- [6] —, "Communication in the presence of noise," *Proc. IRE*, vol. 37, pp. 10–21, 1949; reprinted in *Proc. IEEE*, vol. 72, pp. 1192–1201, 1984.
- [7] B. F. Logan, Jr., "Signals designed for recovery after clipping—I, II, III," *ATT Bell Lab. Tech. J.*, vol. 63, pp. 261–286, 287–306, 379–399, 1984.
- [8] E. C. Titchmarsh, "The zeros of certain integral functions," *Proc. London Math. Soc.*, vol. 25, pp. 283–302, 1926.
- [9] S. Shamai, private communication.

Cascading Runlength-Limited Sequences

Jos H. Weber, *Member, IEEE*, and Khaled A. S. Abdel-Ghaffar

Abstract—In magnetic or optical storage devices, it is often required to map the data into runlength-limited sequences. To ensure that cascading such sequences does not violate the runlength constraints, a number of merging bits are inserted between two successive sequences. A theory is developed, in which the minimum number of merging bits is determined and the efficiency of a runlength-limited fixed-length coding scheme is considered.

Index Terms—Fixed-length codes, modulation systems, runlength-limited sequences.

I. INTRODUCTION

Many modulation systems used in magnetic and optical recording are based on binary runlength-limited codes. A string of bits is said to be runlength-limited if the number of zeros between two consecutive ones is bounded between a certain minimum value d and a certain maximum value k . Such sequences are also called (d, k) constrained. The lower runlength constraint is imposed to reduce intersymbol interference, while the upper runlength constraint is imposed to maintain synchronization. For a general introduction to the theory of runlength-limited sequences and its applications, we refer to the excellent overview papers by Siegel [7] and Immink [3].

In order to prevent the cascading of runlength-limited sequences of fixed length from violating the runlength constraints, some merging techniques have been given by Tang and Bahl [8] and Beenker and Immink [1]. In this paper, we will develop a general theory on cascading runlength-limited sequences, in which the above-mentioned methods appear as special cases.

The following notation will be used. Let d, k, l, r , and n be integers satisfying $0 \leq d < k$, $0 \leq l \leq r \leq k$, and $n \geq 1$, unless explicitly stated otherwise. Let $\mathcal{M}_{dklr}(n)$ denote the set of binary sequences of length n , starting with at most l zeros, ending with at most r zeros, and having at least d and at most k zeros

between two consecutive ones. Such sequences are called $dklr$ -sequences. The cardinality of $\mathcal{M}_{dklr}(n)$ is denoted by $M_{dklr}(n)$. In case there is no upper runlength constraint, we say $k = \infty$. Note that the assumption $l \leq r$ is not a real restriction, since $\mathcal{M}_{dklr}(n)$ with $l > r$ could be obtained by simply reversing each sequence in $\mathcal{M}_{dkrl}(n)$. Let γ^i denote the sequence γ repeated i times. For example, $0^3 10(100)^2 1 = 000101001001$. Let $\text{Rev}[\gamma]$ denote the reversed sequence of γ . The number of "1's" in a sequence will be called the weight of the sequence. Let $\lfloor x \rfloor$ and $\lceil x \rceil$ denote the floor and the ceiling of the real number x , respectively.

A way to guarantee that cascading $dklr$ -sequences gives sequences still satisfying the (d, k) constraint is to insert b appropriate merging bits between any two consecutive sequences. If we use a subset of $\mathcal{M}_{dklr}(n)$ as a runlength-limited code where each codeword corresponds to a message from a message set $\mathcal{A}$ of size A , then the actual number of bits needed to represent a message is thus $N = n + b$. The code rate R can then be defined by

$$R = \frac{\log_2 A}{N}. \quad (1)$$

As an example, we mention the Eight-to-Fourteen Modulation (EFM) code for the compact disk [4], where eight data bits are mapped to a runlength-limited sequence of length 14 with $(d, k) = (2, 10)$. In order to maintain the runlength conditions when cascading these sequences, merging bits are inserted between adjacent sequences. The runlength-limited sequences are judiciously chosen such that three merging bits suffice to maintain the runlength constraints. If the runlength is in danger of becoming too short, all of them are set to "0." If the runlength is in danger of becoming too long, one of the merging bits is set to "1." The code rate of this system is $(\log_2 2^8)/(14 + 3) = 8/17$.

Our first goal will be to minimize the number of merging bits needed to cascade any two sequences from $\mathcal{M}_{dklr}(n)$. Our final goal will then be to maximize the code rate R for given values of d, k , and A by a proper choice of l and r . To this end, note the following with regard to this choice. On one hand, it seems a good idea to take large values for l and r , since $M_{dklr}(n)$ is nondecreasing in l and r . On the other hand, small values of l and r may require fewer merging bits than large values. This dilemma plays a central part in optimizing the efficiency of the coding scheme.

The rest of this paper will be organized as follows. First, in Section II, we derive recursion relations for determining $M_{dklr}(n)$. Further, in Section III, we give lower bounds on the number of merging bits needed for cascading any two sequences from $\mathcal{M}_{dklr}(n)$. Next, in Section IV, we give optimal merging rules for such a cascading scheme. Then, in Section V, we try to find out, for any choice of d and k , which values of l and r give the best overall efficiency for a fixed-length coding scheme. Finally, conclusions are drawn in Section VI.

II. NUMBER OF $dklr$ -SEQUENCES

The number of $dklr$ -sequences of length n has been derived by Tang and Bahl [8] for the special case $l = r = k$. In this section, we will give recursion relations to determine $M_{dklr}(n)$ for arbitrary choices of l and r . Let $\mathcal{M}_{dklr}^w(n)$ denote the set of all $dklr$ -sequences of length n and weight w . We first compute the sizes $M_{dklr}^w(n)$ of such sets.

Since the all-zero sequence of length n is a $dklr$ -sequence if

Manuscript received May 23, 1992; revised March 3, 1993. This work was supported by the National Science Foundation under Grants NCR 89-08105 and NCR 91-15423. The work of K. A. S. Abdel-Ghaffar was supported by an IBM Faculty Development Award. This paper was presented in part at the 12th Symposium on Information Theory in the Benelux, Veldhoven, The Netherlands, May 1991, and at the 1991 IEEE International Symposium on Information Theory, Budapest, Hungary, June 1991.

J. H. Weber is with the Information Theory Group, Department of Electrical Engineering, Delft University of Technology, 2600 GA Delft, The Netherlands.

K. A. S. Abdel-Ghaffar is with the Department of Electrical Engineering and Computer Science, University of California, Davis, CA 95616. IEEE Log Number 9213023.

