A Universal Algorithm for Sequential Data Compression

JACOB ZIV, FELLOW, IEEE, AND ABRAHAM LEMPEL, MEMBER, IEEE

Abstract—**A universal algorithm for sequential data compression is presented. Its performance is investigated with respect to a nonprobabilistic model of constrained sources. The compression ratio achieved by the proposed universal code uniformly approaches the lower bounds on the compression ratios attainable by block-to-variable codes and variable-to-block codes designed to match a completely specified source.**

I. Introduction

In many situations arising in digital communications and data processing, the encountered strings of data display various structural regularities or are otherwise subject to certain constraints, thereby allowing for storage and time-saving techniques of data compression. Given a discrete data source, the problem of data compression is first to identify the limitations of the source, and second to devise a coding scheme which, subject to certain performance criteria, will best compress the given source.

Once the relevant source parameters have been identified, the problem reduces to one of minimum-redundancy coding. This phase of the problem has received extensive treatment in the literature [1]–[7].

When no *a priori* knowledge of the source characteristics is available, and if statistical tests are either impossible or unreliable, the problem of data compression becomes considerably more complicated. In order to overcome these difficulties one must resort to universal coding schemes whereby the coding process is interlaced with a learning process for the varying source characteristics [8], [9]. Such coding schemes inevitably require a larger working memory space and generally employ performance criteria that are appropriate for a wide variety of sources.

In this paper, we describe a universal coding scheme which can be applied to any discrete source and whose performance is comparable to certain optimal fixed code book schemes designed for completely specified sources. For lack of adequate criteria, we do not attempt to rank the proposed scheme with respect to other possible universal coding schemes. Instead, for the broad class of sources defined in Section III, we derive upper bounds on the compression efficiency attainable with full *a priori* knowledge of the source by fixed code book schemes, and then show that the efficiency of our universal code with no *a priori* knowledge of the source approaches those bounds.

The proposed compression algorithm is an adaptation of a simple copying procedure discussed recently [10] in a study on the complexity of finite sequences. Basically, we employ the concept of encoding future segments of the source-output via maximum-length copying from a buffer containing the recent past output. The transmitted codeword consists of the buffer address and the length of the copied segment. With a predetermined initial load of the buffer and the information contained in the codewords, the source data can readily be reconstructed at the decoding end of the process.

The main drawback of the proposed algorithm is its susceptibility to error propagation in the event of a channel error.

II. The Compression Algorithm

The proposed compression algorithm consists of a rule for parsing strings of symbols from a finite alphabet A into substrings, or words, whose lengths do not exceed a prescribed integer L_s, and a coding scheme which maps these substrings sequentially into uniquely decipherable codewords of fixed length L_c over the same alphabet A.

The word-length bounds L_s and L_c allow for bounded-delay encoding and decoding, and they are related by

$$L_c = 1 + \lceil \log (n - L_s) \rceil + \lceil \log L_s \rceil, \tag{1}$$

where $\lceil x \rceil$ is the least integer not smaller than x, the logarithm base is the cardinality α of the alphabet A, and n is the length of a buffer, employed at the encoding end of the process, which stores the latest n symbols emitted by the source. The exact relationship between n and L_s is discussed in Section III. Typically, $n \simeq L_s \alpha^{hL_s}$, where $0 < h < 1$. For on-line decoding, a buffer of similar length has to be employed at the decoding end also.

To describe the exact mechanics of the parsing and coding procedures, we need some preparation by way of notation and definitions.

Consider a finite alphabet A of α symbols, say $A = \{0,1,\cdots,\alpha-1\}$. A string, or word, S of length $\ell(S) = k$ over A is an ordered k-tuple $S = s_1 s_2 \cdots s_k$ of symbols from A. To indicate a substring of S which starts at position i and ends at position j, we write $S(i,j)$. When $i \leq j$, $S(i,j) = s_i s_{i+1} \cdots s_j$, but when $i > j$, we take $S(i,j) = \Lambda$, the null string of length zero.

The concatenation of strings Q and R forms a new string $S = QR$; if $\ell(Q) = k$ and $\ell(R) = m$, then $\ell(S) = k + m$, $Q = S(1,k)$, and $R = S(k+1, k+m)$. For each j, $0 \leq j \leq$

Manuscript received June 23, 1975; revised July 6, 1976. Paper previously presented at the IEEE International Symposium on Information Theory, Ronneby, Sweden, June 21–24, 1976.

J. Ziv was with the Department of Electrical Engineering, Technion–Israel Institute of Technology, Haifa, Israel. He is now with the Bell Telephone Laboratories, Murray Hill, NJ 07974.

A. Lempel was with the Department of Electrical Engineering, Technion—Israel Institute of Technology, Haifa, Israel. He is now with the Sperry Research Center, Sudbury, MA 01776.

$\ell(S)$, $S(1,j)$ is called a *prefix* of S; $S(1,j)$ is a *proper* prefix of S if $j < \ell(S)$.

