Construction of High Rate Run-Length Limited Codes Using Arithmetic Decoding

Andrey Fionov and Boris Ryabko
Siberian State University of Telecommunications and Information Sciences
Institute of Computational Technologies SB RAS
Novosibirsk, Russia
Email: a.fionov@ieee.org, boris@ryabko.net

***Abstract*—We suggest an approach to constructing low-redundant RLL (d,k)-codes whose complexity does not depend on the code length and is determined solely by the achievable redundancy r, the time and space complexity being $O(\log^2(1/r))$ and $O(\log(1/r))$, respectively, as $r \to 0$. First we select codewords whose combinations may constitute all (d,k)-constrained sequences of any length. Then we use arithmetic decoding to produce these codewords with (or close to) optimal probabilities from an input sequence. The coding algorithms and estimates of performance are provided.**

I. Introduction

In many data transmission and storage systems, e.g., those based on optical fibers or magnetic recording, there is a need in modulation codes with constraints on the maximum lengths of runs of consecutive like symbols. A binary run-length limited (RLL) (d,k)-code is defined as a uniquely decodable sequence of bits in which the length of any run of ones between zeros is at least d and at most k (sometimes zeros and ones interchange their roles, but this is only a matter of convention).

In the process of RLL encoding, an unconstrained input sequence of m bits, $m \geq 1$, is reversibly converted into a (d,k)-constrained sequence of n bits, $n > m$. The performance of an RLL code is usually measured by the code rate

$$R_n = m/n.$$

It is sufficient for fixed-to-fixed codes. For a more general class of variable-length RLL codes, it makes sense to define the limit code rate

$$R = \lim_{n\to\infty} R_n. \tag{1}$$

The maximum achievable code rate C, or coding capacity, is defined as

$$\lim_{n\to\infty} \frac{\log N(n)}{n}, \tag{2}$$

where $N(n)$ is the number of (d,k)-constrained sequences of length n. One can see that $R \leq C$ since it is not possible to represent $N(n)$ sequences by less than $\log N(n)$ bits. The redundancy of an RLL code may be defined as the quantity

$$r = C - R. \tag{3}$$

We are interested in constructing RLL codes with $r \to 0$.

The design of RLL codes has a long history, see [1] for a good survey. We can roughly divide the proposed methods into two categories: those based on enumeration of combinatorial objects and those using special ad-hoc coding constructions. Combinatorial methods are block-oriented and thus, as other block RLL codes, are subject to additional constraints in maximal runs of ones on the left and right sides of the block. This is essential for concatenated blocks to have a property of (d,k)-constrained sequences. The rates of enumerative codes are usually close to capacity with $r = O(1/n)$, where n is the block size. But their computational complexity is relatively high. The most efficient method of enumeration for (d,k)-constrained coding was recently suggested in [2], [3] with time and space complexities being $O(\log^3 n \log\log n)$ and $O(n)$, respectively, as $n \to \infty$.

Special ad-hoc coding constructions are numerous. One of the most recent work [4] suggests several methods of constructing block RLL codes with fixed rate $R_n = (n-1)/n$ by using a sequence replacement technique. Another paper [5] describes the same rate codes constructed from nibbles with the aid of numerical base conversion. The suggested codes are computationally efficient for small blocks, yet the redundancy is quite noticeable. For example, the highest achievable rate for $(0,3)$-codes from [4] (with left and right constraints) is $8/9 = 0.89$ for the maximal block size $n = 9$, whereas the capacity for this code is 0.95. So the redundancy is 0.06.

An attempt to attain performance of enumerative codes while preserving low computational complexity was made in [6] with a kind of approximate enumerative coding involving floating point computations. The method was further elaborated in a number of papers, e.g., [7]. The time complexity of these schemes is $O(n \log n)$ and space complexity is $O(n)$ as $n \to \infty$.

In this paper, we suggest an approach to constructing low-redundant RLL codes whose complexity does not depend at all on the code length n, so the time of encoding/decoding and the memory size are $O(1)$ as $n \to \infty$, and is determined solely by the achievable redundancy, the time and space complexity being $O(\log^2(1/r))$ and $O(\log(1/r))$, respectively, as $r \to 0$. The idea of construction is the following. We first select the codewords whose combinations may constitute all (d,k)-constrained sequences of any length. Then we use arithmetic decoding to produce these codewords with optimal (or close to optimal) probabilities from an input sequence.

