[8] C. Ding, G. Xiao, and W. Shan, *The Stability Theory of Stream Ciphers* (Lecture Notes in Computer Science, vol. 561). Heidelberg, Germany: Springer-Verlag, 1991.
[9] C. F. Gauss, *Disquisitiones Arithmeticae*, Leipzig, Germay, 1801; English translation: New Haven, CT, Yale Univ., 1966; reprint by Springer-Verlag, Berlin, Heidelberg, and New York, 1986.
[10] S. W. Golomb, *Shift-Register Sequences.* San Francisco, CA: Holden-Day, 1967; Laguna Hills, CA: Aegean Park, 1982.
[11] T. Helleseth and P. V. Kumar, "Sequences with low correlation," in *Handbook of Coding Theory*, V. Pless and W. C. Huffman, Eds. Amsterdam, The Netherlands: Elsevier, 1998.
[12] D. Jungnickel and A. Pott, "Difference sets: Abelian," in *The CRC Handbook of Combinatorial Designs*, C. J. Colbourn and J. H. Dinitz, Eds. New York: CRC, 1996, pp. 297–307.
[13] E. Lehmer, "On the number of solutions of $u^2 + D \equiv w^2 \pmod p$," *Pacific J. Math.*, vol. 5, pp. 103–118, 1955.
[14] S. L. Ma, "A survey of partial difference sets," *Des., Codes Cryptogr.*, vol. 4, pp. 221–261, 1994.
[15] D. V. Sarwate, "Crosscorrelation properties of pseudorandom and related sequences," *Proc. IEEE*, vol. 68, pp. 593–619, 1980.
[16] T. Storer, *Cyclotomy and Difference Sets.* Chicago, IL: Markham, 1967.
[17] A. L. Whiteman, "The cyclotomic numbers of order twelve," *Acta Arith.*, vol. 6, pp. 53–76, 1960.

Fast Coding of Low-Entropy Sources

Boris Ya. Ryabko and Marina P. Sharova

***Abstract*— The problem of coding low-entropy information sources is considered. Since the run-length code was offered about 50 years ago by Shannon, it is known that for such sources there exist coding methods much simpler than for sources of a general type. However, known coding methods of low-entropy sources do not reach the given redundancy. In this correspondence, a new method of coding low-entropy sources is offered. It permits a given redundancy r with almost the same encoder and decoder memory size as that obtained by Ryabko for general methods, while encoding and decoding much faster.**

***Index Terms*— Complexity of coding, fast algorithm, low-entropy sources, redundancy, run-length coding.**

I. Introduction

We consider the problem of low-entropy source coding whose elementary example is a Bernoulli source generating a sequence of zeros and ones with probabilities q and p, respectively, when $p \to 0$. This problem has attracted attention of many researchers, as for coding of such sources there exist simpler methods than in a general case. The efficiency of a code is measured by redundancy and by complexity of encoding and decoding. The redundancy r is the difference between the average codeword length and the Shannon entropy. Complexity is estimated by the memory size of the encoder and decoder (in bits) and by the average time of encoding and decoding one symbol measured by the number of binary operations on single-bit word when they are implemented on a computer with random-access memory (see the definition in [2]).

One of the well-known compression schemes of low-entropy sources is run-length coding [1]. In this method, a sequence of symbols generated by a source is broken into runs of zeros between two sequential ones: 1, 01, 001, etc., then the lengths of the runs are encoded by the binary codewords. The length of a run can thus be both limited and unlimited.

In coding with unlimited length of runs the scheme offered by Shannon [1] can be used. According to this scheme, one codeword is selected for the least probable symbol 1. For encoding lengths of runs binary words are picked in ascending order, bypassing the word selected for 1. Shannon has proved that by increasing the length of the codeword designating 1 when $p \to 0$ the redundancy of coding tends to zero. It is possible to show that it does not exceed $C_1 p \log(1/p)$, where $C_1 \leq 1$ is a constant.

In [3] Elias proposes to use a prefix code of integers for run-length coding. Elias has constructed three new universal binary representations of integers and by using them has constructed universal codeword sets. For the best representation of integers from [3] the redundancy of it the given code reaches $C_2 p \log\log(1/p)$, where C_2 is a constant.