Given a proper prefix $S(1,j)$ of a string S and a positive integer i such that $i \leq j$, let $L(i)$ denote the largest nonnegative integer $\ell \leq \ell(S) - j$ such that

$$S(i,i + \ell - 1) = S(j + 1,j + \ell),$$

and let p be a position of $S(1,j)$ for which

$$L(p) = \max_{1 \leq i \leq j} \{L(i)\}.$$

The substring $S(j + 1,j + L(p))$ of S is called the *reproducible extension* of $S(1,j)$ into S, and the integer p is called the *pointer* of the reproduction. For example, if $S = 00101011$ and $j = 3$, then $L(1) = 1$ since $S(j + 1,j + 1) = S(1,1)$ but $S(j + 1,j + 2) \neq S(1,2)$. Similarly, $L(2) = 4$ and $L(3) = 0$. Hence, $S(3 + 1,3 + 4) = 0101$ is the reproducible extension of $S(1,3) = 001$ into S with pointer $p = 2$.

Now, to describe the encoding process, let $S = s_1 s_2 \cdots$ denote the string of symbols emitted by the source. The sequential encoding of S entails parsing S into successive source words, $S = S_1 S_2 \cdots$, and assigning a codeword C_i for each S_i. For bounded-delay encoding, the length ℓ_i of each S_i is at most equal to a predetermined parameter L_s, while each C_i is of fixed length L_c as given by (1).

To initiate the encoding process, we assume that the output S of the source was preceded by a string Z of $n - L_s$ zeros, and we store the string $B_1 = ZS(1,L_s)$ in the buffer. If $S(1,j)$ is the reproducible extension of Z into $ZS(1,L_s - 1)$, then $S_1 = S(1,j + 1)$ and $\ell_1 = j + 1$. To determine the next source word, we shift out the first ℓ_1 symbols from the buffer and feed into it the next ℓ_1 symbols of S to obtain the string $B_2 = B_1(\ell_1 + 1,n)S(L_s + 1, L_s + \ell_1)$. Now we look for the reproducible extension E of $B_2(1,n - L_s)$ into $B_2(1,n - 1)$, and set $S_2 = Es$, where s is the symbol next to E in B_2. In general, if B_i denotes the string of n source symbols stored in the buffer when we are ready to determine the ith source word S_i, the successive encoding steps can be formally described as follows.

1) Initially, set $B_1 = 0^{n-L_s}S(1,L_s)$, i.e., the all-zero string of length $n - L_s$ followed by the first L_s symbols of S.

2) Having determined B_i, $i \geq 1$, set

$$S_i = B_i(n - L_s + 1,n - L_s + \ell_i),$$

where the prefix of length $\ell_i - 1$ of S_i is the reproducible extension of $B_i(1,n - L_s)$ into $B_i(1,n - 1)$.

3) If p_i is the reproduction pointer used to determine S_i, then the codeword C_i for S_i is given by

$$C_i = C_{i1}C_{i2}C_{i3},$$

where C_{i1} is the radix-α representation of $p_i - 1$ with $\ell(C_{i1}) = \lceil \log (n - L_s) \rceil$, C_{i2} is the radix-α representation of $\ell_i - 1$ with $\ell(C_{i2}) = \lceil \log L_s \rceil$, and C_{i3} is the last symbol of S_i, i.e., the symbol occupying position $n - L_s + \ell_i$ of B_i. The total length of C_i is given by

$$\ell(C_i) = \lceil \log (n - L_s) \rceil + \lceil \log L_s \rceil + 1$$

in accordance with (1).

4) To update the contents of the buffer, shift out the symbols occupying the first ℓ_i positions of the buffer while feeding in the next ℓ_i symbols from the source to obtain

$$B_{i+1} = B_i(\ell_i + 1,n)S(h_i + 1,h_i + \ell_i),$$

where h_i is the position of S occupied by the last symbol of B_i.

This completes the description of the encoding process. It is easy to verify that the parsing rule defined by (2) guarantees a bounded, positive source word length in each iteration; in fact, $1 \leq \ell_i \leq L_s$ for each i thus allowing for a radix-α representation of $\ell_i - 1$ with $\lceil \log L_s \rceil$ symbols from A. Also, since $1 \leq p_i \leq n - L_s$ for each i, it is possible to represent $p_i - 1$ with $\lceil \log (n - L_s) \rceil$ symbols from A.

Decoding can be performed simply by reversing the encoding process. Here we employ a buffer of length $n - L_s$ to store the latest decoded source symbols. Initially, the buffer is loaded with $n - L_s$ zeros. If $D_i = d_1 d_2 \cdots d_{n-L_s}$ denotes the contents of the buffer after C_{i-1} has been decoded into S_{i-1}, then

$$S_{i-1} = D_i(n - L_s - \ell_{i-1} + 1,n - L_s),$$

where $\ell_{i-1} = \ell(S_{i-1})$, and where D_{i+1} can be obtained from D_i and C_i as follows.

