

Transactions Letters

Design Techniques for Weakly Constrained Codes

Ming Jin, *Member, IEEE*, K. A. S. Immink, *Fellow, IEEE*, and B. Farhang-Boroujeny, *Senior Member, IEEE*

Abstract—A general method of constructing run-length limited (d, k) constrained codes from arbitrary sequences is introduced. This method is then combined with the method of guided scrambling for constructing a class of weakly constrained codes. The proposed codes are analyzed for the case of $d = 0$ and are shown to give results which are better or comparable to those of the best available codes, however, at the cost of failure with some very low probability. For $d > 0$, the code efficiency of the codes constructed according to the proposed method reduces significantly.

Index Terms—Bit stuffing, guided scrambling, run-length limited (RLL) code, weakly constrained codes.

I. INTRODUCTION

CODES BASED on run-length limited (RLL) sequences have found wide application in data storage products [4], [5]. RLL sequences are characterized by two parameters, d and k , respectively, which show minimum and maximum number of zeros allowed between two consecutive ones. We recall that a one corresponds to a change of direction of flux on a magnetic medium, and a zero means no change of flux. The d constraint is applied to reduce the rate of flux change on the magnetic medium, in order to cope with the maximum allowable flux change along the recording track. The k constraint, on the other hand, is dictated by timing recovery considerations [7], [8].

Realization of RLL codes requires manipulation of the original source sequence to satisfy the d and k constraints. This naturally requires addition of some redundancy in the sequence. This is commonly done by first partitioning the source sequence into blocks of length m . Each block then undergoes a mapping which, according to the coding rules, generates a block of $n > m$ symbols for transmission, resulting in a code rate of $R = m/n$. The maximum achievable value of R is given by the noiseless capacity, C , of the constrained channel [4], [5]. The code efficiency is defined as $\eta = R/C$.

Paper approved by E. Ayanoglu, the Editor for Communication Theory and Coding Application of the IEEE Communications Society. Manuscript received February 26, 2001; revised October 23, 2001 and May 19, 2002. This paper was presented in part at the IEEE GLOBECOM'01, San Antonio, TX, November 25–29, 2001.

M. Jin was with the Department of Electrical Engineering, National University of Singapore, Singapore. He is now with Seagate Technology International, Singapore 118249, Singapore.

K. A. S. Immink is with the Institute for Experimental Mathematics, D453261 Essen, Germany.

B. Farhang-Boroujeny was with the Department of Electrical Engineering, National University of Singapore. He is now with the Department of Electrical Engineering, University of Utah, Salt Lake City, UT 84112-9206 USA (e-mail: farhang@ee.utah.edu).

Digital Object Identifier 10.1109/TCOMM.2003.811377

The idea of *weakly constrained codes* was first proposed by Immink [3]. Weakly constrained codes follow the code rules most of the time, but not always. Code violation occurs with a low probability, and this can be made arbitrarily low by reducing the code rate. It is argued that as the channel is not free of errors, failure to satisfy the code constraints with a probability significantly less than the probability of errors due to the channel imperfections will not result in any significant degradation of the system performance.

Immink [3] presented a performance analysis of weakly constrained codes in a rather general context. He demonstrated that repetitive independent scrambling of the original data will give a set of code words which comprises, with a very high probability, at least one member that complies with the code constraint.

In this letter, we first introduce a systematic method of generating a class of variable-length (VL) RLL(d, k) constrained codes. VL codes were pioneered by Franaszek in the early 1970s; see [4, Sec. 7.2]. The VL codes presented by Franaszek generate encoded blocks of fixed length, and are therefore named *synchronous* VL codes. Here, we study asynchronous VL codes, which are made synchronous (i.e., they have the required feature of fixed output block length) by adding a number of dummy bits. As the proposed encoding algorithm cannot fully guarantee the fixed-length requirement without disobeying the maximum-runlength constraint, this leads to a class of *weakly* constrained codes. The encoding algorithm is improved significantly by using a procedure called *guided scrambling* first introduced by Fair *et al.* [1].