TABLE I
EXAMPLES OF $M_{dklr}(n)$

n	$M_{dklr}(n)$									
	$d=0$	$d=0$	$d=0$	$d=1$	$d=1$	$d=1$	$d=2$	$d=2$	$d=2$	$d=3$
	$k=2$	$k=2$	$k=2$	$k=3$	$k=3$	$k=3$	$k=4$	$k=4$	$k=4$	$k=4$
	$l=0$	$l=1$	$l=2$	$l=2$	$l=2$	$l=3$	$l=2$	$l=3$	$l=4$	$l=0$
	$r=2$	$r=1$	$r=2$	$r=2$	$r=3$	$r=3$	$r=2$	$r=3$	$r=4$	$r=2$
1	1	2	2	2	2	2	2	2	2	1
2	2	3	4	3	3	3	3	3	3	1
3	4	5	7	4	4	5	3	4	4	1
4	7	9	13	5	6	7	3	5	6	0
5	13	17	24	8	9	10	4	6	8	1
6	24	31	44	11	13	15	6	8	10	2
7	44	57	81	17	19	22	8	11	13	2
8	81	105	149	24	28	32	10	14	18	1
9	149	193	274	36	41	47	13	19	24	1
10	274	355	504	52	60	69	18	25	31	3

and only if $n \leq l$, we have

$$M_{dklr}^0(n) = \begin{cases} 1 & \text{if } n \leq l \\ 0 & \text{if } n \geq l+1. \end{cases} \quad (2)$$

A sequence γ of length n and weight $w \geq 1$ can be uniquely written as $\gamma = 0^j\delta$, where δ is a sequence of length $n-j$ starting with a "1." If $w = 1$, such a sequence is a $dklr$ -sequence if and only if $0 \leq j \leq l$ and $0 \leq n-j-1 \leq r$. Hence,

$$M_{dklr}^1(n) = \max\{0, \min\{l, n-1\} - \max\{0, n-1-r\} + 1\}. \quad (3)$$

Next, we define a mapping ϕ_i from $\mathcal{M}_{dklr}^{w-1}(n-i)$ into $\mathcal{M}_{dklr}^w(n)$, where $w > 1$ and $n > i$, by

$$\phi_i(\gamma) = \phi_i(0^j\delta) = 0^j10^{i-1}\delta$$

for $i = d+1, d+2, \dots, k+1$. Each sequence in $\mathcal{M}_{dklr}^w(n)$ is an image under exactly one of the ϕ_i 's, and its preimage can be uniquely recovered by deleting the first "1" and its succeeding "0's." Hence,

$$M_{dklr}^w(n) = \sum_{i=d+1}^{\min\{n-1, k+1\}} M_{dklr}^{w-1}(n-i) \quad (4)$$

for $w \geq 2$.

Now, combining

$$M_{dklr}(n) = \sum_{w=0}^n M_{dklr}^w(n) \quad (5)$$

with (2), (3), and (4) gives the desired recursion relations to determine the number of $dklr$ -sequences of length n . This leads to a very simple expression for $n \geq k+l+2$:

$$\begin{aligned} M_{dklr}(n) &= \sum_{w=0}^n M_{dklr}^w(n) = \sum_{w=2}^n M_{dklr}^w(n) \\ &= \sum_{w=2}^n \sum_{i=d+1}^{k+1} M_{dklr}^{w-1}(n-i) \\ &= \sum_{i=d+1}^{k+1} \sum_{w=1}^{n-1} M_{dklr}^w(n-i) = \sum_{i=d+1}^{k+1} M_{dklr}(n-i). \end{aligned} \quad (6)$$

So, for $n \geq k+l+3$, we have

$$\begin{aligned} M_{dklr}(n) &= M_{dklr}(n-1) + M_{dklr}(n-d-1) \\ &\quad - M_{dklr}(n-k-2). \end{aligned} \quad (7)$$

In Table I, we give $M_{dklr}(n)$ for some choices of n, d, k, l , and r . Although $M_{dklr}(n)$ is increasing in n for sufficiently large n , this is not necessarily true if n is small, as can be seen from the $d=3$ column in Table I. Of course, $M_{dklr}(n)$ is nondecreasing in l, r , and k , but nonincreasing in d . An interesting question, that is relevant to Section V, is which choice of l and r maximizes $M_{dklr}(n)$ when the sum $l+r$ is fixed. To this end, note that for $0 \leq l' < l \leq r < r' \leq k$ and $l+r = l'+r'$,

$$M_{dklr}^0(n) - M_{dkl'r'}^0(n) = \begin{cases} 1 & \text{if } l'+1 \leq n \leq l \\ 0 & \text{otherwise} \end{cases} \quad (8)$$

which follows from (2), and

$$\begin{aligned} M_{dklr}^1(n) - M_{dkl'r'}^1(n) &= \begin{cases} n-l'-1 & \text{if } l'+1 \leq n \leq l \\ l-l' & \text{if } l+1 \leq n \leq r \\ l-l'+1+r-n & \text{if } r+1 \leq n \leq r' \\ 0 & \text{otherwise} \end{cases} \end{aligned} \quad (9)$$

which follows from (3). Combining this with (4) and (5) gives

$$M_{dklr}(n) \geq M_{dkl'r'}(n) \quad (10)$$

for $0 \leq l' < l \leq r < r' \leq k$, and $l+r = l'+r'$. Hence, the size of $\mathcal{M}_{dklr}(n)$ is maximized over all l and r satisfying $l+r = S$ by choosing $l = \lfloor S/2 \rfloor$ and $r = \lceil S/2 \rceil$:

$$M_{dk\lfloor S/2 \rfloor \lceil S/2 \rceil}(n) \geq M_{dklr}(n) \quad (11)$$

for all l and r with $l+r = S$. An illustration can be found in Table I by comparing the first two columns in which $d=0$ and $k=2$.

III. LOWER BOUNDS ON THE NUMBER OF MERGING BITS

Let B_{dklr} be defined as the minimum value of b such that for each choice of integers s and t satisfying $0 \leq s \leq r$ and $0 \leq t \leq l$, there exists a sequence β of length b such that the sequence

$$\alpha(s, \beta, t) = 10^s\beta 0^t \quad (12)$$

is (d, k) constrained. The sequence β can be inserted between any $dklr$ -sequence ending with s "0's" and any $dklr$ -sequence starting with t "0's" in order to maintain the (d, k) constraint. So, knowledge of B_{dklr} is very important with regard to minimizing the number of merging bits in a cascading scheme. In this section, we will derive some lower bounds on B_{dklr} .

First, define $\mathcal{Z}_{dk}$ as the set of all positive integers v for which there exists a (d, k) constrained sequence of length v that both

starts and ends with a "1." It is easy to check that

$$\begin{aligned}\mathcal{Z}_{dk} &= \{z(d+1) + i + 1 : z \geq 0 \wedge 0 \leq i \leq z(k-d)\} \\ &= \left\{z(d+1) + i + 1 : 0 \leq z \leq \left\lfloor \frac{d}{k-d} \right\rfloor - 1 \right. \\ &\quad \left. \wedge 0 \leq i \leq z(k-d) \right\} \\ &\cup \left\{j : j \geq \left\lfloor \frac{d}{k-d} \right\rfloor (d+1) + 1\right\}.\end{aligned}\quad (13)$$

We will make use of the following two remarks.

Remark 1: If $y(d+1) + h + 1 \in \mathcal{Z}_{dk}$, where y and h are integers and $0 \leq h \leq d$, then $h \leq y(k-d)$.

Remark 2: If

$$\left(\left\lfloor \frac{d}{k-d} \right\rfloor - 1\right)(k+1) + 2 \leq v \leq \left\lfloor \frac{d}{k-d} \right\rfloor (d+1),$$

then $v \notin \mathcal{Z}_{dk}$.

The first remark follows from (13), and the second remark follows from the first one. The results are now stated in Lemmas 1-4.