Determine $p_i - 1$ and $\ell_i - 1$ from the first $\lceil \log (n - L_s) \rceil$ and the next $\lceil \log L_s \rceil$ symbols of C_i. Then, apply $\ell_i - 1$ shifts while feeding the contents of stage p_i into stage $n - L_s$. The first of these shifts will change the buffer contents from D_i to

$$D_i^{(1)} = d_2 d_3 \cdots d_{n-L_s} d_{p_i} = d_1^{(1)} d_2^{(1)} \cdots d_{n-L_s}^{(1)}.$$

Similarly, if $j \leq \ell_i - 1$, the jth shift will transform $D_i^{(j-1)} = d_1^{(j-1)} d_2^{(j-1)} \cdots d_{n-L_s}^{(j-1)}$ into $D_i^{(j)} = d_2^{(j-1)} d_3^{(j-1)} \cdots d_{n-L_s}^{(j-1)} d_{p_i}^{(j-1)} = d_1^{(j)} d_2^{(j)} \cdots d_{n-L_s}^{(j)}$. After these $\ell_i - 1$ shifts are completed, shift once more, while feeding the last symbol of C_i into stage $n - L_s$ of the buffer. It is easy to verify that the resulting load of the buffer contains S_i in its last $\ell_i = \ell(S_i)$ positions.

The following example will serve to illustrate the mechanics of the algorithm. Consider the ternary ($\alpha = 3$) input string

$$S = 00101021021021202102120 0 \cdots,$$

and an encoder with parameters $L_s = 9$ and $n = 18$. (These parameters were chosen to simplify the illustration; they do not reflect the design considerations to be discussed in Section III.) According to (1), the corresponding codeword length is given by

$$L_c = 1 + \log_3 (18 - 9) + \log_3 9 = 5.$$

Initially, the buffer is loaded with $n - L_s = 9$ zeros, followed by the first $L_s = 9$ digits of S, namely,

$$B_1 = \underbrace{0\,0\,0\,0\,0\,0\,0\,0\,0}_{n - L_s = 9}\ \underbrace{0\,0\,1\,0\,1\,0\,2\,1\,0.}_{L_s = 9}$$

To determine the first source word S_1, we have to find the longest prefix $B_1(10,9 + \ell_1 - 1)$ of

$$B_1(10,17) = 0\,0\,1\,0\,1\,0\,2\,1$$

which matches a substring of B_1 that starts in position $p_1 \leq 9$ and then set $S_1 = B_1(10,9 + \ell_1)$. It is easily seen that the longest match in this case is $B_1(10,11) = 00$, and hence $S_1 = 001$ and $\ell_1 = 3$. The pointer p_1 for this step can be any integer between one and nine; we choose to set $p_1 = 9$. The two-digit radix-3 representation of $p_1 - 1$ is $C_{11} = 22$, and that of $\ell_1 - 1$ is $C_{12} = 02$. Since C_{i3} is always equal to the last symbol of S_i, the codeword for S_1 is given by $C_1 = 22021$.

To obtain the buffer load B_2 for the second step, we shift out the first $\ell_1 = 3$ digits of B_1 and feed in the next 3 digits $S(10,12) = 210$ of the input string S. The details of steps 2, 3, and 4 are tabulated below, where pointer positions are indicated by arrows and where the source words S_i are indicated by the italic substring of the corresponding buffer load B_i

$$B_2 = 0\,0\,0\,0\,0\,0\,0\,0\,1\,\textit{0}\,\textit{1}\,\textit{0}\,\textit{2}\,1\,0\,2\,1\,0, \qquad C_2 = 21102$$

$$B_3 = 0\,0\,0\,0\,1\,0\,1\,0\,2\,\textit{1}\,\textit{0}\,\textit{2}\,\textit{1}\,\textit{0}\,\textit{2}\,\textit{1}\,\textit{2}\,0, \qquad C_3 = 20212$$

$$B_4 = 2\,1\,0\,2\,1\,0\,2\,1\,2\,\textit{0}\,\textit{2}\,\textit{1}\,\textit{0}\,\textit{2}\,\textit{1}\,\textit{2}\,\textit{0}\,\textit{0}, \qquad C_4 = 02220.$$

III. Compression of Constrained Sources

In this section, we investigate the performance of the proposed compression algorithm with respect to a nonprobabilistic model of constrained information sources. After defining the source model, we derive lower bounds on the compression ratios attainable by block-to-variable and variable-to-block codes under full knowledge of the source, and then show that the compression ratio achieved by our universal code approaches these bounds.

A. Definition of the Source Model

Let $A = \{0,1,\cdots,\alpha - 1\}$ be the given α-symbol alphabet, and let A^* denote the set of all finite strings over A. Given a string $S \in A^*$ and a positive integer $m \leq \ell(S)$, let $S\{m\}$ denote the set of all substrings of length m contained in S, and let $S(m)$ denote the cardinality of $S\{m\}$. That is,

$$S\{m\} = \bigcup_{i=0}^{\ell(S)-m} S(i + 1, i + m)$$

and

$$S(m) = |S\{m\}|.$$

Given a subset σ of A^*, let

$$\sigma\{m\} = \{S \in \sigma \mid \ell(S) = m\},$$

and let $\sigma(m)$ denote the cardinality of $\sigma\{m\}$.

