

Yang-Kieffer Algorithms and 1-D Run-length Encoding: Compression Performance Comparison ¹

En-hui Yang

Da-ke He ²

Abstract — It is first shown that for the class of semi-Markov sources, the modified 1-D Run-length Encoding (RLE) algorithm outperforms any finite-order arithmetic coding algorithm in terms of compression rate. This result justifies the popular choice of RLE in the application of video compression. We then show that Yang-Kieffer algorithms are indeed superior to RLE. It is proved that the worst case redundancies of Yang-Kieffer algorithms against any RLE algorithm among all individual sequences of length n are upper bounded by $d \log \log n / \log n$, where d is a constant.

I. INTRODUCTION

In the 1-D Run-length encoding (RLE) algorithm, the sequence to be compressed is parsed into a sequence of symbol-run pairs (the definition of symbol-run pair is left until Section II), which are then encoded by Huffman coding. The modified RLE algorithm substitutes two-step multi-level arithmetic coding [1] for Huffman coding and can generally improve compression performance. RLE has been a dominating player in lossless encoding in the application of video compression. As a matter of fact, the RLE algorithm is adopted as the lossless coding method in nearly all current mainstream video compression standards, such as MPEG 1, MPEG 2, and DV(SMPTE 314M) [2].

However, it is believed that RLE is efficient only for certain types of data sequences such as those encountered in the application of video compression. In the pursuit of more efficient universal compression algorithms which can be applied to a variety of applications including video compression, Yang and Kieffer [3] recently proposed three universal compression algorithms, which are now called by others [4, 5] Yang-Kieffer algorithms or simply YK algorithms. It was proved in [3] that Yang-Kieffer algorithms are all universal in the sense that they can achieve asymptotically the entropy rate of any stationary, ergodic source, and outperform any finite-state data compression algorithms.

In this paper, we aim at comparing the compression performance of Yang-Kieffer algorithms and that of RLE for any individual sequences and any types of sources. In Section II, we show that for semi-Markov sources, the modified RLE algorithm outperforms any finite-order arithmetic coding algorithm in terms of compression rate. As a generalization of renewal sources [6], the class of semi-Markov sources is quite broad; quantized coefficients in video compression may be approximately modeled by the model of semi-Markov sources. This result justifies the choice of RLE over arithmetic coding

in the application of video compression; it also implies that the performance comparison of Yang-Kieffer algorithms and RLE can not be derived by using finite-order arithmetic coding as a benchmark. In Section III, we show that Yang-Kieffer algorithms are indeed superior to RLE. Experimental results are presented and conclusions are drawn in Section IV.

Notation: Throughout this paper, let \mathcal{A} be our source alphabet with cardinality greater than or equal to 2. Let \mathcal{N} be the set of all positive integers. Let \mathcal{A}^* be the set of all finite strings drawn from \mathcal{A} , including the empty string λ , and \mathcal{A}^+ the set of all finite strings of positive length from \mathcal{A} . The notation $|\mathcal{A}|$ stands for the cardinality of \mathcal{A} , and for any $x \in \mathcal{A}^*$, $|x|$ denotes the length of x . For any positive integer n , \mathcal{A}^n denotes the set of all sequences of length n from \mathcal{A} . Similar notation will be applied to other finite sets and finite strings drawn from them. To avoid possible confusion, a sequence from \mathcal{A} is sometimes called an \mathcal{A} -sequence.

II. RLE AND FINITE-ORDER ARITHMETIC CODING

In this section, we compare the compression performance of RLE with that of finite-order arithmetic coding for the class of semi-Markov sources.

Associated with any sequence $x = x_1 x_2 \cdots x_n$ from \mathcal{A} , there is a sequence of symbol-run pairs $\{(y_j, r_j)\}_{j=1}^m$, $m \leq n$. Each (y_j, r_j) corresponds to a consecutive run of symbol y_j of length r_j in x . For example, let $\cdots a_0 a_1 a_1 a_1 a_2 \cdots$, $a_0, a_1, a_2 \in \mathcal{A}$ be a portion of sequence x ; the a_1 's run of length 4 above corresponds to a symbol-run pair $(a_1, 4)$.

