	**5.10747**	**4**
cp.html	**86**	**5.85839**	**5.81173**	**6.09414**	**6.11905**	**5.97269**	**4**
fields.c	**90**	**5.47596**	**5.46331**	**5.69381**	**5.71166**	**5.61578**	**4**
ptt5	**159**	**1.77735**	**1.75991**	**1.80165**	**1.79498**	**1.79498**	**3**
sum	**255**	**6.10683**	**6.03917**	**6.32367**	**6.28533**	**6.26198**	**6**
kennedy.xls	**256**	**4.25681**	**4.21507**	**4.3712**	**4.33206**	**4.3282**	**6**

6. Conclusion

The optimal design of RVLC with good distance properties and codeword length efficiency is still an open problem. In this paper, we present a greedy algorithm for constructing symmetrical RVLCs with good error-correcting capabilities. The proposed algorithm is suitable for any VLC-involved applications to enhance the error-correcting capability of critical time-constrained applications. It should be pointed out that symmetrical RVLCs with $d_{free} \geq 3$ cannot be surely obtained by the proposed algorithm. The major contribution of the proposed algorithm is that it yields more efficient symmetrical RVLCs with the same free distance than other known algorithms.

7. References

[1] Huffman, D. A. A method for the construction of minimum redundancy codes. Proceedings of IRE, 40 (1952) 1098–1101.

[2] ISO/IEC 14496 Coding of Audio-visual Objects. International Organization for Standardization, Geneva (1998).

[3] ITU-T H.263 Video Coding for Low Bit Rate Communication. International Telecommunications Union, Geneva (1995).

[4] Fraenkel, A. S. and Klein, S. T. Bidirectional Huffman coding. Comput. J., 33 (1990) 296–307.

[5] Takishima, Y., Wada, M. and Murakami, H. Reversible variable length codes. IEEE Trans. Commun., 43 (1995) 158–162.

[6] Tsai, C.-W. and Wu, J.-L. Modified symmetrical reversible variable-length code and its theoretical bounds. IEEE Trans. Inform. Theory, 47 (2001) 2543–2548.

[7] Tseng, H. -W. and Chang, C. -C. Construction of Symmetrical Reversible Variable Length Codes Using Backtracking. Comp. J., 46 (2003) 100-105.

[8] Lin, C. –W. and Chuang, Y. –J. and Wu, J. –L. Generic construction algorithms for symmetrical and asymmetrical RVLCs. Proceedings of ICCS ’02, Singapore, 25-28 November (2002), vol. 2, pp.~ 968–972.

[9] Buttigieg, V. Variable-length error-correcting codes. Ph.D. dissertation, Univ. of Manchester, Manchester, U.K. (1995).

[10] Bauer, R. and Hagenauer, J. On variable length codes for iterative source/channel-decoding. In Proc. DCC ’01, Snowbird, UT, 27–29 March (2001), pp.~ 273–282, IEEE Computer Society Press, Los Alamitos, CA.

[11] Laković, K. and Villasenor, J. On Design of Error-Correcting Reversible Variable Length Codes. IEEE Commun. Letters, 6 (2002) pp. 337–339.