A subset σ of A^* is called a *source* if the following three properties hold:

1) $A \subset \sigma$ (i.e., σ contains all the unit length strings),
2) $S \in \sigma$ implies $SS \in \sigma$,
3) $S \in \sigma$ implies $S\{m\} \subset \sigma\{m\}$.

Typically, such a source σ is defined by specifying a finite set of strings over A which are forbidden to appear as substrings of elements belonging to σ, and therefore $\sigma(m) < \alpha^m$ for all m exceeding some m_0.

With every source σ, we associate a sequence $h(1), h(2), \cdots$ of parameters, called the *h-parameters* of σ, where[1]

$$h(m) = \frac{1}{m} \log \sigma(m), \qquad m = 1,2,\cdots. \tag{2}$$

It is clear that $0 \leq h(m) \leq 1$ for all m and, by 2) it is also clear that $mh(m)$ is a nondecreasing function of m. The sequence of h-parameters, however, is usually nonincreasing in m. To avoid any possible confusion in the sequel, we postulate this property as an additional defining property of a source. Namely, we require

4) $h(m) = 1/m \log \sigma(m)$ is a nonincreasing function of m.

B. Some Lower Bounds on the Compression Ratio

Consider a compression coding scheme for a source σ which employs a block-to-variable (BV) code book of M pairs (X_i, Y_i) of words over A, with $\ell(X_i) = L$ for $i = 1,2,\cdots,M$. The encoding of an infinitely long string $S \in \sigma$ by such a code is carried out by first parsing S into blocks of length L, and then replacing each block X_i by the corresponding codeword Y_i. It is assumed, of course, that the code book is exhaustive with respect to σ and uniquely decipherable [2]. Hence, we must have

$$\{X_i\}_{i=1}^{M} = \sigma\{L\}$$

or

$$M = \sigma(L) = \alpha^{Lh(L)}, \tag{3}$$

and

$$\max_{1 \leq i \leq M} \{\ell(Y_i)\} \geq \log M = Lh(L). \tag{4}$$

The compression ratio ρ_i associated with the ith word-pair of the code is given by

$$\rho_i = \frac{\ell(Y_i)}{L}.$$

The *BV compression ratio*, $\rho_{BV}(\sigma,M)$, of the source σ is defined as the minimax value of ρ_i, where the maximization is over all word-pairs of a given code, and the minimization is over the set $C_{BV}(\sigma,M)$ of all BV code books consisting of M word-pairs. Thus,

$$\rho_{BV}(\sigma,M) = \min_{C_{BV}(\sigma,M)} \max_{1 \leq i \leq M} \left\{ \frac{\ell(Y_i)}{L} \right\}$$

$$\geq \frac{\log M}{L} = \frac{Lh(L)}{L} = h(L).$$

[1] Throughout this paper, $\log x$ means the base-α logarithm of x.

For later reference, we record this result in the following lemma.

Lemma 1:

$$\rho_{BV}(\sigma,M) \geq h(L),$$

where

$$Lh(L) = \log M.$$

Now, consider a compression scheme which employs a variable-to-block (VB) code book of M word-pairs (X_i,Y_i), with $\ell(Y_i) = L$ for all $i = 1,2,\cdots,M$. In this case, the compression ratio associated with the ith word-pair is given by

$$\rho_i = \frac{L}{\ell(X_i)},$$

and similarly, the VB compression ratio $\rho_{VB}(\sigma,M)$ of σ is defined as the minimax value of ρ_i over all word-pairs and over the set $C_{VB}(\sigma,M)$ of VB code books with M word-pairs.

Lemma 2:

$$\rho_{VB}(\sigma,M) \geq h(L_M)$$

where

$$L_M = \max\{\ell \mid M \geq \sigma(\ell)\}.$$

Proof: We may assume, without loss of generality, that in every code under consideration

$$\ell(X_1) \leq \ell(X_2) \leq \cdots \leq \ell(X_M),$$

and hence for each $C \in C_{VB}(\sigma,M)$,

$$\max \rho_i(C) = \frac{L(C)}{\ell(X_1)}. \tag{5}$$

Since C is exhaustive with respect to σ, we have

$$M \geq \sigma(\ell_1), \tag{6}$$

where $\ell_1 = \ell(X_1)$; and since C is uniquely decipherable, we have

$$L(C) \geq \log M. \tag{7}$$

From the definition of L_M, inequality (6), and the nondecreasing property of $\sigma(\ell)$, we obtain

$$\ell_1 \leq L_M. \tag{8}$$

From (5), (7), and (8), we have

$$\max \rho_i(C) \geq \frac{\log M}{L_M};$$

and since

$$M \geq \sigma(L_M) = \alpha^{L_M h(L_M)},$$

it follows that for each $C \in C_{VB}(\sigma,M)$,

$$\max \rho_i(C) \geq h(L_M).$$

Q.E.D.