Lemma 1: For $0 \leq d < k$ and $0 \leq l \leq r \leq k$,

$$B_{dklr} \geq d.$$

Proof: From the definition of B_{dklr} , it follows that there exists a sequence β of length B_{dklr} such that $\alpha(0, \beta, 0) = 1\beta 1$ is (d, k) constrained. Hence, $B_{dklr} \geq d$. $\square$

Lemma 2: For $0 \leq d < k < \infty$ and $0 \leq l \leq r \leq k$,

$$B_{dklr} \geq \left\lfloor \frac{m}{k-d} \right\rfloor (d+1) - 1$$

where $m = \min\{l+r, d\}$.

Proof: Suppose that $B = B_{dklr} \leq \lfloor m/(k-d) \rfloor (d+1) - 2$. Then, B can be uniquely written as

$$B = y(d+1) + h - 1$$

where y and h are integers such that $0 \leq y \leq \lfloor m/(k-d) \rfloor - 1$ and $0 \leq h \leq d$. From the definition of B_{dklr} , it follows that there exists a sequence β of length B such that $\alpha(0, \beta, 0)$ is (d, k) constrained, which forces that $B+2 \in \mathcal{Z}_{dk}$. Remark 1 implies $h \leq y(k-d)$. It follows from the definition of B_{dklr} that there exists a sequence β of length B such that $\alpha(s, \beta, t)$, with $0 \leq s \leq r$, $0 \leq t \leq l$, and $s+t = y(k-d) + 1 - h$, is (d, k) constrained. Note that such s and t indeed occur since

$$\begin{aligned}s+t &= y(k-d) + 1 - h \leq \left(\left\lfloor \frac{m}{k-d} \right\rfloor - 1\right)(k-d) + 1 \\ &\leq \left(\frac{m+k-d-1}{k-d} - 1\right)(k-d) + 1 = m \leq l+r.\end{aligned}$$

Hence, $1 + s + B + t + 1 = y(d+1) + y(k-d) + 2 \in \mathcal{Z}_{dk}$. Since

$$0 \leq y(k-d) + 1 \leq \left(\left\lfloor \frac{d}{k-d} \right\rfloor - 1\right)(k-d) + 1 \leq d,$$

then Remark 1 implies that $y(k-d) + 1 \leq y(k-d)$. This contradiction shows that our assumption was wrong, and therefore proves the result. $\square$

Lemma 3: For $0 \leq d < k < \infty$, $0 \leq l \leq r \leq k$, and $r \geq \lfloor d/(k-d) \rfloor (k-d) + 1$,

$$B_{dklr} \geq \left\lfloor \frac{d}{k-d} \right\rfloor (d+1).$$

Proof: The lemma is trivially true if $d = 0$. So, let $d \geq 1$.

Since $l+r \geq r \geq \lfloor d/(k-d) \rfloor (k-d) + 1 \geq d+1$, it follows from Lemma 2 that $B = B_{dklr} \geq \lfloor d/(k-d) \rfloor (d+1) - 1$. Now, let us suppose that $B = \lfloor d/(k-d) \rfloor (d+1) - 1$. It follows from the definition of B_{dklr} that there exists a sequence β of length B such that $\alpha(s, \beta, t)$, with $s = \lfloor d/(k-d) \rfloor (k-d) + 1$ and $t = 0$, is (d, k) constrained. If β would contain only "0's," then

$$\begin{aligned}k &\geq s + B + t = \left\lfloor \frac{d}{k-d} \right\rfloor (k-d) + 1 + \left\lfloor \frac{d}{k-d} \right\rfloor (d+1) - 1 \\ &= \left\lfloor \frac{d}{k-d} \right\rfloor (k+1) \geq k+1.\end{aligned}$$

From this contradiction, we conclude that β contains at least one "1." Let i denote the number of leading "0's" in β . Then, i satisfies $0 \leq i \leq k - \lfloor d/(k-d) \rfloor (k-d) - 1$ and $B - i + 1 \in \mathcal{Z}_{dk}$. However, on one hand,

$$B - i + 1 = \left\lfloor \frac{d}{k-d} \right\rfloor (d+1) - 1 - i + 1 \leq \left\lfloor \frac{d}{k-d} \right\rfloor (d+1),$$

and on the other hand

$$\begin{aligned}B - i + 1 &= \left\lfloor \frac{d}{k-d} \right\rfloor (d+1) - 1 - i + 1 \\ &\geq \left\lfloor \frac{d}{k-d} \right\rfloor (d+1) - k + \left\lfloor \frac{d}{k-d} \right\rfloor (k-d) + 1 \\ &= \left(\left\lfloor \frac{d}{k-d} \right\rfloor - 1\right)(k+1) + 2,\end{aligned}$$

which shows with Remark 2 that $B - i + 1 \notin \mathcal{Z}_{dk}$. Hence, our assumption was wrong, which proves the result. $\square$

Lemma 4: For $0 \leq d < k < \infty$, $0 \leq l \leq r \leq k$, and $l+r \geq \lfloor d/(k-d) \rfloor (k-d) + k + 1$,

$$B_{dklr} \geq \left\lfloor \frac{d}{k-d} \right\rfloor (d+1) + 1.$$

Proof: Since $r \geq (\lfloor d/(k-d) \rfloor (k-d) + k + 1)/2 \geq (2\lfloor d/(k-d) \rfloor (k-d) + 2)/2 = \lfloor d/(k-d) \rfloor (k-d) + 1$, it follows from Lemma 3 that $B = B_{dklr} \geq \lfloor d/(k-d) \rfloor (d+1)$. Now, let us suppose that $B = \lfloor d/(k-d) \rfloor (d+1)$. It follows from the definition of B_{dklr} that there exists a sequence β of length B such that $\alpha(s, \beta, t)$, with $0 \leq s \leq r$, $0 \leq t \leq l$, and $s+t = \lfloor d/(k-d) \rfloor (k-d) + k + 1$, is (d, k) constrained. If β would contain only "0's," then

$$\begin{aligned}k &\geq s + B + t = \left\lfloor \frac{d}{k-d} \right\rfloor (k-d) + k + 1 + \left\lfloor \frac{d}{k-d} \right\rfloor (d+1) \\ &\geq k+1.\end{aligned}$$

From this contradiction, we conclude that β contains at least one "1." Let i denote the number of leading "0's" in β , and let j denote the number of trailing "0's" in β . Then, i and j satisfy $0 \leq i \leq k - s$, $0 \leq j \leq k - t$, and $B - i - j \in \mathcal{Z}_{dk}$. However, on one hand,

$$B - i - j = \left\lfloor \frac{d}{k-d} \right\rfloor (d+1) - i - j \leq \left\lfloor \frac{d}{k-d} \right\rfloor (d+1),$$

and on the other hand,

$$\begin{aligned}B - i - j &= \left\lfloor \frac{d}{k-d} \right\rfloor (d+1) - i - j \\ &\geq \left\lfloor \frac{d}{k-d} \right\rfloor (d+1) - 2k + s + t \\ &= \left\lfloor \frac{d}{k-d} \right\rfloor (k+1) - k + 1 \\ &= \left(\left\lfloor \frac{d}{k-d} \right\rfloor - 1\right)(k+1) + 2,\end{aligned}$$

which shows with Remark 2 that $B - i - j \notin \mathcal{Z}_{dk}$. Hence, our assumption was wrong, which proves the result. $\square$

IV. OPTIMAL MERGING RULES

In this section, we will give for all d, k, l, r, s , and t satisfying $0 \leq s \leq r$ and $0 \leq t \leq l$ a rule to choose a sequence β , whose length does not depend on s and t , such that $\alpha(s, \beta, t) = 10^s 0^t 1$ is (d, k) constrained. It will turn out that, in each case, the number of merging bits used attains the best lower bound on B_{dklr} derived in the previous section. Hence, these merging rules can be called optimal in the sense of minimizing the (fixed) number of merging bits for any choice of d, k, l , and r .