Given a source $\{X_i\}_{i=1}^\infty$, there is a sequence of symbol-run pairs $\{(Y_j, R_j)\}_{j=1}^\infty$, in which Y_j is a \mathcal{A} -valued random variable and R_j is a \mathcal{N} -valued random variable representing the run length of Y_j . We say $\{X_i\}_{i=1}^\infty$ is a *semi-Markov source* if

- (1) $\{Y_j\}_{j=1}^\infty$ is an aperiodic Markov chain with transition probability matrix $[p_{a_i a_j}]_{a_i, a_j \in \mathcal{A}}$ satisfying $p_{a_i a_i} = 0$ for any $a_i \in \mathcal{A}$;
- (2) There is a memoryless channel $Q(\cdot|\cdot) : \mathcal{A} \rightarrow \mathcal{N}$ such that $R_2 R_3 \cdots R_n \cdots$ is the output of the channel in response to the input $Y_2 Y_3 \cdots Y_n \cdots$; and
- (3) Given Y_1, R_1 is independent of $\{(Y_j, R_j)\}_{j=2}^\infty$.

Given $[p_{a_i a_j}]$ and $Q(\cdot|\cdot)$, $\{X_i\}_{i=1}^\infty$ can be made stationary by choosing an appropriate initial distribution of (Y_1, R_1) .

A distribution Q is said to be a *pseudo geometrical distribution* if there exists a positive integer j , and constants $C > 0$ and $0 \leq p < 1$ such that $Q(i) = cp^i$ for all $i \geq j$. It is easy to see that a semi-Markov source $\{X_i\}_{i=1}^\infty$ cannot be modeled by any finite-order Markov chain if there exists an $a \in \mathcal{A}$ such that $Q(\cdot|a)$ is not a pseudo geometrical distribution. For such semi-Markov sources, a k th-order, $0 < k < \infty$, arithmetic coding algorithm will only be suboptimal in the sense that their best possible compression rate in bits per symbol is $\mathcal{H}(\tilde{X}_{k+1}|\tilde{X}_k \cdots \tilde{X}_1)$, where $\{\tilde{X}_i\}_{i=1}^\infty$ is the stationary mean of source $\{X_i\}_{i=1}^\infty$. $\mathcal{H}(\tilde{X}_{k+1}|\tilde{X}_k \cdots \tilde{X}_1)$ is strictly above the source entropy rate \mathcal{H}_X .

¹This work was supported in part by the Natural Sciences and Engineering Research Council of Canada under Grant RGPIN203035-98, by the Communications and Information Technology Ontario, by the Premier's Research Excellence Award and by the Canada Research Chairs Program.

²The authors are with the Department of Electrical and Computer Engineering, University of Waterloo, 200 University Avenue West, Waterloo, Ontario, Canada N2L 3G1. E-mails: ehyang@bbcr.uwaterloo.ca, dhe@bbcr.uwaterloo.ca

In the RLE algorithm, $\{(Y_j, R_j)\}_{j=1}^{\infty}$ is encoded by using Huffman coding. However, the RLE algorithm may not achieve the source entropy rate either. The modified RLE algorithm uses a two-step lossless coding strategy to encode the sequence of symbol-run pairs $(y_1, r_1), \dots, (y_m, r_m)$, $1 \leq m \leq n$, associated with sequence $x = x_1 x_2 \dots x_n$ to be compressed: 1) y_{j+1} of a pair (y_{j+1}, r_{j+1}) is first encoded by using adaptive first-order arithmetic coding with estimated probability $p_a(y_{j+1}|y_j)$; 2) r_{j+1} is then encoded by arithmetic coding with probability $q_r(r_{j+1}|y_{j+1})$. In both steps, esp. in step 2) to handle unbounded alphabet, multi-level arithmetic codes [1] can be employed.

Let $r^{rle}(x)$ be the compression rate in bits per symbol resulting from using the modified RLE algorithm to compress sequence x . Let $r_{ka}^*(x)$ be the best possible compression rate in bits per symbol resulting from using any k th-order arithmetic coding algorithm to compress x . The following theorem is given without proof in this paper to conclude this section.