The long block codes considered in this letter may suffer from severe error propagation as, in worst-case situations, entire decoded words can be in error as the result of a single channel-bit error. With the coding configuration developed by Bliss (see [4, Sec. 6.3] and further) serious error propagation of long block codes can be effectively combated. We assume that our newly developed block code is used in conjunction with such a Bliss-type encoder configuration, and that as a result, the error propagation of long block codes does not play a role.

II. VL RLL(d, k) CONSTRAINED CODES

We consider a simple method for constructing (d, k) constrained codes where bit stuffing [4]–[6] is used to translate arbitrary data into constrained sequences. We assume $k \geq d$. The encoding is done in two steps. In the first step, the encoder scans the input data sequence, a_n , and inserts (stuffs) a 1 after every string of $(k - d)$ consecutive 0's such that the output sequence, b_n , satisfies the RLL ($0, k - d$) constraint. In the second step, a string of d consecutive 0's are inserted after every 1 in b_n .

TABLE I
INPUT BIT PATTERNS (PHRASES) AND THE PROBABILITY
OF THEIR OCCURRENCE IN A RANDOM SEQUENCE

category	index (i)	phrase	p_i
I	0	1	2^{-1}
	1	01	2^{-2}
	\vdots	\vdots	\vdots
	i	$0^i 1$	2^{-i-1}
	\vdots	\vdots	\vdots
	k-d-1	$0^{k-d-1} 1$	2^{-k+d}
II	k-d	0^{k-d}	2^{-k+d}

This results in an output sequence c_n satisfying the (d, k) constraint. The decoding is simply done by removing the inserted bits, which as can be verified, can be done in a unique fashion. We note that the number of inserted bits, and thus, the length of coded data, depends on the input, thus, the name variable length.

To analyze the above bit-stuffing method, we assume the input data bits, a_n , are independent and 0's and 1's are equally probable. In addition, we identify two categories of bit patterns (phrases) and their probability of occurrence, p_i , in the input data as listed in Table I. We note that in the first step of bit stuffing, no action will be taken on the phrases in Category I. However, the bit phrase in Category II is expanded by one bit. We also note that the average phrase length in a random input sequence is

$$\bar{T} = \sum_{i=0}^{k-d-1} (i+1)p_i + (k-d)p_{k-d} = 2(1 - 2^{-k+d}). \quad (1)$$

Moreover, the average inserted bits in the first step, and the average phrase length in the bit-stuffed sequence b_n are $\bar{t} = 2^{-k+d}$, and $\bar{T} + \bar{t}$, respectively. The average code rate of b_n is, thus

$$\bar{R}(0, k-d) = \frac{\bar{T}}{\bar{T} + \bar{t}} = \frac{2^{k-d+1} - 2}{2^{k-d+1} - 1}. \quad (2)$$

Hence, assuming that a_n contains m_a bits, the average length of b_n will be $\bar{m}_b = m_a / \bar{R}(0, k-d)$ bits.

In the second step, the encoder inserts a string of d consecutive 0's after every 1 in b_n . Noting that the average number of 0's in b_n is similar to a_n and is equal to $m_a/2$, the average length of c_n is

$$\bar{m}_c = \bar{m}_b + \left(\bar{m}_b - \frac{m_a}{2} \right) d = \frac{m_a(d+2)}{2} \frac{2^{k-d} - \frac{1}{d+2}}{2^{k-d} - 1}. \quad (3)$$

We obtain the average code rate of c_n as

$$\bar{R}(d, k) = \frac{m_a}{\bar{m}_c} = \frac{2}{d+2} \frac{2^{k-d} - 1}{2^{k-d} - \frac{1}{d+2}}. \quad (4)$$