An effective run-length coding method was offered by Golomb [4]. In [5] it was shown that for particular values of a run-length coding scheme Golomb's code is optimal.

However, the known methods of coding low-entropy sources [1], [3]–[5] do not allow reaching the given redundancy. In this correspondence, a new method of coding low-entropy sources is offered. It permits reaching a given redundancy r with almost the same encoder and decoder memory size as obtained in [6] for general methods, while encoding and decoding is much faster.

Here we consider a problem of coding a Bernoulli source with known statistics. Note that the offered code construction is applicable also for the Bernoulli sources with unknown statistics and for more complex models.

II. Algorithm of Coding Low-Entropy Sources

Let a Bernoulli source generating a sequence of zeros and ones with probabilities q and p, respectively, when $p \to 0$, be given. Let $r > 0$ be the given redundancy of a code. Our problem is to construct a method of source coding permitting us to reach the given redundancy r.

In our method encoding is implemented in two stages: first, a message is compressed by a simple code and an output sequence is then encoded by a fast and effective code. After the first stage the length of the input sequence is essentially reduced, and applying a complex fast algorithm at the second stage provides little total time of encoding and decoding per letter of the initial message. At the second stage it is possible to use many codes, for example, the arithmetic code [7], [8] or the code from [6]. We shall use the code from [6] since it has the estimates of the average time and memory. Note, however, that the use of some versions of the universal arithmetic code gives the same result. For code from [6] the dependence of the memory size V and the average time T of encoding and decoding of one letter on the redundancy r' as $r' \to 0$ satisfies the following

Manuscript received February 1, 1998; revised January 14, 1999. The material in this correspondence was presented in part at the IEEE International Symposium on Information Theory, Cambridge, MA, August 1998.

B. Ya. Ryabko is with the Siberian State Academy of Telecommunications and Computer Science, 630102 Novosibirsk, Russia.

M. P. Sharova is with the Novosibirsk State University, 630102 Novosibirsk, Russia.

Communicated by I. Csiszár, Associate Editor for Shannon Theory.
Publisher Item Identifier S 0018-9448(99)07675-0.

estimates:

$$V = O\left(\frac{1}{r'}\log\frac{1}{r'}\right)$$
$$T = O\left(\log^3\left(\frac{1}{r'}\right)\log\log\frac{1}{r'}\right). \quad (1)$$

Our method will require memory size as above, and a significantly smaller average time of encoding and decoding.

Let us describe the offered method of coding in more detail. Let it be required to reach some redundancy r, $r > 0$. We divide the input message into blocks of the lengths $l = \lceil 1/\sqrt{p}\rceil$. If a block consists completely of zeros, then its code is zero. Otherwise, we encode it according to the following rule: the first letter of the codeword is 1 followed by the same block of the length l. The length $l = \lceil 1/\sqrt{p}\rceil$ is chosen such that the average codeword length (per source letter) obtained after the first stage is minimal. The blocks do not need to be stored in memory as it is enough to store only counters of zeros, whose lengths are $O(\log 1/p)$. Note that compression at the first stage is achieved only if the source is low-entropy. Otherwise, the method does not reduce the length of message and does not simplify encoding.

For example, let $p = 1/8$ and the sequence

$$0010000000001100000000000$$

be coded. Then $l = 3$, and the encoded sequence looks like 1001 0 0 0 1110 0 0 0 (gaps are used for convenience of reading).

Let now $y_1y_2\cdots y_t$ be a sequence obtained after the first stage of encoding, $y_i \in A$, $A = \{0, 1\}$. Consider the second stage of encoding realized by a fast code from [6]. Note that the sequence $y_1y_2\cdots y_t$ cannot already be considered as a Bernoulli one, therefore, we offer a new method. First, for convenience, we present a sequence $y_1y_2\cdots y_t$ as

$$\underline{0}\cdots\underline{0}\,\underline{1}\,\underbrace{y_1\cdots y_l}_{l}\,\underline{0}\cdots\underline{0}\,\underline{1}\,\underbrace{y_1\cdots y_l}_{l}\cdots.$$

In this sequence we mark "blocks" of length l ($l = \lceil 1/\sqrt{p}\rceil$) following after appearance of 1, and "special" symbols 0 and 1 not included in "blocks" (in the sequence "special" symbols are underlined).