It is essential for the proposed method that the bits of input unconstrained sequence be equiprobable and independent. It is

978-1-4673-0170-1/12/$31.00 ©2012 IEEE

a common assumption, see, e.g., [8], because in modern systems the data transmitted via fiber-optic channels or prepared for recording are usually compressed and/or encrypted and, hence, indistinguishable from completely random bits. On the other hand, if the input sequence is somehow biased and/or is not binary, we can add a special encoder as the first stage of the scheme which converts this input sequence into one needed for the main stage of our (d, k)-coding method. We shall not consider this scheme in details because its implementation is quite obvious.

The paper is organized as follows. In Section 2 we present the ideas of our method and prove its main properties. The aspects of practical implementation and achieved performance are considered in Section 3.

II. The Essence of the Method

A. Constructing a $(0, k)$-code

Consider the code alphabet $A = \{a_0, a_1, \ldots, a_k\}$ whose symbols (codewords) are the following binary sequences:

$$a_0 = 0,\ a_1 = 10,\ \ldots,\ a_k = \underbrace{1\ldots1}_{k}0.$$

It is clear that any combination of these codewords will produce a $(0, k)$-constrained code. It is also true that any $(0, k)$-constrained sequence can be represented as a word over A. For example, a $(0, 3)$-constrained sequence 0100110001110 can be represented as a word $a_0a_1a_0a_2a_0a_0a_3$. The mapping rule is simple:

$$\underbrace{1\ldots1}_{i}0 \longrightarrow a_i, \quad i \geq 0.$$

So the alphabet A can be used to construct all $(0, k)$-constrained infinite sequences.

Constructing sequences from the symbols of A is exactly the channel coding problem first considered by Shannon in [9]. The channel capacity of the code over A equals the coding capacity of $(0, k)$-code

$$C = \log X_0,$$

where X_0 is the greatest real solution of the characteristic equation

$$\frac{1}{X} + \frac{1}{X^2} + \cdots + \frac{1}{X^{k+1}} = 1. \tag{4}$$

Let $p = 1/X_0$ and set the probability distribution

$$P = \left[p_0 = p,\ p_1 = p^2, \ldots,\ p_k = p^{k+1}\right] \tag{5}$$

over A, i.e. $p_i = \Pr(a_i)$. Now we are ready to solve the $(0, k)$-coding problem: we shall encode the input sequence $u_1u_2u_3\ldots$ using the codewords of A in such a way that the codewords appear exactly according to probability distribution P. Let us show how to do that with the aid of source coding techniques.

A source encoder ϕ converts a message $x_1x_2x_3\ldots$ of symbols over some alphabet A with a specified probability distribution P into a binary code sequence $y_1y_2y_3\ldots$. We write this as

$$\phi_{A,P}(x_1x_2x_3\ldots) = y_1y_2y_3\ldots.$$

The corresponding decoder ϕ^{-1}, given the same A and P, recovers the source message from the code sequence:

$$x_1x_2x_3\ldots = \phi^{-1}_{A,P}(y_1y_2y_3\ldots).$$

Suppose that the encoder we are using is an entropy encoder, i.e., it compresses messages down to the entropy of P with zero redundancy. In this case any code bit y_i will be independent of other code bits and will appear with probabilities $\Pr(y_i = 0) = \Pr(y_i = 1) = 1/2$, which we call "completely random". Now if we feed the decoder with any sequence of completely random bits $u_1u_2u_3\ldots$, it will produce some sequence $v_1v_2v_3\ldots = \phi^{-1}_{A,P}(u_1u_2u_3\ldots)$ which inevitably will consist of symbols from A and obey the distribution P. And this is what we need to make our $(0, k)$-code. Of course, $\phi_{A,P}(v_1v_2v_3\ldots)$ will be equal to $u_1u_2u_3\ldots$ and by this way we can decode the $(0, k)$-code.

Denote by $H(P)$ the entropy of distribution P,

$$H(P) = -\sum_{i=0}^{k} p_i \log p_i.$$

Denote by $L(P)$ the average codeword length,

$$L(P) = \sum_{i=0}^{k} p_i(i + 1).$$

Consider the ratio $H(P)/L(P)$. It is the entropy of one code bit provided that the length of code sequence is not limited.