Remarks

1) Since the value of L in the context of Lemma 1 satisfies the definition of L_M as given in Lemma 2, it follows that the bounds of both lemmas are essentially the same, despite the basic difference between the respective coding schemes.

2) By 2), the second defining property of a source, the per-word bounds derived above apply to indefinitely long messages as well, since the whole message may consist of repeated appearances of the same worst case word.

3) By 4), the nonincreasing property of the h-parameters, the form of the derived bounds confirms the intuitive expectation that an increase in the size M of the employed code book causes a decrease in the lower bound on the attainable compression ratio.

C. Performance of the Proposed Algorithm

We proceed now to derive an upper bound on the compression ratio attainable by the algorithm of Section II. To this end, we consider the worst case source message of length $n - L_s$, where n is the prospective buffer length and L_s is the maximal word-length. The bound obtained for this case will obviously apply to all messages of length $n - L_s$ or greater.

First, we assume that only the h-parameters of the source under consideration are known to the designer of the encoder. Later, when we discuss the universal performance of the proposed algorithm, we will show that even this restricted *a priori* knowledge of the source is actually unessential.

We begin by choosing the buffer length n to be an integer of the form

$$n = \sum_{m=1}^{\lambda} m\alpha^m + \sum_{m=\lambda+1}^{\ell} m\sigma(\ell) + (\ell+1)(N_{\ell+1}+1), \tag{9}$$

where

$$N_{\ell+1} = \sum_{m=1}^{\lambda} (\ell-m)\alpha^m + \sum_{m=\lambda+1}^{\ell} (\ell-m)\sigma(\ell), \tag{10}$$

$\lambda = \lfloor \ell h(\ell) \rfloor$, the integer part of $\log \sigma(\ell) = \ell h(\ell)$, and

$$\ell = L_s - 1. \tag{11}$$

The specific value of the parameter L_s is left for later determination (see the first remark following the proof of Theorem 1). The reasoning that motivates the given form of n will become clear from subsequent derivations.

Consider a string $S \in \sigma\{n - L_s\}$, and let $N(S)$ denote the number of words into which S is parsed by the algorithm during the encoding process. Recalling that each of these words is mapped into a codeword of fixed length L_c (see (1)), it follows that the compression ratio $\rho(S)$ associated with the string S is given by

$$\rho(S) = \frac{L_c}{n - L_s} N(S).$$

Hence, the compression ratio ρ attainable by the algorithm for a given source σ is

$$\rho = \frac{L_c}{n - L_s} N, \tag{12}$$

where

$$N = \max_{S \in \sigma\{n - L_s\}} N(S).$$

Let $Q \in \sigma\{n - L_s\}$ be such that $N(Q) = N$, and suppose that the algorithm parses Q into $Q = Q_1 Q_2 \cdots Q_N$. From step 2) of the encoding cycle it follows that if $\ell(Q_i) = \ell(Q_j) < L_s$, for some $i < j < N$, then $Q_i \neq Q_j$. (Note that when Q_j is being determined at the jth cycle of the encoding process, all of Q_i is still stored in the buffer, and since $\ell(Q_j) < L_s$, the longest substring in the buffer that precedes Q_j and is a prefix of Q_j must be of length $\ell(Q_j) - 1$.)

Denoting by K_m the number of Q_i, $1 \leq i \leq N - 1$, of length m, $1 \leq m \leq L_s$, we have

$$N = 1 + \sum_{m=1}^{L_s} K_m.$$

By the above argument, and by property 3) of the source, we have

$$K_m \leq \sigma(m), \qquad \text{for } 1 \leq m \leq \ell = L_s - 1.$$

Since

$$n - L_s = \ell(Q_N) + \sum_{m=1}^{\ell+1} m K_m,$$

and n and L_s are both fixed, it is clear that by overestimating the values of K_m for $1 \leq m \leq \ell$ at the expense of $K_{\ell+1}$, we can only overestimate the value of N. Therefore, since $\sigma(m) \leq \sigma(m+1)$ and $\sigma(m) = \alpha^{mh(m)} \leq \alpha^m$, we obtain

$$N \leq K'_{\ell+1} + \sum_{m=1}^{\ell} K'_m = N', \tag{13}$$

where

$$K'_m = \begin{cases} \alpha^m, & \text{for } 1 \leq m \leq \lambda = \lfloor \ell h(\ell) \rfloor \\ \sigma(\ell), & \text{for } \lambda < m \leq \ell \end{cases} \tag{14}$$

and

$$K'_{\ell+1} = \left\lceil \frac{1}{\ell + 1} \left(n - L_s - \sum_{m=1}^{\ell} m K'_m \right) \right\rceil. \tag{15}$$