Encoding of various y_i is implemented with the help of various encoders "tuned" to various probabilities of appearance of zeros and ones and is in the following.

The "special" symbols 0 and 1 are encoded with the help of an encoder K_0 with probabilities q^l and $(1-q^l)$ for 0 and 1, respectively.

Let us consider encoding of symbols inside the "block" $y_1\cdots y_l$ of length l. Let

$$y_1\cdots y_{i-1} = \underbrace{0\cdots0}_{i-1} \qquad (i = 1, 2, \cdots, l).$$

Then symbol y_i following $(i-1)$ zeros is encoded with the help of an encoder K_i with probabilities π_i, if $y_i = 1$, or $(1-\pi_i)$, if $y_i = 0$, where

$$\pi_i = \frac{1-q}{1-q^{l-i+1}} = \frac{1}{1+q+q^2+\cdots+q^{l-i}}. \quad (2)$$

Let us explain the appearance of these probabilities. In fact,

$$\pi_i = P\left(y_i = 1 | y_1\cdots y_{i-1} = 0\cdots0;\ \sum_{i=1}^{l} y_i > 0\right)$$
$$= \frac{P\left(y_i = 1,\ y_1\cdots y_{i-1} = 0\cdots0 \middle| \sum_{i=1}^{l} y_i > 0\right)}{P\left(y_1\cdots y_{i-1} = 0\cdots0 \middle| \sum_{i=1}^{l} y_i > 0\right)}$$
$$= \frac{1-q}{1-q^{l-i+1}}$$

since

$$P\left(y_i = 1,\ y_1\cdots y_{i-1} = 0\cdots0 \middle| \sum_{i=1}^{l} y_i > 0\right)$$
$$= \frac{P\left(y_i = 1,\ y_1\cdots y_{i-1} = 0\cdots0;\ \sum_{i=1}^{l} y_i > 0\right)}{P\left(\sum_{i=1}^{l} y_i > 0\right)}$$
$$= \frac{p\cdot q^{i-1}}{1-q^l}$$

and

$$P\left(y_1\cdots y_{i-1} = 0\cdots0 \middle| \sum_{i=1}^{l} y_i > 0\right)$$
$$= \frac{P\left(y_1\cdots y_{i-1} = 0\cdots0;\ \sum_{i=1}^{l} y_i > 0\right)}{P\left(\sum_{i=1}^{l} y_i > 0\right)}$$
$$= \frac{q^{i-1}\cdot(1-q^{l-(i-1)})}{1-q^l}.$$

In addition,

$$P\left(y_i = 0 | y_1\cdots y_{i-1} = 0\cdots0;\ \sum_{i=1}^{l} y_i > 0\right) = 1-\pi_i.$$

The symbols following 1 in the "block" $y_1\cdots y_l$ are encoded with the help of an encoder K with initial probabilities q and p for 0 and 1, respectively.

It is important to note that the probabilities π_i are not stored in the encoder and the decoder memory (it would require too much memory), but they are recursively calculated. First, y_1 is encoded with probability π_1 (if $y_i = 1$) or $(1-\pi_1)$ (if $y_i = 0$), and π_1 is stored in memory (it is calculated prior to the beginning of encoding). If $y_1 = 1$, then all the symbols following y_1 are encoded with probabilities q and p for 0 and 1, respectively. Otherwise, π_2 is calculated and y_2 is encoded with probability π_2 (if $y_2 = 1$) or $(1-\pi_2)$ (if $y_2 = 0$). If $y_2 = 1$, then the symbols following y_2 are encoded with probabilities q and p for 0 and 1, respectively. Otherwise, π_3 is calculated and y_3 is encoded with probability π_3 or $(1-\pi_3)$, etc. Formally, the algorithm of encoding can be described as follows:

Step 1. π_i is calculated and y_i is encoded with the probability π_i, if $y_i = 1$, or $(1-\pi_i)$, if $y_i = 0$.