To evaluate the code efficiency of the above method, we may compare the numerical results derived from (4) with the theoretical bounds (i.e., the noiseless capacity of the constrained channel) and also those of the well-known industry-standard RLL (d, k) codes. For example, from (4), we obtain $\bar{R}(1, 7) = 0.65969$ and $\bar{R}(2, 7) = 0.48819$. These values should be compared against the maxentropic capacity values 0.6793 and 0.5174, respectively. We may also notice that the respective values for the well-known industry-standard

codes are 2/3 and 1/2. Another good example is RLL $(0, k)$ codes. From (4), for $k \gg 1$, we get

$$\bar{R}(0, k) \approx 1 - \frac{1}{2} 2^{-k}.$$

This also is very close to the capacity

$$C(0, k) \approx 1 - \frac{1}{2.77} 2^{-k}.$$

III. FIXED-LENGTH WEAKLY CONSTRAINED CODES

Although the above bit stuffing is attractive as an inexpensive-to-implement method, it may be found impractical because of its serious problem of *error propagation*. A single bit error in the decoded data may corrupt the rest of the data sequence. To resolve this problem, we propose a fixed-length weakly constrained code, using the above bit-stuffing method. The discussion in this section is limited to RLL $(0, k)$ codes. The reason that we limit the discussion to RLL $(0, k)$ codes is that the proposed weakly constrained codes will only give good results when $d = 0$. When $d > 0$, addition of d 0's after each 1 without any consideration of the subsequent information bits, as done in this letter, reduces the code efficiency of the constructed d constrained codes significantly.

We divide the input sequence into blocks of m bits each. Each block is coded separately into $n = m + r$ bits. Thus, a maximum of r stuffed bits is allowed in each block. If the number of the required stuffed bits is less than r , the additional bits will be filled up with dummy bits, which will be ignored in the decoding process. If the number of the required stuffed bits is larger than r , constraint failure will occur. The success of this method should be assessed by studying the probability of code failure (PCF). We, therefore, proceed with such a study. A possible study which may lead to an exact expression for PCF involves use of multinomial distribution functions [2]. However, our attempt along this line has led to cumbersome equations which are hard to handle. On the other hand, analysis based on some approximations gives results which match computer simulations very closely. We proceed with two analysis methods of this type. The first method is an oversimplified analysis, which within the range of interest gives an upper bound to the PCF. It also serves as a means of more in-depth understanding of the problem, and it paves the way for the development of more accurate results in the second method. The analytical PCF formulas developed by the second method match very closely with simulations.

A. Analysis Based on Single Phrase Length

In a RLL $(0, k)$ constrained code that is constructed according to the procedure discussed in Section II, the phrase length is a random variable which takes values of 1 to $k + 1$ with the respective probabilities listed in Table I. To simplify the analysis, we assume that all phrases are of equal length \bar{T} , where \bar{T} is the average phrase length. Accordingly, in a block of length m , there are $M = m/\bar{T}$ phrases, and the probability distribution function (PDF) of a number of Category II phrases, x , in a given block follows the binomial distribution function [2]

$$f_1(x) = \binom{M}{x} p^x (1-p)^{M-x} \quad (5)$$

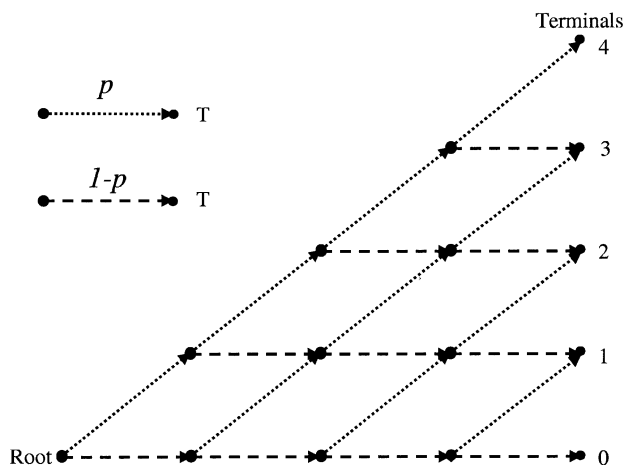


Fig. 1. Trellis diagram explanation for binomial distribution based on single-phrase approach.