From (14), (15), and (9), we obtain $K'_{\ell+1} = N_{\ell+1}$, and

$$N' = N_{\ell+1} + \sum_{m=1}^{\lambda} \alpha^m + \sum_{m=\lambda+1}^{\ell} \sigma(\ell)$$

which, together with (9) and (10), yields

$$n - L_s - \ell N' = \sum_{m=1}^{\lambda} m\alpha^m + \sum_{m=\lambda+1}^{\ell} m\sigma(\ell) + N_{\ell+1} - \ell \left[\sum_{m=1}^{\lambda} \alpha^m + \sum_{m=\lambda+1}^{\ell} \sigma(\ell) \right] = 0,$$

or

$$N \leq N' = \frac{n - L_s}{\ell} = \frac{n - L_s}{L_s - 1}. \tag{16}$$

Hence, from (12) and (16), we have

$$\rho \leq \frac{L_c}{\ell} = \frac{L_c}{L_s - 1}. \tag{17}$$

Note that despite the rater rudimentary overestimation of N by N', the upper bound of (17) is quite tight, since the fact that no source word is longer than L_s immediately implies $\rho \geq L_c / L_s$.

We can state now the following result.

Theorem 1: If the buffer length n for a source with known h-parameters is chosen according to (9), then

$$\rho \leq h(L_s - 1) + \epsilon(L_s),$$

where

$$\epsilon(L_s) = \frac{1}{L_s - 1} \left(3 + 3 \log (L_s - 1) + \log \frac{L_s}{2} \right).$$

Proof: From (1) we have

$$L_c = 1 + \lceil \log L_s \rceil + \lceil \log (n - L_s) \rceil \leq 3 + \log (L_s - 1) + \log (n - L_s).$$

From (9) and (10) we obtain

$$n - L_s = \ell \left[\sum_{m=1}^{\lambda} (\ell - m)\alpha^m + \sum_{m=\lambda+1}^{\ell} (\ell - m)\sigma(\ell) + \sum_{m=1}^{\lambda} \alpha^m + \sum_{m=\lambda+1}^{\ell} \sigma(\ell) \right],$$

and since $\alpha^m \leq \sigma(\ell)$, for $1 \leq m \leq \lambda$, we have

$$n - L_s \leq \ell \sigma(\ell) \sum_{m=1}^{\ell} (\ell - m + 1) = \frac{1}{2} \ell^2 (\ell + 1) \sigma(\ell),$$

or

$$\log (n - L_s) \leq 2 \log \ell + \log \frac{\ell + 1}{2} + \ell h(\ell).$$

Since $\ell = L_s - 1$, we obtain

$$L_c \leq 3 + 3 \log (L_s - 1) + \log \frac{L_s}{2} + (L_s - 1) h(L_s - 1),$$

or

$$L_c \leq (L_s - 1)[h(L_s - 1) + \epsilon(L_s)].$$

Substituting this result into (17), we obtain the bound of Theorem 1.

Q.E.D.

Remarks

1) The value of $\epsilon(L_s)$ decreases with L_s and, consequently, the compression ratio ρ approaches the value of $h(L_s - 1)$, the h-parameter associated with the second largest word-length processed by the encoder. Given any $\delta > 0$, one can always find the least integer ℓ_s such that $\rho - h(\ell_s - 1) \leq \delta$. The magnitude of the acceptable de-

viation δ determines the operational range of L_s, namely, $L_s \geq \ell_s$.

2) Since our code maps source words of variable length at most L_s into codewords of fixed length L_c, we adopt as a reference for comparison the best *VB* code C_{VB} discussed in Subsection III-B. The counterpart of our block-length L_c is $L(C)$ of (5), and by (7) we can write

$$L_c \approx \log M \approx L_M h(L_M), \tag{18}$$

where M is the code book size and $h(L_M)$ is the lower bound (see Lemma 2) on the compression ratio ρ_{VB} attainable by C_{VB}. Since for sufficiently large L_s, we have also

$$L_c \approx (L_s - 1)\rho \approx (L_s - 1)h(L_s - 1), \tag{19}$$

it follows that $\rho \approx \rho_{VB}$.

We turn now to investigate the universal performance of the proposed algorithm, i.e., the performance when no *a priori* knowledge of the source to be compressed is available.

Given δ_1 and h_1 such that $0 < \delta_1 < h_1 < 1$, let ℓ_1 be the least positive integer satisfying

$$\delta_1 \geq \epsilon(\ell_1 + 1) = \frac{1}{\ell_1}\left(3 + 3\log \ell_1 + \log \frac{\ell_1 + 1}{2}\right),$$

and let $K = \alpha^{\ell_1 h_1}$ and $\lambda_1 = \lfloor \ell_1 h_1 \rfloor$. Consider the encoder E_1 which employs a buffer of length $n_1 = n(h_1, \ell_1)$, obtained from (9) and (10) by setting $\ell = \ell_1$, $\lambda = \lambda_1$, and $\sigma(\ell) = K$, and whose maximal word-length L_s is equal to $\ell_1 + 1$.