We start by treating the cases in which there is no upper runlength constraint ($k = \infty$):

Merging rule for $k = \infty$ (14)

$$\beta = 0^d.$$

It can be immediately seen that $\alpha(s, 0^d, t) = 10^{s+d+t} 1$ is indeed (d, ∞) constrained for all s and t . Hence, we have established the following theorem.

Theorem 1: For $d \geq 0$ and $0 \leq l \leq r$,

$$B_{d\infty lr} = d.$$

Proof: Combine merging rule (14) with Lemma 1. $\square$

Next, we consider the cases in which there is no lower runlength constraint ($d = 0$):

Merging rule for $0 = d < k < \infty$ (15)

$$\begin{cases} \text{if } l + r \leq k, \text{ then choose } \beta = \epsilon; \\ \text{if } l + r > k, \text{ then choose } \beta = 1. \end{cases}$$

Here, ϵ is the empty sequence (of length zero). It can be immediately seen that $\alpha(s, \epsilon, t) = 10^{s+t} 1$ is indeed $(0, k)$ constrained for all s and t if $l + r \leq k$, and that $\alpha(s, 1, t) = 10^s 10^t 1$ is $(0, k)$ constrained for all s and t if $l + r > k$. Hence, we have established the following theorem.

Theorem 2: For $0 \leq l \leq r \leq k < \infty$,

$$B_{0klr} = \begin{cases} 0 & \text{if } l + r \leq k \\ 1 & \text{if } l + r > k. \end{cases}$$

Proof: Combine merging rule (15) with Lemmas 1 and 4. $\square$

Finally, we consider the cases in which there are both a lower and an upper runlength constraint ($0 < d < k < \infty$). Before giving the merging rule, we define some sequences.

• The sequence

$$\sigma(b, d, i, j) = \begin{cases} 0^b & \text{if } q = 1 \\ 0^{d+v-i}(10^{d+v})^{q-2} 10^{d+v-j} & \text{if } q \geq 2 \wedge u = 0 \\ 0^{d+v+1-i}(10^{d+v+1})^{u-1} (10^{d+v})^{q-u-1} 10^{d+v-j} & \text{if } q \geq 2 \wedge u \geq 1 \end{cases}$$

for $0 \leq i \leq d \leq b$ and $0 \leq j \leq d$, where $q = \lfloor (b + i + j + 1)/(d + 1) \rfloor$, $p = b + i + j + 1 - (d + 1)q$, $v = \lfloor p/q \rfloor$, and $u = p - qv$.

• The sequence

$$\chi(b, d, i, j) = \begin{cases} 0^b & \text{if } q = 0 \\ 0^i (10^{d+v+1})^u (10^{d+v})^{q-1-u} 10^{d+v-j} & \text{if } q \geq 1 \end{cases}$$

for $0 \leq i \leq b$ and $0 \leq j \leq d$, where $q = \lfloor (b - i + j)/(d + 1) \rfloor$, $p = b - i + j - (d + 1)q$, $v = \lfloor p/q \rfloor$, and $u = p - qv$.

• The sequence

$$\psi(b, d, i, j) = \begin{cases} 0^b & \text{if } q < 0 \\ 0^i 10^{b-1-i} & \text{if } q = 0 \\ 0^i (10^{d+v+1})^u (10^{d+v})^{q-u} 10^j & \text{if } q \geq 1 \end{cases}$$

for $0 \leq i \leq b$ and $0 \leq j \leq b$, where $q = \lfloor (b - i - j - 1)/(d + 1) \rfloor$, $p = b - i - j - 1 - (d + 1)q$, $v = \lfloor p/q \rfloor$, and $u = p - qv$.

Note that the length of any of these sequences equals the parameter b . With the aid of Tables II, III, and IV, we are now ready to give the merging rule:

Merging rule for $0 < d < k < \infty$ (16)

$$\begin{aligned} & \text{if } l = r = 0, \text{ then choose } \beta = 0^d; \\ & \text{if } 1 \leq l + r \leq d, \\ & \quad \text{then choose } \beta = \sigma\left(\left\lceil \frac{l+r}{k-d} \right\rceil (d+1) - 1, d, s, t\right); \\ & \text{if } l + r \geq d + 1 \wedge r \leq \left\lceil \frac{d}{k-d} \right\rceil (k-d), \\ & \quad \text{then choose } \beta \text{ from Table II;} \\ & \text{if } l + r \leq \left\lceil \frac{d}{k-d} \right\rceil (k-d) + k \wedge r \geq \left\lceil \frac{d}{k-d} \right\rceil (k-d) + 1, \\ & \quad \text{then choose } \beta \text{ from Table III;} \\ & \text{if } l + r \geq \left\lceil \frac{d}{k-d} \right\rceil (k-d) + k + 1, \\ & \quad \text{then choose } \beta \text{ from Table IV.} \end{aligned}$$

It can be checked in a straightforward, but tedious, way that in each case, the sequence $\alpha(s, \beta, t)$ is indeed (d, k) constrained. As an example, we treat the case $s \geq d + 1 \wedge t \leq d$ if $l + r \geq \lceil d/(k-d) \rceil (k-d) + k + 1$. It can be seen from Table IV that β is set equal to $\chi(\lceil d/(k-d) \rceil (d+1) + 1, d, 0, t)$. So, following the first "1" in $\alpha(s, \beta, t)$, we have a run of $s + 0 = s$ "0's," which satisfies the constraint

$$d \leq d + 1 \leq s \leq r \leq k.$$

It follows from the definition of χ that if $u = 0$, then the other runs of "0's" between two consecutive "1's" in $\alpha(s, \beta, t)$ are of length $d + v$, which satisfies the constraint

$$\begin{aligned} d & \leq d + v = d + \frac{p}{q} \leq d + \left\lceil \frac{d}{q} \right\rceil \\ & = d + \left\lceil \frac{d}{\left\lceil \frac{d}{k-d} \right\rceil (d+1) + 1 + t} \right\rceil \\ & \leq d + \left\lceil \frac{d}{\left\lceil \frac{d}{k-d} \right\rceil} \right\rceil \leq d + (k-d) = k. \end{aligned}$$