Step 2. If $y_i = 1$, then all the symbols following y_i are encoded with initial probabilities q and p for 0 and 1, respectively. Otherwise, turn to the following symbols and return to Step 1.

$$\begin{array}{cccccccccccccc} \underline{1} & \underbrace{0} & \underbrace{0} & \underbrace{1} & \underline{0} & \underline{0} & \underline{0} & \underline{1} & \underbrace{1} & \underbrace{1} & \underbrace{0} & \underline{0} & \underline{0} & \underline{0} \\ y_1 & y_2 & y_3 & y_4 & y_5 & y_6 & y_7 & y_8 & y_9 & y_{10} & y_{11} & y_{12} & y_{13} & y_{14} \\ K_0 & K_1 & K_2 & K_3 & K_0 & K_0 & K_0 & K_0 & K_1 & \hat{K} & \hat{K} & K_0 & K_0 & K_0. \end{array}$$

Note (see (2)) that

$$\frac{1}{\pi_2} = \frac{1}{\pi_1} - q^{l-1}$$
$$\frac{1}{\pi_3} = \frac{1}{\pi_2} - q^{l-2}$$

etc. Therefore, the probabilities π_i may be calculated under the following recursive formula:

$$\frac{1}{\pi_{i+1}} = \frac{1}{\pi_i} - q^{l-i}.$$

Thus the calculation can be arranged under the following scheme:

$$\begin{aligned} \hat{q} &:= \hat{q}/q \\ \hat{\pi}^{-1} &:= \hat{\pi}^{-1} - \hat{q} \end{aligned} \tag{3}$$

with the initial data

$$\begin{aligned} \hat{q} &:= q^l \\ \hat{\pi}^{-1} &:= 1 + q + \cdots + q^{l-1}. \end{aligned}$$

The following "block" is encoded in the same way, and before encoding of each new "block" new initial data are formed.

Note that after the first stage, the sequence $y_1 \cdots y_t$ can be uniquely decoded. Actually, for every "special" symbol 0 there is a specific block

$$\underbrace{0 \cdots 0}_{l}$$

of initial sequence, but for every "block" of length l following after appearance of 1 there is the same block of the length l of initial sequence.

In order to reach necessary redundancy r in method from [6], all the probabilities must be presented the numbers with $t = \lceil \log(1/r) + 3 \rceil$ binary digits. To reach this accuracy when presenting the probabilities π_i it is sufficient to do calculations with numbers which are presented by words of length

$$\hat{t} = \lceil 4 + \log(1/r) + 2 \log l \rceil = \lceil 4 + \log(1/r) + \log(1/p) \rceil.$$

It can be easily shown that $\hat{t}$ bits are sufficient to attain necessary precision for representation of π_i.

It should be noted that the calculation time π_i does not exceed $C(\log^2(1/r) + \log^2(1/p))$, where C is a constant (we make use one subtraction, one multiplication, and two division of the numbers, which are presented by words of length $\log(1/r)$ bits).

Consider the described construction on the previous example. Let $p = 1/8$, $q = 7/8$, $l = 3$, and the sequence be the following matrix (see the top of this page). Then the "special" symbols are encoded by the encoder K_0 according to the probabilities

$$\begin{aligned} p(y_1) = p(y_8) &= 1 - (7/8)^3 = 169/512, \\ p(y_5) = p(y_6) &= p(y_7) = p(y_{12}) = p(y_{13}) \\ &= p(y_{14}) = (7/8)^3 = 343/512. \end{aligned}$$

According to (3) the symbol y_2 of the first "blocks" $y_2y_3y_4$ is encoded by the encoder K_1 with the probability

$$1 - \pi_1 = 1 - 1/(1 + 7/8 + (7/8)^2) = 105/169.$$

As $y_2 = 0$ we encode y_3 by the encoder K_2 with the probability

$$1 - \pi_2 = 1 - 1/(1 + 7/8) = 7/15.$$

As $y_3 = 0$ we encode y_4 by the encoder K_3 with the probability $\pi_3 = 1$.

The symbol y_9 of the second "blocks" $y_9y_{10}y_{11}$ is encoded by the encoder K_1 with the probability

$$\pi_1 = 1/(1 + 7/8 + (7/8)^2) = 64/169.$$

As $y_9 = 1$ the symbols y_{10} and y_{11} are encoded by the encoder $\hat{K}$ with probabilities $p(y_{10}) = 1/8$, $p(y_{11}) = 7/8$.