It follows from Theorem 1 that if this encoder is applied to a source σ_1 such that $h_{\sigma_1}(\ell_1) = h_1$, then the resulting compression ratio $\rho_1(\sigma_1)$ satisfies

$$\rho_1(\sigma_1) \leq h_{\sigma_1}(\ell_1) + \delta_1 = h_1 + \delta_1. \tag{20}$$

Suppose now that h_1 and δ_1 were chosen to fit the prescribed upper value $\rho_1 = h_1 + \delta_1$ of a prospective compression ratio range (ρ_0, ρ_1) with

$$0 < \delta_1 R < \rho_0 < \rho_1 < 1, \qquad R > 1, \tag{21}$$

where R is an adjustment parameter to be determined later. As shown above, the encoder E_1 is then matched exactly to the upper value ρ_1 of the prospective compression range. In order to accommodate the whole given range, we propose to employ a slightly larger encoder E_0 whose buffer is of length $n_0 = n_1 - \ell_1 + \ell_0$, where n_1 and ℓ_1 are the parameters of E_1, and where ℓ_0 is an integer, greater than ℓ_1, for which the solution h_0 of the equation

$$n_1 - \ell_1 + \ell_0 = n(h_0, \ell_0) \tag{22}$$

satisfies

$$\rho_0 - \epsilon(\ell_0) < h_0 \leq \rho_0 - \epsilon(\ell_0 + 1). \tag{23}$$

Noting that $n_0 - \ell_0 = n_1 - \ell_1$ is fixed, it is clear from (9) and (10) that as ℓ_0 increases h_0 decreases; also, (21) and the fact that $\ell_0 > \ell_1$ imply that $\rho_0 - \epsilon(\ell_0 + 1) > \rho_0 - \epsilon(\ell_1 + 1) \geq \rho_0 - \delta_1 > 0$. Hence, there exist $h_0 > 0$ and $\ell_0 > \ell_1$ that satisfy both (22) and (23).

In analogy with (20), it is also clear that if E_0 is applied to a source σ_0 such that $h_{\sigma_0}(\ell_0) = h_0$, then the resulting compression ratio $\rho_0(\sigma_0)$ satisfies

$$\rho_0(\sigma_0) \leq h_{\sigma_0}(\ell_0) + \delta_0 = h_0 + \delta_0, \tag{24}$$

where $\delta_0 = \epsilon(\ell_0 + 1)$.

From (23) and (24), we have $\rho_0(\sigma_0) \leq \rho_0 - \epsilon(\ell_0 + 1) + \delta_0 = \rho_0$, and

$$\delta_0 \leq \rho_0 - h_0 < \epsilon(\ell_0) \leq \epsilon(\ell_1 + 1) \leq \delta_1 < \frac{1}{R}\rho_0.$$

Hence, h_0 can be made arbitrarily close to ρ_0, and consequently, the encoder E_0 matches as closely as desired the lower end ρ_0 of the given compression range.

Theorem 2: Let σ be a source for which a matched encoder E achieves a compression ratio $\rho(\sigma)$ within the range (ρ_0, ρ_1). Then the compression ratio $\rho_0(\sigma)$, achieved for σ by E_0, satisfies

$$\rho_0(\sigma) \leq \rho(\sigma) + \Delta,$$

where

$$\Delta \leq \frac{\lceil \log d \rceil}{\ell_1} \qquad d = \max\left\{\frac{h_1}{h_0}, \frac{1}{1 - h_0}\right\}.$$

(Typically, $(h_1/h_0) > (1/1 - h_0)$ and $d = (h_1/h_0)$.)

Proof: To prove the theorem we shall consider the obviously worst case of applying E_0 to the source σ_1 whose matched encoder E_1 realizes ρ_1.

Let $\rho_0(\sigma_1)$ denote the compression ratio achievable by E_0 when applied to σ_1. According to (12), we have

$$\rho_0(\sigma_1) = \frac{L_{c0}}{n_0 - (\ell_0 + 1)} N_0(\sigma_1),$$

where

$$L_{ci} = 1 + \lceil \log (\ell_i + 1) \rceil + \lceil \log (n_i - \ell_i - 1) \rceil, \qquad i \in \{0,1\},$$

and $N_i(\sigma_j)$, $i,j \in \{0,1\}$, is the maximum number of words into which a string $S \in \sigma_j\{n_i - \ell_i - 1\}$ is parsed by E_i.

Since $n_0 - \ell_0 = n_1 - \ell_1$ and $\ell_0 > \ell_1$, it is easy to verify that[2] $N_0(\sigma_1) \leq N_1(\sigma_1)$. Also by (16),

$$N_1(\sigma_1) \leq N_1'(\sigma_1) = \frac{n_1 - (\ell_1 + 1)}{\ell_1} = \frac{n_0 - (\ell_0 + 1)}{\ell_1}.$$

Hence

$$\rho_0(\sigma_1) \leq \frac{L_{c0}}{\ell_1} = \frac{L_{c1}}{\ell_1} + \frac{L_{c0} - L_{c1}}{\ell_1}$$

$$\leq \rho_1 + \frac{L_{c0} - L_{c1}}{\ell_1},$$

and since

$$L_{c0} - L_{c1} = \lceil \log (\ell_0 + 1) \rceil - \lceil \log (\ell_1 + 1) \rceil \leq \lceil \log k \rceil,$$

[2] The assertion here is analogous to that of [10, theorem 1].