where $p = p_k = 2^{-k}$ is the probability that a chosen phrase belongs to Category II, and the first term on the right-hand side is the binomial coefficient/combinatorial number

$$\binom{M}{x} = \frac{M!}{x!(M-x)!}.$$

The PCF is then given by

$$p_f = \sum_{x=r+1}^M f_1(x). \quad (6)$$

A more in-depth understanding of (5) is obtained by referring to the trellis diagram depicted in Fig. 1. The horizontal (dashed) and diagonal (dotted) arrows indicate occurrences of phrases from Category I and Category II, respectively. Attached to each arrow is the probability of occurrence of a phrase from the respective category. For convenience, we may imagine that the phrases in the data block are selected successively from left to right, as they appear in the trellis. The trellis begins with a root and ends at a number of terminals. Each terminal may be reached through a number of different paths. So, attached to each terminal, there is a set of paths. Each terminal is highlighted with a number which indicates the number of observations of Category II phrases along each of the paths in the respective set. The probability of occurrence of each path is obtained by multiplying the respective probabilities along the path. Simple inspection shows that all the paths ending at the same terminal (i.e., in a set) have the same probability of occurrence. Thus, the number of paths in a set multiplied by the probability of occurrence of each path in the set is equal to the probability of reaching the associated terminal. This, clearly, gives the PDF of a number of phrases from Category II in the data block, i.e., the binomial distribution of (5).

Obviously, in practice, where the block length m is fixed and the phrase length varies, the trellis of Fig. 1, and thus, (5) may not give a good prediction of PCF. An exact evaluation of PCF requires consideration of the length of branches (arrows) in the trellis and the fact that the summation of the branch lengths along each path cannot be larger than the block length, m . As noted earlier, a full consideration of this point will result in a procedure which involves use of multinomial distribution which is difficult to handle. We proceed with our next analysis which

considers the length of the Category II phrase accurately, but still uses an average length for the phrases from Category I.

B. Analysis Based on Double Phrase Length

For convenience of the presentation here, let us assume that the average phrase length in Category I is \bar{T}_1 and the Category II phrase has a length of $3\bar{T}_1$. Fig. 2 presents a trellis which is built up for this scenario, in the case where $m = 13\bar{T}_1$. Here, we have three types of branches; two horizontal and one diagonal. We note that near the end of each sequence of horizontal branches, we reach a point where the remaining length is smaller than a Category II phrase length, thus, the Category I phrases occur with probability 1. Such branches are indicated by solid line arrows. The PDF associated with each of the terminals in Fig. 2 can be calculated in the same manner as done in Fig. 1. That is, the probability of reaching each terminal is obtained by identifying all the paths connecting the root to the terminal and adding the respective probabilities.

We note that unlike Fig. 1, in the trellis of Fig. 2, the paths ending at the same terminal do not share the same probability of occurrence. For example, the paths ending at terminal 2 in Fig. 2 may be divided into the subsets 0, 1, and 2, with the respective probabilities $p_0 = p^2(1-p)^5$, $p_1 = p^2(1-p)^6$, and $p_2 = p^2(1-p)^7$. Moreover, we may observe that the number of paths in each of these sets is a binomial combinatorial number (compare the form of the trellis in each subset in Fig. 2 with each set in Fig. 1). Generalization of this observation is obvious and will result in the PDF equation (for $x \geq 1$)

$$f_2(x) = \binom{M_{x+1}}{x} p^x (1-p)^{M_{x+1}-x} + \sum_{i=1}^{\delta_x} \binom{M_{x+1}+i-1}{x-1} p^x (1-p)^{M_{x+1}-x+i} \quad (7)$$

where $M_x = \lfloor (m - (k - \bar{T}_1)x) / \bar{T}_1 \rfloor$ is the number of phrases on the paths ending at terminal x , $\delta_x = M_x - M_{x+1}$ is the number of solid line arrows ending at terminal x , and $\lfloor x \rfloor$ denotes the integer part of x . A more accurate estimate of PCF is, thus, given by

$$p_f = \sum_{x>r} f_2(x). \quad (8)$$

To evaluate the accuracy of the results above, in Fig. 3, we have presented a set of PCF curves obtained by using (6) and (8) and also from computer simulations, for a block length $m = 400$ with different maximum run-length constraints k . The results clearly show that the double-phrase approximation gives almost perfect results. The single-phrase approach, however, does not give accurate results, particularly when k is small. We also note that the single-phrase approach provides an upper limit to PCF. This may be explained as follows.