Proposition 1: For any probability distribution P the limit rate of $(0, k)$-code constructed by the described method

$$R = \frac{H(P)}{L(P)}.$$

Proof: Let the sequence $u_1\ldots u_m$ is encoded into $v_1\ldots v_n$. Since the encoding is one-to-one the entropies of both sequences are equal,

$$H(u_1\ldots u_m) = H(v_1\ldots v_n).$$

But with $n \to \infty$

$$H(v_1\ldots v_n) = n\frac{H(P)}{L(P)}.$$

According to our assumptions about the input sequence,

$$H(u_1\ldots u_m) = m.$$

So

$$\frac{m}{n} = \frac{H(P)}{L(P)}$$

as $n \to \infty$. ■

Proposition 2: For the probability distribution (5) the limit rate of $(0, k)$-code constructed by the described method equals the coding capacity, $R = C$.

Proof: According to Proposition 1, $R = H(P)/L(P)$. From (5) we have

$$\frac{H(P)}{L(P)} = \frac{-(p\log p + 2p^2\log p + \cdots + (k+1)p^{k+1}\log p)}{p + 2p^2 + \cdots + (k+1)p^{k+1}} = -\log p = \log X_0 = C.$$

■

B. Constructing a (d,k)-code

To construct a (d,k)-code for $0 < d < k$ consider the code alphabet $A = \{a_d, a_{d+1}, \ldots, a_k\}$ whose symbols (codewords) are the following binary sequences:

$$a_0 = 0\underbrace{1\ldots1}_{d},\ a_1 = 0\underbrace{1\ldots1}_{d+1},\ \ldots,\ a_k = 0\underbrace{1\ldots1}_{k}.$$

Combinations of these codewords will produce a (d,k)-constrained code.

To construct an optimal (d,k)-code, find the greatest real solution X_0 of the characteristic equation

$$\frac{1}{X^{d+1}} + \frac{1}{X^{d+2}} + \cdots + \frac{1}{X^{k+1}} = 1 \tag{6}$$

and set the probability distribution

$$P = \left[p_d = p^{d+1},\ p_{d+1} = p^{d+2}, \ldots,\ p_k = p^{k+1}\right]$$

over A, where $p = 1/X_0$. All further constructions and arguments are essentially the same as for $(0,k)$-code.

III. Practical Implementation

For practical implementation of our method we suggest to substitute an ideal entropy coder by arithmetic coder. Arithmetic coding is well-known for its extremely low redundancy at high computational efficiency. We use a method which is close to [10] and [11] with small optimizations relevant to our application. For the sake of consistency we describe the encoding and decoding algorithms. The main our topic is, however, to discuss the impact on redundancy of constructed RLL codes.

We start with the computation of X_0, the greatest real root of characteristic equation for the required code, and the corresponding probability distribution P. For ease of description, denote the size of alphabet A by s, $s = k + 1 - d$, and index the symbols and corresponding probabilities from 1 to s.

Compute from P the cumulative probabilities Q needed for arithmetic coding and defined as following:

$$Q_1 = 0, \quad Q_i = Q_{i-1} + P_{i-1},\ i = 2, 3, \ldots, s.$$

Define also $\hat{Q}_i = Q_i + P_i$. Then any symbol a_i is represented by the interval $[Q_i, \hat{Q}_i)$ in the cumulative distribution. Note that $\hat{Q}_s = 1$.

The next step is to approximate the cumulative probabilities by binary fractions of the form $q/2^t$, where t is the size (in bits) of internal registers used in coding. Since p is always greater than $1/2$, the least probability p^{k+1} can be represented with $k+1$ bits in the numerator. But as the interval where all symbols are to be distributed in arithmetic coding may be almost as small as 2^{t-2}, the size of registers must satisfy the condition $t - 2 \geq k + 1$, that is $t \geq k + 3$. Notice that the initial approximation of cumulative probabilities and their further implicit approximation via mapping onto integer-scaled intervals in arithmetic encoding, are the only causes of coding redundancy.

Let us now describe the algorithm of arithmetic coding. Introduce some operations: $a \operatorname{div} b$ denotes integer division with truncation; $\mathrm{msb}(b)$ denotes the most significant bit of b; $\mathrm{pmsb}(b)$ denotes the bit of b preceding to the most significant one. Denote by l and h the t-bit registers used to represent the lower and higher bounds of the current coding interval, h being inclusive, i.e., the greatest integer belonging to the interval. Denote by f a special counter used in the step of renormalization. Initially, $l = 0$, $h = 2^t - 1$, $f = 0$.