If $u \geq 1$, then all of these runs are either of length $d + v$ or of

TABLE II
MERGING RULE IN CASE $0 < D < K < \infty \wedge L + R \geq D + 1 \wedge R \leq \lceil D / (K - D) \rceil (K - D)$

s, t	β
$s \leq d \wedge t \leq d$	$\sigma\left(\left\lceil \frac{d}{k-d} \right\rceil (d+1) - 1, d, s, t\right)$
$s \geq d+1 \wedge t = 0$	$\chi\left(\left\lceil \frac{d}{k-d} \right\rceil (d+1) - 1, d, k - \left\lceil \frac{d}{k-d} \right\rceil (k-d), 0\right)$
$s \geq d+1 \wedge 1 \leq t \leq d$	$\chi\left(\left\lceil \frac{d}{k-d} \right\rceil (d+1) - 1, d, 0, t\right)$
$s = 0 \wedge t \geq d+1$	$\text{Rev}\left[\chi\left(\left\lceil \frac{d}{k-d} \right\rceil (d+1) - 1, d, k - \left\lceil \frac{d}{k-d} \right\rceil (k-d), 0\right)\right]$
$1 \leq s \leq d \wedge t \geq d+1$	$\text{Rev}\left[\chi\left(\left\lceil \frac{d}{k-d} \right\rceil (d+1) - 1, d, 0, s\right)\right]$
$s \geq d+1 \wedge t \geq d+1$	$\psi\left(\left\lceil \frac{d}{k-d} \right\rceil (d+1) - 1, d, k - \left\lceil \frac{d}{k-d} \right\rceil (k-d) - 1, 0\right)$

TABLE III
MERGING RULE IN CASE $0 < D < K < \infty \wedge L + R \leq \lceil D / (K - D) \rceil (K - D) + K \wedge R \geq \lceil D / (K - D) \rceil (K - D) + 1$

s, t	β
$s \leq d \wedge t \leq d$	$\sigma\left(\left\lceil \frac{d}{k-d} \right\rceil (d+1), d, s, t\right)$
$s \geq d+1 \wedge t \leq d$	$\chi\left(\left\lceil \frac{d}{k-d} \right\rceil (d+1), d, 0, t\right)$
$s \leq d \wedge t \geq d+1$	$\text{Rev}\left[\chi\left(\left\lceil \frac{d}{k-d} \right\rceil (d+1), d, 0, s\right)\right]$
$d+1 \leq s \leq \left\lceil \frac{d}{k-d} \right\rceil (k-d) \wedge t \geq d+1$	$\psi\left(\left\lceil \frac{d}{k-d} \right\rceil (d+1), d, k - \left\lceil \frac{d}{k-d} \right\rceil (k-d), 0\right)$
$s \geq \left\lceil \frac{d}{k-d} \right\rceil (k-d) + 1 \wedge t \geq d+1$	$\psi\left(\left\lceil \frac{d}{k-d} \right\rceil (d+1), d, k - s, s - \left\lceil \frac{d}{k-d} \right\rceil (k-d)\right)$

TABLE IV
MERGING RULE IN CASE $0 < D < K < \infty \wedge L + R \geq \lceil D / (K - D) \rceil (K - D) + K + 1$

s, t	β
$s \leq d \wedge t \leq d$	$\sigma\left(\left\lceil \frac{d}{k-d} \right\rceil (d+1) + 1, d, s, t\right)$
$s \geq d+1 \wedge t \leq d$	$\chi\left(\left\lceil \frac{d}{k-d} \right\rceil (d+1) + 1, d, 0, t\right)$
$s \leq d \wedge t \geq d+1$	$\text{Rev}\left[\chi\left(\left\lceil \frac{d}{k-d} \right\rceil (d+1) + 1, d, 0, s\right)\right]$
$s \geq d+1 \wedge t \geq d+1$	$\psi\left(\left\lceil \frac{d}{k-d} \right\rceil (d+1) + 1, d, 0, 0\right)$

length $d + v + 1$, which also satisfies the constraint

$$\begin{aligned}
 d \leq d + v \leq d + v + 1 &= d + 1 + \left\lfloor \frac{p}{q} \right\rfloor = d + 1 + \left\lfloor \frac{p-1}{q} \right\rfloor \\
 &\leq d + 1 + \left\lfloor \frac{d-1}{q} \right\rfloor \\
 &= d + 1 + \left\lfloor \frac{\left\lceil \frac{d}{k-d} \right\rceil (d+1) + 1 + t}{d+1} \right\rfloor
 \end{aligned}$$

$$\leq d + 1 + \left\lfloor \frac{d-1}{\left\lceil \frac{d}{k-d} \right\rceil} \right\rfloor \leq d + 1 + \left\lfloor \frac{d-1}{d} (k-d) \right\rfloor \leq k.$$

Hence, we have established the following theorem.

Theorem 3: For $0 < d < k < \infty$ and $0 \leq l \leq r \leq k$,

$$B_{dklr} = \begin{cases} d & \text{if } l = r = 0 \\ \left\lceil \frac{l+r}{k-d} \right\rceil (d+1) - 1 & \text{if } 1 \leq l+r \leq d \\ \left\lceil \frac{d}{k-d} \right\rceil (d+1) - 1 & \text{if } l+r \geq d+1 \wedge r \leq \left\lceil \frac{d}{k-d} \right\rceil (k-d) \\ \left\lceil \frac{d}{k-d} \right\rceil (d+1) & \text{if } l+r \leq \left\lceil \frac{d}{k-d} \right\rceil (k-d) + k \\ \wedge r \geq \left\lceil \frac{d}{k-d} \right\rceil (k-d) + 1 & \\ \left\lceil \frac{d}{k-d} \right\rceil (d+1) + 1 & \text{if } l+r \geq \left\lceil \frac{d}{k-d} \right\rceil (k-d) + k + 1. \end{cases}$$

Proof: Combine merging rule (16) with Lemmas 1-4. $\square$

In most practical situations, the upper runlength value k is at least twice the lower runlength value d . For these cases, we have the following corollary.

Corollary 1: For $0 < 2d \leq k < \infty$ and $0 \leq l \leq r \leq k$,

$$B_{dklr} = \begin{cases} d & \text{if } r \leq k - d \\ d + 1 & \text{if } l + r \leq 2k - d \wedge r \geq k - d + 1 \\ d + 2 & \text{if } l + r \geq 2k - d + 1. \end{cases}$$

Proof: Apply Theorem 3, noting that $\lceil d/(k-d) \rceil = 1$ if $k \geq 2d > 0$. $\square$

V. FIXED-LENGTH CODES BASED ON $dklr$ -SEQUENCES

In this section, we consider a coding scheme in which data are converted into a (d, k) constrained bit stream. In this scheme, messages are one-to-one mapped to $dklr$ -sequences of length n . Between any two of such sequences, we insert b appropriate merging bits. In order to keep the encoding and decoding simple, we choose both n and b to be fixed. The actual number of bits assigned to a message is thus $N = n + b$.

The one-to-one mapping can be established by a look-up table, in which each message corresponds to a unique $dklr$ -sequence. For long codewords, the size of the look-up table could lead to insuperable practical problems, so a more structured technique like the one described in [1] must be used. Mostly, the choice of the merging bits depends on the tail of the preceding sequence and the head of the next sequence, in which case some buffering is necessary. One could also make the merging bits independent of the surrounding bits, as was done in "Construction 1" in [1], for which some price must be paid in the code rate. In this case, the merging bits may be attached to the codewords such that the resulting sequences are concatenatable without any further need for merging bits [5].