where $k = (\ell_0/\ell_1)$, we obtain

$$\rho_0(\sigma_1) \le \rho_1 + \frac{1}{\ell_1}\lceil \log k \rceil. \tag{25}$$

To obtain an upper bound on k, we first observe that

$$\ell_0 \sum_{m=\lambda_0+1}^{\ell_0} (\ell_0 - m + 1)\alpha^{\ell_0 h_0} \le n_0 - \ell_0$$

$$= n_1 - \ell_1 \le \ell_1 \sum_{m=1}^{\ell_1} (\ell_1 - m + 1)\alpha^{\ell_1 h_1},$$

which reduces to

$$k^2(1 - h_0)^2 \le \alpha^{\ell_1 h_1 (1 - k(h_0/h_1))}. \tag{26}$$

Now, either $k(1 - h_0) < 1$, or else the exponent on the right side of (26) must be nonnegative and thus $k \le (h_1/h_0)$. In either case,

$$k \le \max\left\{\frac{h_1}{h_0}, \frac{1}{1 - h_0}\right\} = d.$$

Q.E.D.

Theorems 1 and 2 demonstrate the efficiency and universality of the proposed algorithm. They show that an encoder designed to operate over a prescribed compression range performs, practically, as well as one designed to match a specific compression ratio within the given range. Moreover, for given complexity i.e., a given codeword length, the compression efficiency is comparable to that of an optimal variable-to-block code book designed to match a given source.

References

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

[2] R. M. Karp, "Minimum-redundancy coding for the discrete noiseless channel," *IRE Trans. Inform. Theory*, vol. IT-17, pp. 27–38, Jan. 1961.

[3] B. F. Varn, "Optimal variable length codes," *Inform. Contr.*, vol. 19, pp. 289–301, 1971.

[4] Y. Perl, M. R. Gary, and S. Even, "Efficient generation of optimal prefix code: Equiprobable words using unequal cost letters," *J. ACM*, vol. 22, pp. 202–214, April 1975.

[5] A. Lempel, S. Even, and M. Cohn, "An algorithm for optimal prefix parsing of a noiseless and memoryless channel," *IEEE Trans. Inform. Theory*, vol. IT-19, pp. 208–214, March 1973.

[6] F. Jelinek and K. S. Schneider, "On variable length to block coding," *IEEE Trans. Inform. Theory*, vol. IT-18, pp. 765–774, Nov. 1972.

[7] R. G. Gallager, *Information Theory and Reliable Communication*. New York: Wiley, 1968.

[8] J. Ziv, "Coding of sources with unknown statistics—Part I; Probability of encoding error," *IEEE Trans. Inform. Theory*, vol. IT-18, pp. 384–394, May 1972.

[9] L. D. Davisson, "Universal noiseless coding," *IEEE Trans. Inform. Theory*, vol. IT-19, pp. 783–795, Nov. 1973.

[10] A. Lempel and J. Ziv, "On the complexity of finite sequences," *IEEE Trans. Inform. Theory*, vol. IT-22, pp. 75–81, Jan. 1976.

[11] B. M. Fitingof, "Optimal coding in the case of unknown and changing message statistics," *Prob. Inform. Transm.*, vol. 2, pp. 3–11, 1966.

On Binary Sliding Block Codes

TOBY BERGER, SENIOR MEMBER, IEEE, AND JOSEPH KA-YIN LAU

***Abstract*—Sliding block codes are an intriguing alternative to the block codes used in the development of classical information theory. The fundamental analytical problem associated with the use of a sliding block code (SBC) for source encoding with respect to a fidelity criterion is that of determining the entropy of the coder output. Several methods of calculating and of bounding the output entropy of an SBC are presented. The local and global behaviors of a well-designed SBC also are discussed. The so-called "101-coder," which eliminates all the isolated zeros from a binary input, plays a central role. It not only provides a specific example for application of the techniques developed for calculating and bounding the output entropy, but also serves as a medium for obtaining indirect insight into the problem of characterizing a good SBC. An easily implementable SBC subclass is introduced in which the outputs can be calculated by simple logic circuitry. The study of this subclass is shown to be closely linked with the theory of algebraic group codes.**

Manuscript received April 16, 1976. This work was supported in part by the National Science Foundation under Grant GK-41229 and by a John Simon Guggenheim Memorial Foundation Fellowship. Portions of this paper were first presented at the IEEE Information Theory Workshop, Lenox, Massachusetts, June 1975.

The authors are with the School of Electrical Engineering, Cornell University, Ithaca, NY 14853.

I. Introduction

SHANNON'S development of the theory of source coding subject to a fidelity criterion [1], [2] dealt almost exclusively with block coding, i.e., the mapping of consecutive, nonoverlapping, fixed-length blocks of source data into a so-called "code book" containing a constrained number of entries. The fundamental theorems of rate-distortion theory, which relate optimal source code performance to an information-theoretic minimization, involve complicated random coding arguments [3], [4] in general cases. Also, in many situations, block coding structures are exceedingly difficult to implement.