Encoding of each subsequent symbol is done by the following procedure. Let the current symbol be a_i and the cumulative interval be $[q_i, \hat{q}_i)$. Compute new interval bounds for this symbol:

$$h \leftarrow l + (\hat{q}_i(h - l + 1) \operatorname{div} 2^t) - 1,$$
$$l \leftarrow l + q_i(h - l + 1) \operatorname{div} 2^t.$$

Then perform renormalization of the interval with possible output of some code bits:

while $\mathrm{msb}(l) = \mathrm{msb}(h)$ do
  output bit $\mathrm{msb}(l)$,
  output f bits $1 - \mathrm{msb}(l)$, $f \leftarrow 0$,
  $l \leftarrow 2l,\ h \leftarrow 2h + 1 \pmod{2^t}$;
while $\mathrm{pmsb}(l) = 1$ and $\mathrm{pmsb}(h) = 0$ do
  $f \leftarrow f + 1$,
  $l \leftarrow 2l,\ h \leftarrow 2h + 1 \pmod{2^t}$;
$\mathrm{msb}(l) \leftarrow 0,\ \mathrm{msb}(h) \leftarrow 1$.

The described above encoding step may be applied arbitrary long. Nevertheless, if the encoded sequence is ended, we need to finalize the remaining interval: output $\mathrm{pmsb}(l)$, then output $f + 1$ bits $1 - \mathrm{pmsb}(l)$.

Let us now describe the algorithm of decoding. The decoder needs one extra t-bit register w to store current code bits. Initially, $l = 0$, $h = 2^t - 1$, $w =$ first t code bits.

The following algorithm decodes next one symbol at each invocation. Set

$$z \leftarrow (2^t(w - l + 1) - 1) \operatorname{div} (h - l + 1).$$

Find $a_i \in A$ for which $q_i \leq z < \hat{q}_i$. Compute new interval bounds:

$$h \leftarrow l + (\hat{q}_i(h - l + 1) \operatorname{div} 2^t) - 1,$$
$$l \leftarrow l + q_i(h - l + 1) \operatorname{div} 2^t.$$

Perform renormalization:

while $\mathrm{msb}(l) = \mathrm{msb}(h)$ do
  $w \leftarrow 2w +$ next code bit,
  $l \leftarrow 2l,\ h \leftarrow 2h + 1 \pmod{2^t}$;
$b \leftarrow \mathrm{msb}(w)$;
while $\mathrm{pmsb}(l) = 1$ and $\mathrm{pmsb}(h) = 0$ do

TABLE I
MAXIMUM REDUNDANCIES OF SOME $(0,k)$-CODES OBTAINED BY MODELING FOR $t=32$

k	max r	k	max r
4	0		
6	$8\cdot 10^{-17}$	8	$7\cdot 10^{-16}$
10	$2\cdot 10^{-15}$	12	$9\cdot 10^{-15}$
14	$4\cdot 10^{-14}$	16	$1\cdot 10^{-13}$
18	$6\cdot 10^{-13}$	20	$2\cdot 10^{-12}$

$$w \leftarrow 2w + \text{ next code bit},$$
$$l \leftarrow 2l,\ h \leftarrow 2h+1 \pmod{2^t};$$
$$\mathrm{msb}(w) \leftarrow b,\ \mathrm{msb}(l) \leftarrow 0,\ \mathrm{msb}(h) \leftarrow 1.$$

The decoded symbol is a_i.

Let us discuss the problem of redundancy. First, as it was already noted, we have to approximate the optimal probabilities by finite binary fractions which can be done with accuracy $1/2^t$. Second, operations div in encoder and decoder further slightly change these approximate values. Although we may to some extent control the way of initial approximation, the changes of probabilities in arithmetic coder are more intricate and cannot be effectively controlled. We can only state that the maximum error in each probability, which arises when the interval after renormalization has the smallest size, cannot be greater than $1/2^{t-2}$.

As a result, we have, in effect, another entropy $H(P^*)$ and another average codeword length $L(P^*)$. They cause the code rate different from capacity and redundancy of RLL code

$$r = \frac{H(P)}{L(P)} - \frac{H(P^*)}{L(P^*)}.$$

When we use the simplest approximation of probabilities by truncation, $H(P^*)$ becomes sometimes smaller and sometimes greater than $H(P)$ which hinders to derive an analytic estimate of redundancy. However, we can use a computer program to experimentally model all possible deviations for reasonably small alphabets, e.g., when $k \le 20$, $t = 32$, and inaccuracy is 2^{-29}. The results for $(0,k)$-codes are presented in Table 1.