In the single-phrase analysis, we ignore the important fact that upon occurrence of a phrase from Category II, the remaining length in the present block will reduce by the length of the Category II phrase. That is, we ignore the fact that the length of the horizontal paths in the trellis decreases as we move to the upper branches. On the contrary, in the double-phrase analysis, this reduction of trellis path lengths is considered. Ignoring the

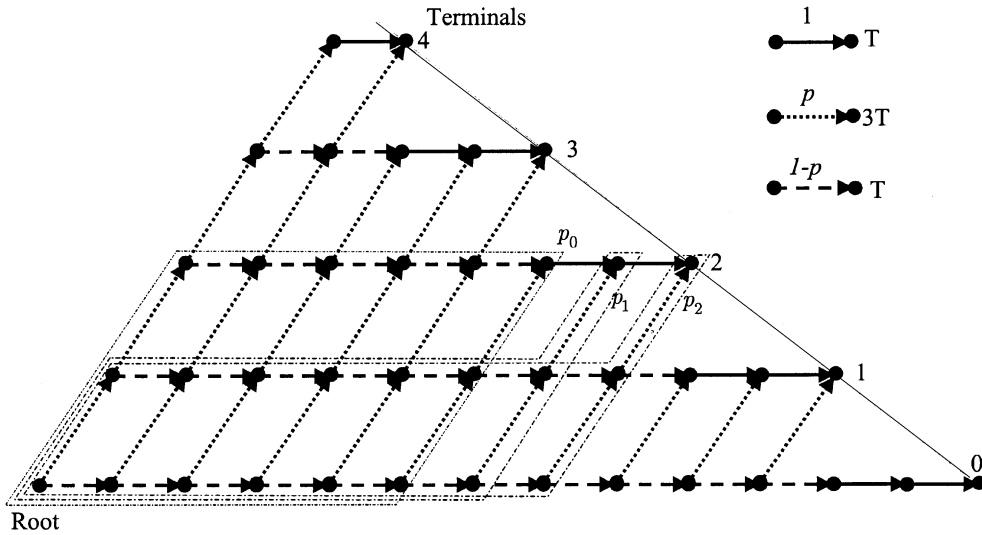


Fig. 2. Trellis diagram explanation based on double-phase approach.

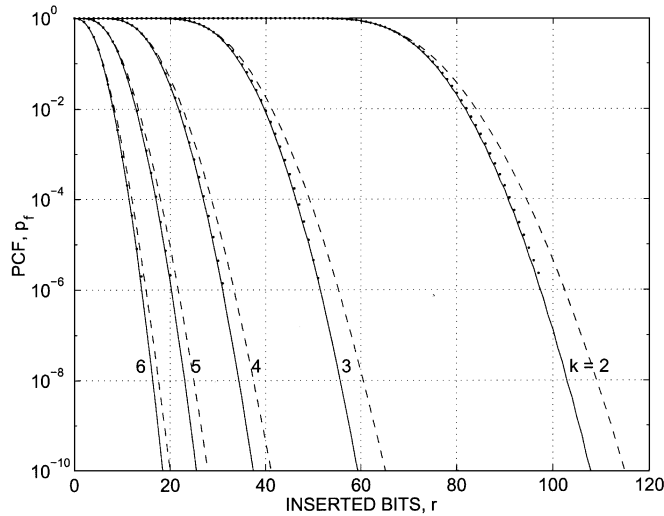


Fig. 3. PCF p_f , for weakly constrained codes. Block length $m = 400$. Solid-line curves are the theoretical results based on double-phase approximation. Dots are simulation results based on examination of 10^7 data blocks. Dashed-line curves are the theoretical results based on single-phase approximation.

reduction in the path lengths obviously results in overcounting the number of paths that end at higher terminals. This, in turn, results in overestimating the chance of reaching higher terminals that satisfy $x > r$, i.e., those that correspond to the code failure.

Further results of (8) are presented in Fig. 4. Here, the code efficiency is fixed at $\eta \approx 0.98$ and is used to select the value of r according to

$$r = \left\lfloor \frac{m}{\eta C(0, k)} - m \right\rfloor. \quad (9)$$

We note that PCF, p_f , decreases rapidly with the increase of both the run length, k , and the block length, m . However, when k is small, say $k < 8$, PCF is high, and thus, may not be acceptable.