Theorem 1 *Let $X = \{X_i\}_{i=1}^{\infty}$ be a semi-Markov source with alphabet \mathcal{A} . Assume that $X = \{X_i\}_{i=1}^{\infty}$ satisfies the property that with probability one, there exists an $a \in \mathcal{A}$ such that $Q(\cdot|a)$ is not a pseudo geometrical distribution. Then*

$$r^{rle}(X_1 X_2 \dots X_n) < r_{ka}^*(X_1 X_2 \dots X_n)$$

with probability one as $n \rightarrow \infty$.

III. YK ALGORITHMS AND RLE

Now we compare the compression performance of YK algorithms with that of any RLE algorithm. As in previous sections, the RLE algorithm refers to the one using Huffman codes and the modified RLE algorithm refers to the modified version using multi-level arithmetic codes. Let $x \in \mathcal{A}^n$ be the sequence to be compressed. Let $r^{rle}(x)$ be the compression rate in bits per symbol resulting from using the RLE algorithm to compress x . $r^{rle}(x)$ is defined as in Section II. On the side of YK algorithms, for simplicity we choose the sequential algorithm proposed in [3]. However, all results and derivations of this paper apply equally well to other versions of YK algorithms. In the sequential algorithm, the greedy grammar transform parses x into a sequence of phrases. Let $x_i \dots x_{i+j}$ be one of the phrases. We encode $x_i \dots x_{i+j}$ sequentially by using an arithmetic code with estimated probability $p(x_i \dots x_{i+j})$. For the details of YK algorithms and greedy grammar transform, please see [3]. Let $r^s(x)$ be the compression rate in bits per symbol resulting from using the sequential algorithm to compress x . It is proved in [3] that the sequential algorithm encodes x into $nr^s(x)$ bits, which are bounded by

$$nr^s(x) \leq H_p(x) + 2t + |\mathcal{A}|, \quad (1)$$

where t is the number of parsed phrases and $H_p(x)$ denotes the unnormalized empirical entropy of the sequence of parsed phrases, i.e.,

$$H_p(x) = \sum_{\beta \in \mathcal{S}(j_t) \cup \mathcal{A}} c(\beta) \log \left(\frac{t}{c(\beta)} \right), \quad (2)$$

In (2), $\mathcal{S}(j_t)$ is the final variable set of the generated grammar and $c(\beta)$, for each $\beta \in \mathcal{S}(j_t) \cup \mathcal{A}$, denotes the number of times the \mathcal{A} -sequence represented by β appears in the sequence of

parsed phrases $x_1, x_2 \dots x_{n_2}, \dots, x_{n_{t-1}+1} \dots x_{n_t}$. For convenience, we use u_i , $1 \leq i \leq t$ to represent the i th parsed phrase, i.e. $u_i = x_{n_{i-1}+1} \dots x_{n_i}$, where $n_0 = 0$ and $n_t = n$.

First we analyze the difference between $r^s(x)$ and $r^{rle}(x)$. Let

$$R_n^{s,rle} \triangleq \max_{x \in \mathcal{A}^n} [r^s(x) - r^{rle}(x)].$$

The quantity $R_n^{s,rle}$ is called the *worst case redundancy* of the sequential algorithm against the RLE algorithm. The following theorem gives the upper bound of $R_n^{s,rle}$.

Theorem 2 *There is a constant d_1 , which depends only on \mathcal{A} , such that*