Thus the probabilities only for three encoders K_0, K_i, $\hat{K}$, one of which is variable, and counters of zeros of the size $O(\log(1/p))$ are stored in memory. Hence, the given method has the same memory of the encoder and the decoder as "general" methods. As "general" methods we understand the codes intended for coding sources of any kind (not only low-entropy sources). Here as a "general" method we use the code from [6].

The properties of this method are characterized by the following theorem.

Theorem: Let there be given a Bernoulli source generating a sequence of zeros and ones with probabilities q and p $(p < 1/2)$, respectively, and $r > 0$. Let the above-described code with $l = \lceil 1/\sqrt{p} \rceil$ at the first stage and the redundancy $\overline{r} = r/2$ at the second stage be used. Then the general redundancy of the code does not exceed r, and the memory size V of the encoder and decoder and the average time T of encoding and decoding of one symbol satisfies the following inequalities:

$$V < \frac{C_1}{r} \log\left(\frac{1}{r}\right)$$
$$T < C_2 \sqrt{p} \log^3\left(\frac{1}{rp}\right) \log\log\left(\frac{1}{rp}\right) + C_3$$

where C_1, C_2, and C_3 are constants.

Proof: The coding is constructed by the following scheme:

$$S \to S' \to S''$$

where S is a word of length l, S' is a word of length l_1, obtained after the first stage of coding, S'' is a word of length l_2 obtained after the second stage of coding. Let h_0 and h_1 be the original entropy and the entropy after the first stage, respectively. Since after the first stage the sequence can be uniquely decoded, the entropies of the source and encoded message must be equal, i.e.,

$$h_0 l = h_1 l_1.$$

Then

$$h_1 = h_0 \frac{l}{l_1}. \tag{4}$$

By definition, the redundancy $\overline{r}$ at the second stage is of the form

$$\overline{r} = \frac{l_2}{l_1} - h_1. \tag{5}$$

The general redundancy R is of the form

$$R = \frac{l_2}{l} - h_0. \tag{6}$$

From (4)–(6) we have

$$R = \frac{l_2}{l} - h_0 = \frac{l_2}{l_1} \cdot \frac{l_1}{l} - h_0 \frac{l}{l_1} \cdot \frac{l_1}{l} = \frac{l_1}{l}\left(\frac{l_2}{l_1} - h_0 \frac{l}{l_1}\right)$$
$$= \frac{l_1}{l}\left(\frac{l_2}{l_1} - h_1\right) = \frac{l_1}{l}\,\overline{r} = l'\overline{r}$$

where $l' = l_1/l$ is the average codeword length (per source letter) obtained after the first stage. Let us show that

$$l' < 2\sqrt{p} + p. \tag{7}$$

Actually, as the probability of appearance of a block consisting of zeros only is equal to $q^l = (1-p)^l$, then

$$l' = \frac{1}{l}\left(q^l + (l+1)(1-q^l)\right) = 1 - q^l + \frac{1}{l}$$
$$= 1 - (1-p)^l + \frac{1}{l} \le lp + \frac{1}{l} < p\left(\frac{1}{\sqrt{p}} + 1\right) + \sqrt{p}$$
$$= 2\sqrt{p} + p$$

that coincides with (7). (Here we have taken advantage of the known inequality $(1+x)^n \ge 1 + nx (x > -1)$.)

Therefore, when $p < 1/2$

$$R = l'\overline{r} < \frac{r}{2}(2\sqrt{p} + p) < r$$

i.e., the general redundancy does not exceed r.

Evaluation of the average time T is based on summing up the number of binary operations performed in encoding and decoding. At the first stage the time of coding is equal to $O(1)$. According to (1) the time of encoding of one symbol of the sequence compressed at the first stage is equal to $O(\log^3(1/R)\,\log\,\log\,(1/R))$. As noted above, the calculation time π_i does not exceed $C(\log^2(1/r) + \log^2(1/p))$. Multiplying the total time of encoding at the second stage by the average codeword length l' we obtain that the time of encoding of one symbol satisfies the inequality

$$T < C_2\sqrt{p}\,\log^3\left(\frac{1}{rp}\right)\log\log\left(\frac{1}{rp}\right) + C_3$$

where C_2, C_3 are constants. The time of decoding is determined similarly.

REFERENCES

[1] C. Shannon, "A mathematical theory of communication," *Bell Syst. Tech. J.*, vol. 27, no. 3, pp. 379–423, 1948.