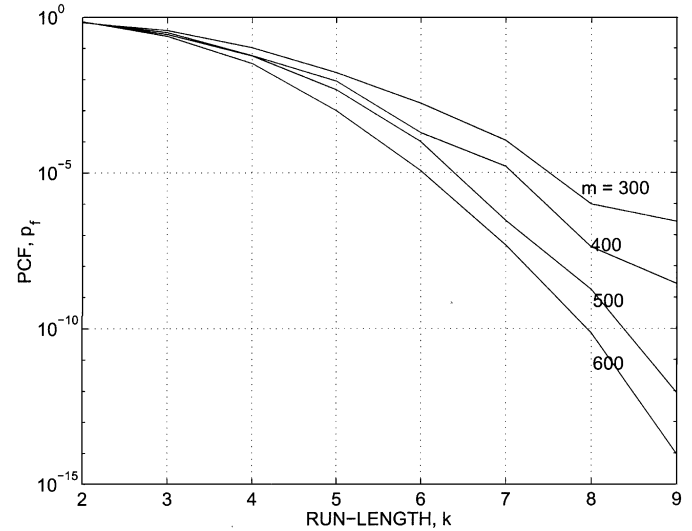


Fig. 4. PCF p_f , as a function of the run length k , with the input data block length m as a parameter. Code efficiency is fixed at $\eta \approx 0.98$.

IV. IMPROVED WEAKLY CONSTRAINED CODES USING GUIDED SCRAMBLING (GS) TECHNIQUE

For smaller values of k , we propose using the idea of GS [1], [9] to achieve a low PCF. Fig. 5 depicts a schematic of the encoder which uses the idea of GS. The GS encoder takes a block of input data (m bits) and generates $N = 2^{r_1}$ randomly scrambled sequences of length $m + r_1$ bits each. The scrambled sequences are bit stuffed with a maximum of r_2 bits to satisfy the constraint. Assuming that the scrambled sequences are sufficiently distinct, which is achievable according to [1], the probability of failure of not being able to get at least one successfully constrained sequence from the structure of Fig. 5 is

$$p_{f,gs} = (p_f)^N. \quad (10)$$

We note that the code rate here is $R = m/(m + r_1 + r_2)$. Fig. 6 shows the PCF, $p_{f,gs}$, as a function of code efficiency, η , with different input data block length, m , for fixed values of

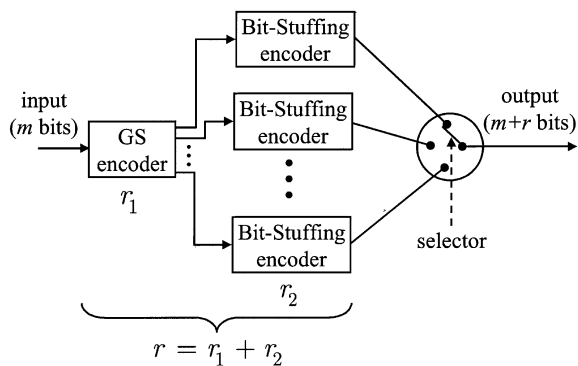


Fig. 5. Combined GS and bit-stuffing encoder scheme for weakly constrained codes.

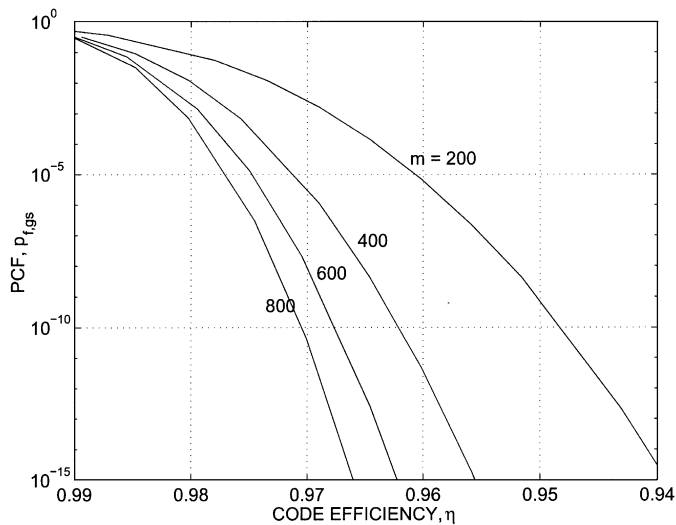


Fig. 6. PCF, $p_{f,gs}$, as a function of the code efficiency, η . $k = 3, r_1 = 3$.