$$R_n^{s,rle} \leq d_1 \frac{\log \log n}{\log n}.$$

Proof: Figure 1 shows an example where x is parsed by three different parsers. In Case (a), x is parsed by the greedy grammar transform [3] into a sequence of phrases u_1, u_2, \dots, u_t , where $\sum_{i=1}^t |u_i| = |x| = n$; in Case (b), x is parsed into a sequence of symbol-run pairs $(a_1, l_1), (a_2, l_2), \dots, (a_k, l_k)$, where $\sum_{j=1}^k l_j = |x|$; in Case (c), x is first parsed into u_1, u_2, \dots, u_t and then each u_i , $1 \leq i \leq t$ is parsed into a sequence of symbol-run pairs $(a_{i,1}, l_{i,1}), (a_{i,2}, l_{i,2}), \dots, (a_{i,k_i}, l_{i,k_i})$, where $\sum_{j=1}^{k_i} l_{i,j} = |u_i|$. It is easy to see that in the worst case, every u_i breaks a run of symbols, i.e. u_i 's last run-length parsing (a_{i,k_i}, l_{i,k_i}) and u_{i+1} 's first run-length parsing $(a_{i+1,k_1}, l_{i+1,k_1})$ are a single parsing in Case (b). For example, in Figure 1, (a_{j+1}, l_{j+1}) is broken into two symbol-run pairs, $(a_{i,3}, l_{i,3})$ and $(a_{i+1,1}, l_{i+1,1})$. By removing $(a_{i,1}, l_{i,1})$ and (a_{i,k_i}, l_{i,k_i}) for any $1 \leq i \leq t$, we find that every parsed symbol-run pair in Case (c) has an exact one-to-one match in the parsed sequence of symbol-run pairs $(a_1, l_1), (a_2, l_2), \dots, (a_k, l_k)$ in Case (b). In view of these, we propose the following algorithm to encode each u_i ,

Step 1: Encode $|u_i|$ by using the Elias' universal doubly compound representation of integers [7], which will be called the Elias code in the rest of this paper.

Step 2: Encode the first pair $(a_{i,1}, l_{i,1})$ as follows: $a_{i,1}$ is encoded by its binary representation; $l_{i,1}$ is encoded by the Elias code.

Step 3: Encode $(a_{i,2}, l_{i,2}), \dots, (a_{i,k_i-1}, l_{i,k_i-1})$ as the corresponding RLE algorithm does.

Step 4: Instead of encoding (a_{i,k_i}, l_{i,k_i}) , use the RLE algorithm to encode (a_{i,k_i}, l') , where (a_{i,k_i}, l') , $l' \geq l_{i,k_i}$, has the minimum codeword length among all pairs $(a_{i,k}, l)$ with $l \geq l_{i,k_i}$.

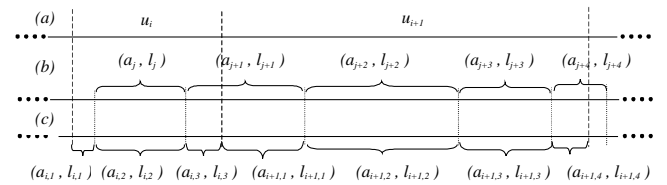


Fig. 1: A segment of a sequence parsed by (a) the sequence parser in the greedy grammar transform; (b) a run-length parser; (c) (a) followed by (b).

By contrast, the RLE algorithm encodes x by first parsing x into the sequence $(a_1, l_1), (a_2, l_2), \dots, (a_k, l_k)$ as in Case (b). From the discussion above, we know that the number of bits allocated in Step 3 to encode $(a_{i,2}, l_{i,2}), \dots, (a_{i,k_i-1}, l_{i,k_i-1})$ is exactly the same as that is needed in the RLE algorithm. In Step 4, the number of bits allocated is no more than what the RLE algorithm uses to encode the corresponding parsed symbol-run pair which may have been broken into two parts in Case (c). Thus the proposed algorithm encodes x into no more than $nr^{rle}(x)$ bits plus the number of bits introduced in Step 1 and Step 2, which we shall call the *overhead*. We use $V(u_i)$ to denote the overhead introduced by encoding u_i using the proposed algorithm.

To continue the proof, we need a few more definitions. Let z denote a positive integer to be coded. $L_E(z)$ is defined as the codeword length function of the Elias code, i.e. $L_E(z) = 1 + \log z + 2 \log(1 + \log z)$. Similarly, we use $L_{RLE}(a, l)$ to denote the codeword length function of the RLE algorithm, where (a, l) , $a \in \mathcal{A}$ and $l \in \mathcal{N}$, is a symbol-run pair.