Let the message set be denoted by $\mathcal{A}$ and its size by A . In most applications, $\mathcal{A}$ consists of all binary sequences of a certain length a , where $A = 2^a$. In the remainder of this section, let the lower runlength constraint d , the upper runlength constraint k , and the number of messages A be given. Our goal will be to find choices for l, r, n , and b such that N , the number of bits assigned to a message from $\mathcal{A}$ in the (d, k) constrained bit stream, is minimized or, in other words, the code rate $R = (\log_2 A)/N$ is maximized. In order to compare such a code rate with the maximum achievable rate of any code with the same runlength constraints, we define the code efficiency E by

$$E = \frac{R}{C_{dk}} \quad (17)$$

where C_{dk} is the capacity of a (d, k) constrained sequence, which can be defined by

$$C_{dk} = \lim_{n \rightarrow \infty} \frac{\log_2 M_{dkkk}(n)}{n}. \quad (18)$$

Extended tables on the capacity can be found in [3], [6]. It should be pointed out that the discrepancy between R and C_{dk} is due to the finiteness of A , i.e., if A is allowed to increase, the efficiency E of the coding scheme will approach one.

The above-described coding scheme has been pioneered by Tang and Bahl [8] and Beenker and Immink [1]. Tang and Bahl considered the case $l = r = k$. For $0 < 2d \leq k < \infty$, they gave a merging rule needing $d + 2$ merging bits. Beenker and Immink introduced the $dklr$ -concept, and came up with the idea of reducing the maximum number of starting and ending "0's" in a sequence in order to lower the necessary number of merging bits. For $0 < 2d \leq k < \infty$, their method with $l = r = k - d$ uses

only d merging bits. For some examples considered in [1], Beenker and Immink's method turned out to perform better than the one of Tang and Bahl. As we will prove in this section, this is true in general. However, we will also see that it is possible sometimes to improve on Beenker and Immink's method in efficiency. Furthermore, we will also consider the values of d and k outside the $0 < 2d \leq k < \infty$ region.

Just as in the previous section, we will first consider the case in which there is no upper runlength constraint ($k = \infty$), then the case in which there is no lower runlength constraint ($0 = d < k < \infty$), and finally the case in which there are both a lower and an upper runlength constraint ($0 < d < k < \infty$).

A. The Case $k = \infty$

It follows from Theorem 1 that the minimal number of merging bits is independent of l and r if $k = \infty$. Hence, we choose l and r as large as possible: $l = r = \infty$. The code rate is then maximized by mapping $\mathcal{A}$ to a subset of $\mathcal{M}_{d\infty\infty}(n^*)$, where n^* is the smallest positive integer such that $M_{d\infty\infty}(n^*) \geq A$. Using merging rule (14), all $b = d$ merging bits are set to "0." Hence, a code rate of

$$R = \frac{\log_2 A}{n^* + d} \quad (19)$$

can be obtained.

As an example, we consider messages of 4 bits that are converted into a $(4, \infty)$ constrained bit stream. Hence, $A = 2^4 = 16$, and it follows from Table I that $n^* = 9$. So, a code rate $R = 4/(9 + 4) = 4/13 = 0.308$ can be achieved. Since the capacity equals $C_{4\infty} = 0.406$, we have a code efficiency $E = 0.76$.

In Table V, we give more examples of the maximum code rates and efficiencies achieved by this scheme, when converting messages of a bits into a (d, ∞) constrained bit stream. Note that $R = A/(n^* + d)$ and $E = R/C_{d\infty}$.

B. The Case $0 = d < k < \infty$

It follows from Theorem 2 and merging rule (15) that we can use either no merging bits ($b = 0$) or a single merging bit which is set to "1" ($b = 1$) if $0 = d < k < \infty$. If $b = 0$, then l and r must satisfy $l + r \leq k$. Furthermore, the all-zero sequence, which is in $\mathcal{M}_{0klr}(n)$ if and only if $n \leq l$, cannot be included in the code since repeated occurrence of this sequence may violate the upper runlength constraint. With (11), it follows that $M_{0klr}(n)$ is maximized over all l and r satisfying $l + r \leq k$ by choosing $l = \lfloor k/2 \rfloor$ and $r = \lfloor k/2 \rfloor$. If $b = 1$, then we can choose $l = r = k$, and the inclusion of the all-zero sequence into the code does not cause problems. Hence, in order to maximize the code rate, one should determine the smallest positive integer n_1^* such that $M_{0kkk}(n_1^*) \geq A$, and the smallest positive integer n_0^* such that $M_{0k\lfloor k/2 \rfloor \lfloor k/2 \rfloor}(n_0^*) \geq A + q$, where $q = 1$ if $n_0^* \leq \lfloor k/2 \rfloor$ and $q = 0$ otherwise. If $n_0^* \leq n_1^*$, we should use the method without merging bits; if $n_0^* = n_1^* + 1$, either method can be used; and if $n_0^* \geq n_1^* + 2$, we should use the method with the single merging bit.

As an example, we consider messages of 3 bits that are converted into a $(0, 2)$ constrained bit stream. Hence, $A = 2^3 = 8$, and so it follows from Table I that $n_0^* = n_1^* = 4$. The method using no merging bits thus gives a code rate $R = 3/(4 + 0) = 0.75$, while the method using the single merging bit gives a code rate $R = 3/(4 + 1) = 0.6$. Since $C_{02} = 0.879$, the efficiencies are $E = 0.85$ and $E = 0.68$, respectively.

In the previous example, the method without a merging bit performed better than the one with a merging bit. The following theorem shows that this is no coincidence.

TABLE V
CODE RATES AND EFFICIENCIES FOR $D = 1, 2, 3$ AND $K = \infty$

a	$d = 1$ $C_{1\infty} = 0.694$			$d = 2$ $C_{2\infty} = 0.552$			$d = 3$ $C_{3\infty} = 0.465$		
	n^*	R	E	n^*	R	E	n^*	R	E
1	1	0.500	0.72	1	0.333	0.60	1	0.250	0.54
2	3	0.500	0.72	3	0.400	0.72	3	0.333	0.72
3	4	0.600	0.86	5	0.429	0.78	6	0.333	0.72
4	6	0.571	0.82	7	0.444	0.80	8	0.364	0.78
5	7	0.625	0.90	9	0.455	0.82	10	0.385	0.83
6	9	0.600	0.86	11	0.462	0.84	12	0.400	0.86
7	10	0.636	0.92	12	0.500	0.91	14	0.412	0.89
8	12	0.615	0.89	14	0.500	0.91	17	0.400	0.86
12	17	0.667	0.96	22	0.500	0.91	25	0.429	0.92
16	23	0.667	0.96	29	0.516	0.93	34	0.432	0.93
32	46	0.681	0.98	58	0.533	0.97	68	0.451	0.97

Theorem 4: For $0 < k < \infty$ and $n \geq 1$,

$$M_{0kkk}(n) \leq M_{0k\lfloor k/2 \rfloor \lfloor k/2 \rfloor}(n+1)$$

where equality occurs if and only if $k = 1$.

Proof: We first give a one-to-one map ϕ from $\mathcal{M}_{0kkk}(n)$ into $\mathcal{M}_{0k\lfloor k/2 \rfloor \lfloor k/2 \rfloor}(n+1)$. Each $\gamma \in \mathcal{M}_{0kkk}(n)$ can be uniquely written as $\gamma = \delta 0^j$, where δ is either a sequence ending with a "1" or the empty sequence ϵ (of length zero) and $0 \leq j \leq k$. Now, define ϕ by

$$\phi(\gamma) = \phi(\delta 0^j) = \begin{cases} 1\delta 0^j & \text{if } j \leq \lfloor k/2 \rfloor \\ 01\delta 0^{j-2}1 & \text{otherwise.} \end{cases} \quad (20)$$

To show that ϕ is indeed one-to-one, note that $\phi(\gamma)$ starts with a "1" if $j \leq \lfloor k/2 \rfloor$ and with a "0" otherwise. Hence, from $\phi(\gamma)$, we can determine which case applies to γ , and next we can determine in each case δ and the value of j , and thus γ . This shows that ϕ is one-to-one, and so $M_{0k\lfloor k/2 \rfloor \lfloor k/2 \rfloor}(n+1) \geq M_{0kkk}(n)$.