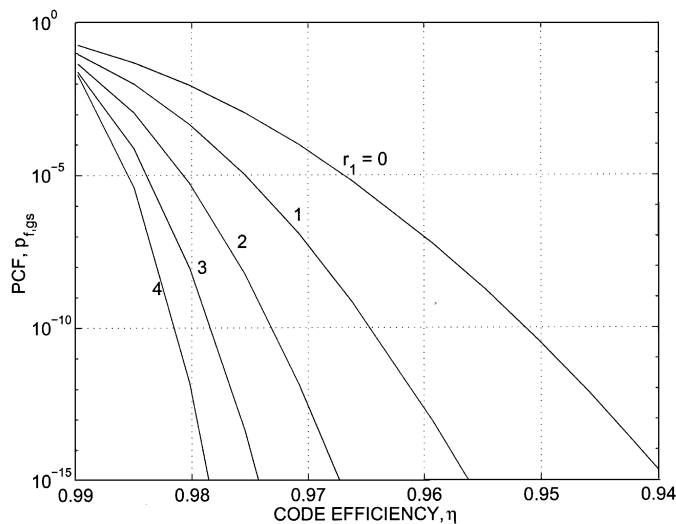
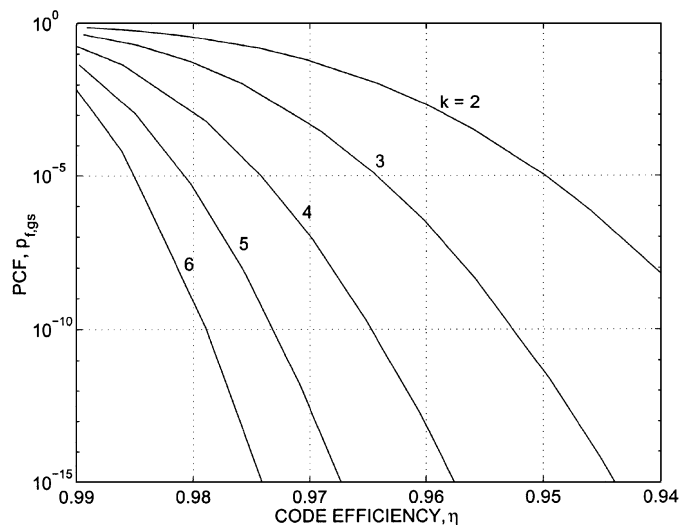
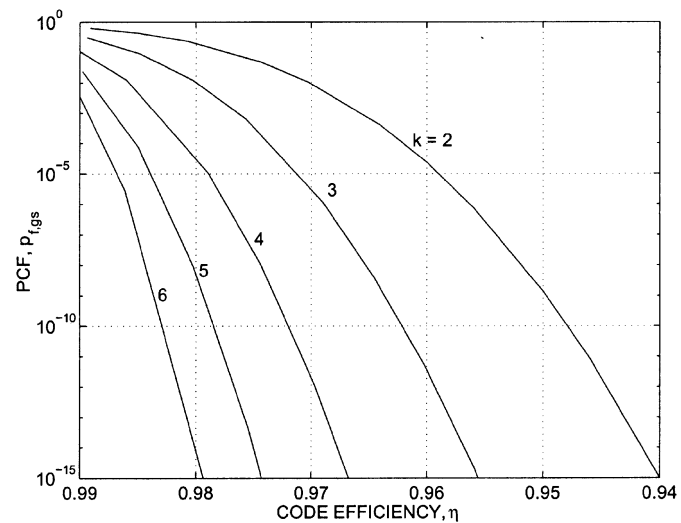


Fig. 7. PCF, $p_{f,gs}$, as a function of the code efficiency, η . $m = 400, k = 5$.