For any sequence $y = y_1 \cdots y_m \in \mathcal{A}^m$, $m \in \mathcal{N}$, we can parse y into the following sequence of symbol-run pairs, $(b_1, g_1), (b_2, g_2), \dots, (b_k, g_k)$, where $b_j \in \mathcal{A}$, $g_j \in \mathcal{N}$, $1 \leq j \leq k$ and $\sum_{j=1}^k g_j = m$. Now we use the proposed algorithm to encode $y = y_1 \cdots y_m$ as if we are encoding a u_i . Since both the Elias code and the Huffman code used above are prefix codes, it is not hard to see we have constructed an uniquely decodable code for each $y = y_1 \cdots y_m$. Thus by McMillan's inequality [8],

$$\sum_{m=1}^{\infty} \sum_{y \in \mathcal{A}^m} [2^{-L_E(m) - \log |\mathcal{A}| - L_E(g_1) - \Delta}] \leq 1,$$

where

$$\Delta \triangleq \sum_{j=2}^{k-1} L_{RLE}(b_j, g_j) + \min_{g' \geq g_k} L_{RLE}(b_k, g').$$

Let p^* be the probability distribution on \mathcal{A}^+ such that for any positive integer m and any $y^m = y_1 \cdots y_m \in \mathcal{A}^m$,

$$p^*(y^m) = Q 2^{-L_E(m) - \log |\mathcal{A}| - L_E(g_1) - \Delta}. \quad (3)$$

In the above equality, the constant $Q \geq 1$ is selected so that p^* is the probability distribution on \mathcal{A}^+ . Note that $L_E(m)$, $\log |\mathcal{A}|$ and $L_E(g_1)$ are simply the overhead, and we write $V(y^m) = L_E(m) + \log |\mathcal{A}| + L_E(g_1)$. Since each u_i is a sequence from \mathcal{A} , then it makes sense to write $p^*(u_i)$ for any $1 \leq i \leq t$. From (3) and the discussion above, it then follows that

$$\begin{aligned} \sum_{i=1}^t -\log p^*(u_i) &\leq nr^{rle}(x) - t \log Q + \sum_{i=1}^t V(u_i) \\ &\stackrel{1)}{\leq} nr^{rle}(x) + \sum_{i=1}^t V(u_i). \end{aligned} \quad (4)$$

In the above, the inequality 1) is due to the fact that $Q \geq 1$. The total overhead $\sum_{i=1}^t V(u_i)$ can be easily bounded by,

$$\sum_{i=1}^t V(u_i) = t \log |\mathcal{A}| + \sum_{i=1}^t L_E(|u_i|) + \sum_{i=1}^t L_E(l_{i,1})$$

$$\begin{aligned} &\stackrel{1)}{\leq} t \log |\mathcal{A}| + 2 \sum_{i=1}^t L_E(|u_i|) \\ &\leq (2 + \log |\mathcal{A}|)t + 2t \log \left(\frac{n}{t}\right) + \\ &\quad 4t \log \left[\frac{1}{t} + \log \left(\frac{n}{t}\right)\right]. \end{aligned} \quad (5)$$

The above inequality 1) is due to the fact that $l_{i,1} \leq |u_i|$. The last inequality is from repeatedly applying Jensen's inequality [9, Theorem 2.6.2, pp. 25]. In view of the information inequality [9, Theorem 2.6.3, pp. 26],

$$H_p(x) \leq \sum_{i=1}^t -\log p^*(u_i)$$

which, together with (1), (4) and (5), implies

$$\begin{aligned} nr^s(x) &\leq nr^{rle}(x) + |\mathcal{A}| + (4 + \log |\mathcal{A}|)t + 2t \log \left(\frac{n}{t}\right) + \\ &\quad 4t \log \left[\frac{1}{t} + \log \left(\frac{n}{t}\right)\right]. \end{aligned} \quad (6)$$

Dividing both sides of (6) by n and applying Lemma 5 in [3], we get

$$R_n^{s,rle} \leq O\left(\frac{1}{n}\right) + O\left(\frac{1}{\log n}\right) + O\left(\frac{\log \log n}{\log n}\right) + O\left(\frac{\log \log \log n}{\log n}\right).$$

This completes the proof of Theorem 2.