To show strict inequality for $k > 1$, consider the sequence η defined by $\eta = 01$ if $n = 1$ and by $\eta = 01^{n-1}0$ otherwise. Notice that $\eta \in \mathcal{M}_{0k\lfloor k/2 \rfloor \lfloor k/2 \rfloor}(n+1)$. However, it can be seen from (20) that η is not the image of any sequence in $\mathcal{M}_{0kkk}(n)$ under ϕ .

To show equality for $k = 1$, it is sufficient to give a one-to-one map τ from $\mathcal{M}_{0101}(n+1)$ into $\mathcal{M}_{0111}(n)$. Let $\mu \in \mathcal{M}_{0101}(n+1)$. The image $\tau(\mu)$ is obtained by deleting the first symbol of μ . Since this first symbol is always a "1," τ is indeed one-to-one. $\square$

It can be concluded from Theorem 4 that $n_0^* \leq n_1^* + 1$. Hence, the method using no merging bits always maximizes the code rate. The coding scheme leading to the maximal code rate, for the case in which there is no lower runlength constraint, thus reduces to determining the smallest positive integer n^* such that $M_{0k\lfloor k/2 \rfloor \lfloor k/2 \rfloor}(n^*) \geq A + q$, where $q = 1$ if $n^* \leq \lfloor k/2 \rfloor$ and $q = 0$ otherwise, and next defining a one-to-one mapping from $\mathcal{A}$ to a subset of $\mathcal{M}_{0k\lfloor k/2 \rfloor \lfloor k/2 \rfloor}(n^*)$ not containing the all-zero sequence. No merging bits are needed. The maximal code rate reads

$$R = \frac{\log_2 A}{n^*}. \quad (21)$$

In Table VI, we give examples of the maximum code rates and efficiencies achieved by this scheme, when converting messages of a bits into a $(0, k)$ constrained bit stream. Note that $R = a/n^*$ and $E = R/C_{0k}$.

TABLE VI
CODE RATES AND EFFICIENCIES FOR $D = 0$ AND $K = 1, 2, 3$

a	$k = 1$ $C_{01} = 0.694$			$k = 2$ $C_{02} = 0.879$			$k = 3$ $C_{03} = 0.947$		
	n^*	R	E	n^*	R	E	n^*	R	E
1	2	0.500	0.72	2	0.500	0.57	2	0.500	0.53
2	4	0.500	0.72	3	0.667	0.76	3	0.667	0.70
3	5	0.600	0.86	4	0.750	0.85	4	0.750	0.79
4	7	0.571	0.82	5	0.800	0.91	5	0.800	0.84
5	8	0.625	0.90	7	0.714	0.81	6	0.833	0.88
6	10	0.600	0.86	8	0.750	0.85	7	0.857	0.90
7	11	0.636	0.92	9	0.778	0.89	8	0.875	0.92
8	13	0.615	0.89	10	0.800	0.91	9	0.889	0.94
12	18	0.667	0.96	15	0.800	0.91	14	0.857	0.90
16	24	0.667	0.96	19	0.842	0.96	18	0.889	0.94
32	47	0.681	0.98	37	0.865	0.98	35	0.914	0.97

C. The Case $0 < d < k < \infty$

Merging rule (16) can be used to cascade any two $dklr$ -sequences of length n without violating the (d, k) constraint in case $0 < d < k < \infty$. However, when cascading more than two sequences, problems may arise if the all-zero sequence of length n is a $dklr$ -sequence, i.e., if $n \leq l$. For example, we consider the case $d = 1, k = 3, l = r = 2$. According to Corollary 1, a single merging bit suffices to cascade any two sequences in $\mathcal{M}_{1322}(n)$. If $n \geq 3$, the all-zero sequence is not a $dklr$ -sequence, and no complications occur in cascading more than two sequences. If $n = 2$, cascading the sequences 01, 00, and 10 using a single merging bit causes insuperable problems. The merging bit following the first sequence and the merging bit preceding the third sequence must both be set to "0" to satisfy the d -constraint, which gives a run of four "0's," thus violating the k -constraint. If $n = 1$, the all-zero sequence is also a $dklr$ -sequence, but here problems can be avoided by choosing the merging bit "1" if both surrounding bits equal "0" and by choosing the merging bit "0" otherwise. In fact, this is the famous MFM-code with rate $R = 0.5$ (see [3]).

In order to make it possible to apply merging rule (16) without any reservations in case $0 < d < k < \infty$, we will abandon using the all-zero sequence. A study on adapting the rule such that this sequence could be included might lead to an improvement in the code rate for some particular cases, but this will not be dealt with here. Let $\hat{\mathcal{M}}_{dklr}(n)$ denote the set $\mathcal{M}_{dklr}(n)$ excluding the all-zero sequence. Let $\hat{M}_{dklr}(n)$ denote the cardinality of $\hat{\mathcal{M}}_{dklr}(n)$. Hence, $\hat{M}_{dklr}(n) = M_{dklr}(n) - q$, where $q = 1$ if $n \leq l$ and $q = 0$ otherwise.

We consider for each value of b which l and r maximize $\hat{M}_{dklr}(n)$ over all possible choices satisfying $B_{dklr} \leq b$. It can be deduced easily from (11), (16), and Theorem 3 that we should evaluate the options from Table VII for b, l , and r in order to optimize the code rate. Here, i ranges from 1 up to $\lfloor d/(k-d) \rfloor - 1$ and j is just an index allowing easy reference to the possible options. When coding a message set $\mathcal{A}$ of size A , the smallest positive integer n_j^* for each j is determined such that $\hat{M}_{dklr}(n_j^*) \geq A$, with l and r chosen according to Table VII. Hence, option j requires $n_j^* + b$ bits to be assigned for each message in $\mathcal{A}$, where b is chosen according to Table VII. The option j for which $n_j^* + b$ is minimal then optimizes the code rate.