[2] A. V. Aho, J. E. Hopcroft, and J. D. Ullman, *The Design and Analysis of Computer Algorithms.* Reading, MA: Addison-Wesley, 1976.

[3] P. Elias, "Universal codeword sets and representation of the integers," *IEEE Trans. Inform. Theory*, vol. IT-21, pp. 195–203, Mar. 1975.

[4] S. W. Golomb, "Run length encodings," *IEEE Trans. Inform. Theory*, vol. IT-12, pp. 399–401, July 1966.

[5] R. G. Gallager and D. C. Van Voorhis, "Optimal source codes for geometrically distributed integer alphabets," *IEEE Trans. Inform. Theory*, vol. IT-21, pp. 228–230, Mar. 1975.

[6] B. Y. Ryabko, "Fast and effective coding of information sources," *IEEE Trans. Inform. Theory*, vol. 40, pp. 96–99, Jan. 1994.

[7] J. Rissanen, "Arithmetic coding as number representations," *Acta Polytec, Scand.*, vol. 31, pp. 44–51, 1979.

[8] J. Rissanen and G. G. Langdon, "Universal modeling and coding," *IEEE Trans. Inform. Theory*, vol. IT-27, pp. 12–23, Jan. 1981.

On Channel-Constrained Vector Quantization and Index Assignment for Discrete Memoryless Channels

Mikael Skoglund, *Member, IEEE*

***Abstract*—This correspondence introduces an approach to the design of channel robust vector quantizers. Design criteria are derived based on a new expression for the channel distortion as a function of the codevectors of the quantizer. The introduced framework for design and analysis holds for arbitrary discrete memoryless channels. Simulations demonstrate good performance. The correspondence also presents new results regarding the channel distortion as a function of the index assignment given to the codevectors of a vector quantizer.**

***Index Terms*—Channel distortion, channel robust source coding, combined source-channel coding, discrete memoryless channel, Hadamard transform, index assignment, vector quantization.**

I. INTRODUCTION

Channel robust source coding, with an emphasis on vector quantization (VQ)[1] for noisy channels, is a research topic that has drawn much recent attention [1]–[10]. The existing methods for VQ over noisy channels can be divided, roughly, into two main groups. In the first, referred here to as *robust* VQ (RVQ), a VQ trained for a noiseless channel is made robust toward channel errors by means of an index assignment (IA) algorithm [4], [6], [7], [11], [12]. In the second, referred to as *channel-optimized* VQ (COVQ), the VQ is designed assuming a known channel, taking the resulting channel distortion into consideration in the design [1]–[3], [5], [7], [8]. Ideally, the runtime channel parameters should then agree exactly with the ones assumed during training. In this correspondence we also wish to consider a third class of methods, here referred to as *channel-constrained* VQ (CCVQ), where the VQ is trained for minimum quantization distortion under a constraint determined by the channel [13]–[15]. The aim of the design is to provide the quantizer with "inherent" channel robustness. The code is then applied as an ordinary VQ, that is, as if the channel were noiseless.

In the design and analysis of VQ transmission for noisy channels, expressions for the resulting *channel distortion* (that is, the part of the total distortion due to channel errors) are helpful tools. Most previous work regarding analytical expressions for the channel distortion (see, e.g., [9], [10], [12]–[14], and [16]–[20]) requires the two assumptions that: i) the channel is *binary and symmetric*, and that ii) the transmitted VQ indices are *equiprobable* [17], [19], corresponding to *full encoder entropy* [10], [12], [14], [20]. Some results not relying on the full entropy assumption ii) can be found in [10] and [19], and results for binary asymmetric channels can be found in [19]. However, no previous results valid for nonbinary

Manuscript received November 1, 1998; revised May 1, 1999. The material in this correspondence was presented in part at the 1996 International Symposium on Information Theory and its Applications (ISITA), Victoria, BC, Canada, September 1996.

The author is with the Department of Signals, Sensors, and Systems, Royal Institute of Technology, SE-100 44 Stockholm, Sweden (e-mail: skoglund@s3.kth.se).

Communicated by P. A. Chou, Associate Editor for Source Coding.

Publisher Item Identifier S 0018-9448(99)07314-9.

[1] We will employ the acronym "VQ" as meaning both "vector quantization" and "vector quantizer."