Next we analyze the difference between $r^s(x)$ and $r^{rta}(x)$. Similarly, we define

$$R_n^{s,rta} \triangleq \max_{x \in \mathcal{A}^n} [r^s(x) - r^{rta}(x)].$$

The quantity $R_n^{s,rta}$ is called the *worst case redundancy* of the sequential algorithm against the modified RLE algorithm.

Theorem 3 *There is a constant d_2 , which depends only on \mathcal{A} , such that*

$$R_n^{s,rta} \leq d_2 \frac{\log \log n}{\log n}.$$

Proof: In the proof of Theorem 2, the codeword length function L_{RLE} is used to construct p^* . In the current proof, we use instead coding probabilities to construct p^* . Assume that the exact arithmetic is used. Let p_a and q_r be the conditional probability functions used by the modified RLE algorithm in Step 1) and 2) respectively.

Let \tilde{p}^* be the probability distribution on \mathcal{A}^+ such that for any positive integer m and any $y^m = y_1 \cdots y_m \in \mathcal{A}^m$,

$$\begin{aligned} \tilde{p}^*(y^m) &= Q' |\mathcal{A}|^{-1} m^{-3} p_a(b_k | b_{k-1}) \cdot \\ &\quad \max_{g' \geq g_k} q_r(g' | b_k) \prod_{j=2}^{k-1} p_a(b_j | b_{j-1}) q_r(g_j | b_j). \end{aligned} \quad (7)$$

In the above equality, the constant Q' is selected so that \tilde{p}^* is the probability distribution on \mathcal{A}^+ . The following lemma, which is essential to the existence of \tilde{p}^* , will be proved in Appendix A.

Lemma 1 *Let $y^m = y_1 \cdots y_m$ be a sequence from \mathcal{A} . $(b_1, l_1), \dots, (b_{K(y^m)}, g_{K(y^m)})$, where $K(y^m)$ is a random variable, is a sequence of symbol-run pairs associated with y^m . p_a, q_r are conditional probability functions defined in Section II. For convenience, we write*

$p_a(b_{K(y^m)}|b_{K(y^m)-1}) \max_{g' \geq g_{K(y^m)}} q_r(g'|b_{K(y^m)})$ as Δ_1 and $\prod_{j=2}^{K(y^m)-1} p_a(b_j|b_{j-1} q_r(g_j|b_j))$ as Δ_2 . Then

$$\sum_{y^m \in \mathcal{A}^m} \Delta_1 \Delta_2 \leq m|\mathcal{A}|.$$

In view of Lemma 1 and (7), one can easily verify that $Q' \geq 1/2$. A similar argument to the proof of Theorem 2 can then lead to

$$R_n^{s,rla} \leq O\left(\frac{1}{n}\right) + O\left(\frac{1}{\log n}\right) + O\left(\frac{\log \log n}{\log n}\right).$$

This completes the proof of Theorem 3.

IV. CONCLUSION

In order to give some insights into practical applications of this study, we present some experimental results on quantized AC coefficients in DCT(Discrete Cosine Transform)-based video compression in this final section.

Because MPEG 1, MPEG 2 and DV use quite similar RLE, we pick only DV to demonstrate. The algorithms used for comparison are: the RLE algorithm, the so-called YK algorithm (i.e. the improved sequential algorithm in [3]), the Unix Gzip based on LZ77. In all these three algorithms, AC coefficients in each DCT block are zig-zag scanned and concatenated into one sequence. Zig-zag scanning is illustrated in [2]. In the sequence of AC coefficients, the last run of 0's in each DCT block is replaced by a special symbol identifiable from the symbols in the source alphabet \mathcal{A} . Since the size of each DCT block is known, the decoder knows how many number of 0's should be filled into a block when a special symbol is decoded. Tables 1 and 2 list the experimental results on two different 525/60 DV video sequences. All compression rates are expressed in terms of bits per AC coefficient. In each row, the best compression rate is highlighted.