$r_1 = 3$ and $k = 3$. These results show that the failure probability decreases with the increase of data block length, m . To achieve small $p_{f,gs}$, it is preferred to choose large values of m . On the other hand, to limit the error propagation, it is better to choose



(a)



(b)

Fig. 8. PCF, $p_{f,gs}$, as a function of the code efficiency, η . (a) $m = 400, r_1 = 2$. (b) $m = 400, r_1 = 3$.

small m . For the following presentation, we choose $m = 400$ as a compromise choice.

Fig. 7 presents a set of plots showing the PCF, $p_{f,gs}$, versus code efficiency, η , for $k = 5$ and $m = 400$ and various values of r_1 . Note that $r_1 = 0$ corresponds to no scrambling. From the results, we clearly see the effect of GS in improving the failure probability. For example, without scrambling, a PCF of 10^{-10} can be achieved at a code efficiency of 0.952, while if $r_1 = 3$, i.e., eight scrambled codes are examined, for the same PCF, the achievable code efficiency is improved to 0.978.

Fig. 8 presents two sets of curves which show how the PCF improves as k increases, for values of $r_1 = 2, 3$. The results are interesting and also impressive. They show very efficient codes ($\eta > 0.95$) can be designed for even small values of k . For example, if we choose $m = 400, r_1 = 2$ and failure probability of 10^{-10} , the code efficiency of the resulting RLL (0, 2), (0, 3), and (0, 6) codes in Fig. 8(a) are about 0.933, 0.953, and 0.978, respectively. In Fig. 8(b), the corresponding code efficiency will improve to the values 0.948, 0.962, and 0.984,

respectively, if $r_1 = 3$ is selected. These should be compared with the values 0.9101, 0.9388, and 0.9467 corresponding to the industry standards 4/5 RLL (0, 2), 8/9 RLL (0, 3), and 16/17 RLL (0, 6) codes, respectively [4].

V. CONCLUSION

In this letter, we introduced a general method of constructing RLL(d, k) constrained codes from arbitrary sequences using a simple bit-stuffing technique. We noted that this method, although it leads to a class of relatively efficient codes, has the problem of error propagation and variable code rate. This problem was then resolved by dividing the input data into blocks of length m and applying the proposed method to individual blocks, but keeping the number of stuffed bits fixed for each block. We noticed that this leads to a class of weakly constrained codes with a fixed code rate. These codes were analyzed for the case of $(0, k)$ constraint; the only case that we had found giving good results. We found that these codes can only give acceptable results for larger values of k (≥ 8). The use of GS was then suggested as a very effective method of achieving high efficiency, for smaller values of k .

REFERENCES

- [1] I. J. Fair, W. D. Gover, W. A. Krzymien, and R. I. MacDonald, "Guided scrambling: a new line coding technique for high bit rate fiber optic transmission systems," *IEEE Trans. Commun.*, vol. 39, pp. 289–297, Feb. 1991.
- [2] W. Feller, *An Introduction to Probability Theory and Its Application*, 2nd ed. New York: Wiley, 1971.
- [3] K. A. S. Immink, "Weakly constrained codes," *Electron. Lett.*, vol. 33, no. 23, pp. 1943–1944, Nov. 1997.
- [4] —, *Codes For Mass Data Storage Systems*. Eindhoven, The Netherlands: Shannon Foundation, 1999.
- [5] E. A. Lee and D. G. Messerschmitt, *Digital Communication*. Boston, MA: Kluwer, 1988.
- [6] P. E. Bender and J. K. Wolf, "A universal algorithm for generating optimal and nearly optimal run-length limited, charge-constrained binary sequences," in *Proc. 1993 IEEE Int. Symp. Information Theory*, Jan. 17–22, 1993, p. 6.
- [7] J. W. M. Bergmans, *Digital Baseband Transmission and Recording*. Boston, MA: Kluwer, 1996.
- [8] S. X. Wang and A. M. Taratorin, *Magnetic Information Storage Technology*. San Diego, CA: Academic, 1999.
- [9] A. Kunisa, "Run-length control based on guided scrambling for digital magnetic recording," *IEICE Trans. Electron.*, vol. E82-C, no. 12, pp. 2209–2217, Dec. 1999.