An analytic estimate of redundancy can be easily obtained provided the following conjecture is true: the initial approximation of probabilities can be made within the accuracy of $1/2^{t-3}$ so that to ensure the condition $H(P^*) \ge H(P)$ regardless the deviations of the probabilities caused by the encoder. In view of this conjecture the estimate of redundancy is given by the following

Proposition 3: Provided that $H(P^*) \ge H(P)$ the redundancy of $(0,k)$-code satisfies the inequality

$$r < \frac{k^2+4k+3}{2^{t-2}}.$$

Proof: First find an upper bound for $L(P^*)$. For that suppose (pessimistically) that any $p^* < p + 1/2^{t-3}$. Then

$$L(P^*) = \sum_{i=0}^{k} p_i^*(i+1)$$
$$< \sum_{i=0}^{k} p_i(i+1) + \frac{1+2+\cdots+k+1}{2^{t-3}}$$
$$= L(P) + \frac{k^2+4k+3}{2^{t-2}} = \frac{2^{t-2}L(P) + k^2 + 4k + 3}{2^{t-2}}.$$

The following chain of equalities and inequalities proves the proposition:

$$r = C - \frac{H(P^*)}{L(P^*)} < C - \frac{H(P)2^{t-2}}{L(P)2^{t-2} + k^2 + 4k + 3}$$
$$= \frac{CL(P)2^{t-2} + C(k^2+4k+3) - H(P)2^{t-2}}{L(P)2^{t-2} + k^2 + 4k + 3}$$
$$= \frac{C(k^2+4k+3)}{L(P)2^{t-2} + k^2 + 4k + 3} < \frac{C(k^2+4k+3)}{L(P)2^{t-2}}$$
$$< \frac{k^2+4k+3}{2^{t-2}}.$$

■

The estimate of redundancy given by Proposition 3 seems to be quite pessimistic. Indeed, for $t = 32$ and $k = 20$ we calculate that $r < 4\cdot 10^{-7}$. The figures of Table 1 present more optimistic data.

Yet, the estimate of Proposition 3 enables us to speak of complexity. The complexity of encoding and decoding is determined by multiplications of t-bit integer numbers (all other operations are inferior). If we use square-time algorithms of multiplication (which is adequate for small t), the estimates of time and space complexities are:

$$T = O(t^2) = O(\log^2 1/r), \quad S = O(t) = O(\log 1/r), \quad r \to 0.$$

REFERENCES

[1] K. A. S. Immink, "A survey of codes for optical disc recording," *IEEE Journal on Selected Areas of Communications, Special Issue on Signal processing and coding for digital storage*, Vol. 19, pp. 751-764, 2001.

[2] Yu. Medvedeva and B. Ryabko, "Fast enumeration of run-length-limited words," *2009 IEEE Int. Symp. on Inform. Theory (ISIT2009)*, Seoul, Korea, 2009, pp. 640-643.

[3] Yu. S. Medvedeva and B. Ya. Ryabko, "Fast enumeration algorithm for words with given constraints on run lengths of ones," *Problems of Information Transmission*, Vol. 46, No. 4, pp. 369-378, 2010.

[4] A. van Wijngaarden and K. A. S. Immink, "Construction of maximum run-length limited codes using sequence replacement techniques," *IEEE Journal on Selected Areas of Communications, Special Issue on Signal processing and coding for digital storage*, Vol. 28, pp. 200–207, 2010.

[5] K. A. S. Immink, "High-rate maximum runlength constrained coding schemes using base conversion," *Proc. ISITA 2010*, Taichung, Taiwan, Oct. 2010.

[6] K. A. S. Immink, "A practical method for approaching the channel capacity of constrained channels," *IEEE Trans. Inform. Theory*, Vol. 43, No. 5, pp. 1389–1399, 1997.

[7] I. Tal, T. Etzion and R. M. Roth, "On row-by-row coding for 2-D constraints," arXiv: 0808.0596v1 [cs.IT], 2008.

[8] Y. Sankarasubramaniam and S. W. McLaughlin, "Fixed-rate maximum-runlength-limited codes from variable-rate bit stuffing," *IEEE Trans. Inform. Theory*, Vol. 53, No. 8, pp. 2769–2790, 2007.

[9] C. E. Shannon, "A mathematical theory of communication," *Bell Sys. Tech. J.*, vol. 27, pp. 379–423, pp. 623–656, 1948.

[10] I. H. Witten, R. Neal and J. G. Cleary, "Arithmetic coding for data compression," *Communications of the ACM*, Vol. 30, No. 6, pp. 520–540, 1987.

[11] B. Ya. Ryabko and A. N. Fionov, "An efficient method for adaptive arithmetic coding of sources with large alphabets," *Problems of Information Transmission*, Vol. 35, No. 4, pp. 95–108, 1999.