Tab. 1: Results on randomly chosen luminance frames

Cases	RLE	The YK	Gzip
One Frame 1	1.762	1.74	2.362
One Frame 2	1.833	1.829	2.311
Five Frames 1	1.769	1.719	2.055
Five Frames 2	1.832	1.797	2.128
Ten Frames 1	1.769	1.712	2.060
Ten Frames 2	1.833	1.788	2.128

In Table 1, we can clearly see that both the RLE algorithm and the YK algorithm beat Gzip by considerable margin in compression rates. As we have mentioned earlier, experimental results again suggest that the sequence of quantized AC coefficients in DCT-based video compression can be approximated by semi-Markov sources fairly well, for which RLE turns out to be quite efficient. However, the YK algorithm is still competitive to the RLE algorithm on all frames, esp. on color difference frames. In the context of universal lossless data compression, YK algorithms are indeed superior to RLE algorithms when sources are not guaranteed to be semi-Markov.

Tab. 2: Results on randomly chosen color difference

Cases	RLE	The YK
One Frame 1	0.276	0.259
One Frame 2	0.555	0.553
50 Frames 1	0.275	0.216
50 Frames 2	0.561	0.527
100 Frames 1	0.267	0.212
100 Frames 2	0.556	0.518

APPENDIX A

In this appendix, we prove Lemma 1. Define $T_m \triangleq \min\{k : \sum_{j=1}^k g_j \geq m\}$. T_m is sometimes called the *stopping time*. Then

$$P\{T_m = k | (b_1, g_1)\} = \sum_{\substack{y^m \in \mathcal{A}^m, K(y^m) = k, \\ (b_1, g_1) \text{ is parsed}}} \Delta_3 \Delta_2, \quad (\text{A1})$$

where $\Delta_3 \triangleq p_a(b_k | b_{k-1}) q_r(\hat{g} \geq g_k | b_k)$ and $q_r(\hat{g} \geq g_k | b_k) = \sum_{\hat{g} \geq g_k} q_r(\hat{g} | b_k)$. From (A1) and the fact that

$$q_r(\hat{g} \geq g_k | b_k) \geq \max_{g' \geq g_k} q_r(g' | b_k),$$

we have

$$\begin{aligned} \sum_{y^m \in \mathcal{A}^m} \Delta_1 \Delta_2 &\leq |\mathcal{A}| + \\ &\sum_{(b_1, g_1) \in \mathcal{A} \times \{0, \dots, m-1\}} \sum_{k=2}^{\infty} P\{T_m = k | (b_1, g_1)\} \\ &\leq m|\mathcal{A}|. \end{aligned}$$

This completes the proof of Lemma 1.

REFERENCES

- [1] E.-H. Yang and Y. Jia, "Universal lossless coding of sources with large or unbounded alphabets," *Numbers, Information and Complexity* (Ingo Althofer, *et al*, eds.), Kluwer Academic Publishers, pp. 421-442, 2000.
- [2] Proposed SMPTE STANDARD for Television—data structure for DV-based audio data and compressed video—25 and 50 mb/s. *SMPTE Journal*, pages 308-324, May 1999.
- [3] E.-H. Yang and J. C. Kieffer, "Efficient universal lossless data compression algorithms based on a greedy sequential grammar transform—Part one: Without context models," *IEEE Trans. Inform. Theory*, vol. IT-46, pp. 755-788, 2000.
- [4] A. Banerji and S. Goel, "Architectures for efficient implementation of the YK lossless data compression algorithm," to appear in *Proc. DCC'2001(Snowbird, Utah)*, USA, March 27-29, 2001.
- [5] A. Banerji and D. Dillon, "Lossless compression for satellite packet networks using the YK algorithm," to appear in *Proc. DCC'2001(Snowbird, Utah)*, USA, March 27-29, 2001.
- [6] I. Csiszar and P. C. Shields, "Redundancy rates for renewal and other processes," *IEEE Trans. Inform. Theory*, vol. IT-42, pp. 2065-2072, 1996.
- [7] P. Elias, "Universal codeword sets and representations of the integers," *IEEE Trans. Inform. Theory*, vol. IT-21, pp. 193-203, March 1975.
- [8] N. Abramson, *Information Theory and Coding*. New York: McGraw-Hill, 1963.
- [9] T. M. Cover and J. A. Thomas, *Elements of Information Theory*. New York: Wiley, 1991.