As an example, we consider messages of 5 bits that are converted into a $(1, 3)$ constrained bit stream. It follows from Table VII that we should compare the options with indexes $j = 1$ (with $b = 3, l = r = 3$), $j = 2$ (with $b = 2, l = 2, r = 3$),

TABLE VII
OPTIONS FOR CHOOSING B, L , AND R IF $0 < D < K < \infty$

j	b	l	r
1	$\left\lfloor \frac{d}{k-d} \right\rfloor (d+1) + 1$	k	k
2	$\left\lfloor \frac{d}{k-d} \right\rfloor (d+1)$	$\left\lfloor \frac{\left\lfloor \frac{d}{k-d} \right\rfloor (k-d) + k}{2} \right\rfloor$	$\left\lfloor \frac{\left\lfloor \frac{d}{k-d} \right\rfloor (k-d) + k}{2} \right\rfloor$
3	$\left\lfloor \frac{d}{k-d} \right\rfloor (d+1) - 1$	$\left\lfloor \frac{d}{k-d} \right\rfloor (k-d)$	$\left\lfloor \frac{d}{k-d} \right\rfloor (k-d)$
$i+3$	$i(d+1) - 1$	$\left\lfloor \frac{i(k-d)}{2} \right\rfloor$	$\left\lfloor \frac{i(k-d)}{2} \right\rfloor$

TABLE VIII
CODE RATES AND EFFICIENCIES FOR $(D, K) = (1, 3), (2, 4), (5, 7)$

$(d, k) = (1, 3)$ $C_{13} = 0.552$ $j = 3$				$(d, k) = (2, 4)$ $C_{24} = 0.406$ $j = 2$				$(d, k) = (5, 7)$ $C = 0.228$ $j = 4$			
a	n^*	R	E	n^*	R	E		n^*	R	E	
3	5	0.500	0.91	6	0.333	0.82	15	0.150	0.66		
4	7	0.500	0.91	9	0.333	0.82	22	0.148	0.65		
5	9	0.500	0.91	11	0.357	0.88	28	0.152	0.67		
6	11	0.500	0.91	14	0.353	0.87	30	0.171	0.75		
7	13	0.500	0.91	16	0.368	0.91	36	0.171	0.75		
8	15	0.500	0.91	19	0.364	0.90	42	0.170	0.75		
12	22	0.522	0.95	29	0.375	0.92	58	0.190	0.83		
16	29	0.533	0.97	38	0.390	0.96	76	0.198	0.87		
32	58	0.542	0.98	78	0.395	0.97	146	0.212	0.93		

and $j = 3$ (with $b = 1, l = r = 2$). We can see from Table I that $n_1^* = 8, n_2^* = 9$, and $n_3^* = 9$. The rates are $R = 5/(8+3) = 0.455, R = 5/(9+2) = 0.455$, and $R = 5/(9+1) = 0.5$ for $j = 1, 2, 3$, respectively. Hence, the $j = 3$ option optimizes the code rate. Since $C_{13} = 0.552$, the efficiencies are $E = 0.82, E = 0.82$, and $E = 0.91$ for $j = 1, 2, 3$, respectively.

In Table VIII, we give more examples of the maximum code rates and efficiencies achieved by the scheme described in this subsection when converting messages of a bits into a (d, k) constrained bit stream. For all values of a in this table, the options $j = 3, 2$, and 4 are optimal for $(d, k) = (1, 3), (2, 4)$, and $(5, 7)$, respectively. As usual, $R = a/(n^* + b)$ and $E = R/C_{dk}$.

In the remainder of this subsection, we will investigate which of the options from Table VII optimizes the code rate. Therefore, we first derive a number of theorems that compare the different options. The $j = 1$ and the $j = 3$ options are compared in Theorem 5, the $j = 2$ and the $j = 3$ options in Theorem 6 (for $k \geq \lfloor 5d/2 \rfloor$) and Theorem 7 (for $k = 2d$), and finally, all options with $j \geq 4$ in Theorem 8. The general technique we use to establish that $\hat{M}_{dkl'r'}(n') \leq \hat{M}_{dkl'r}(n)$ is by exhibiting a one-to-one map ϕ from $\hat{\mathcal{M}}_{dkl'r'}(n')$ into $\hat{\mathcal{M}}_{dkl'r}(n)$. This technique was already used to establish Theorem 4. In the proofs, we will use the following notation. If $\gamma \in \hat{\mathcal{M}}_{dkl'r}(n)$, let $a(\gamma)$ denote the number of leading "0's" in γ , and let $b(\gamma)$ denote the number of trailing "0's" in γ . Hence, $0 \leq a(\gamma) \leq l, 0 \leq b(\gamma) \leq r$, and γ can be written uniquely as $\gamma = 0^{a(\gamma)}\delta(\gamma)0^{b(\gamma)}$, where $\delta(\gamma)$ is a sequence of length $n - a(\gamma) - b(\gamma)$, both starting and ending with a "1."

Theorem 5: For $0 < d < k < \infty, l = r = \lfloor d/(k-d) \rfloor (k-d)$, and $n \geq 1$,

$$\hat{M}_{dkkk}(n) \leq \hat{M}_{dklr}(n+2).$$

Proof: Let $\gamma \in \hat{\mathcal{M}}_{dkkk}(n)$. The result follows by observing that the map ϕ defined by

$$\phi(\gamma) = \begin{cases} 0\gamma 0 & \text{if } a(\gamma) < l \text{ and } b(\gamma) < r \\ 0\gamma 1 & \text{if } a(\gamma) < l \text{ and } b(\gamma) \geq r \\ 1\gamma 0 & \text{if } a(\gamma) \geq l \text{ and } b(\gamma) < r \\ 1\gamma 1 & \text{if } a(\gamma) \geq l \text{ and } b(\gamma) \geq r \end{cases}$$

is a one-to-one map from $\hat{\mathcal{M}}_{dkkk}(n)$ into $\hat{\mathcal{M}}_{dklr}(n+2)$. $\square$
Theorem 6: For $d > 0, k \geq \lfloor 5d/2 \rfloor, l = r = k - d, l' = \lfloor (2k - d)/2 \rfloor, r' = \lfloor (2k - d)/2 \rfloor$, and $n \geq 1$,

$$\hat{M}_{dkl'r'}(n) \leq \hat{M}_{dklr}(n+1).$$

Proof: Let $\gamma \in \hat{\mathcal{M}}_{dkl'r'}(n)$. The result follows by observing that the map ϕ defined by

$$\phi(\gamma) = \begin{cases} 0^{a(\gamma)}\delta(\gamma)0^{b(\gamma)+1} & \text{if } a(\gamma) \leq l \text{ and } b(\gamma) < r \\ 0^{a(\gamma)}\delta(\gamma)0^{b(\gamma)}1 & \text{if } a(\gamma) \leq l \text{ and } b(\gamma) \geq r \\ 0^{b(\gamma)+k-2d+1}\delta(\gamma)0^{a(\gamma)-k+2d-1}1 & \text{if } a(\gamma) > l \text{ and } b(\gamma) < d \\ 0^{k-\lfloor 5d/2 \rfloor}10^{b(\gamma)+\lfloor d/2 \rfloor}\delta(\gamma)0^{a(\gamma)-k+2d-1}1 & \text{if } a(\gamma) > l \text{ and } b(\gamma) \geq d \end{cases}$$

is a one-to-one map from $\hat{\mathcal{M}}_{dkl'r'}(n)$ into $\hat{\mathcal{M}}_{dklr}(n+1)$. $\square$

Theorem 7: For $d > 0, k = 2d, l = \lfloor 3d/2 \rfloor, r = \lfloor 3d/2 \rfloor$, and $n \geq 2d + 1$,

$$\hat{M}_{dkdd}(n+1) \leq \hat{M}_{dklr}(n).$$

Proof: Let $\gamma \in \hat{\mathcal{M}}_{dkdd}(n+1)$. Since $n \geq 2d + 1$, the weight of γ is at least two. Hence, γ can be written uniquely as

$$\gamma = 0^{a(\gamma)}\delta(\gamma)0^{b(\gamma)} = 0^{a(\gamma)}10^{c(\gamma)}\tau(\gamma)0^{b(\gamma)}$$

where $d \leq c(\gamma) \leq 2d$ and $\tau(\gamma)$ is a sequence of length $n - a(\gamma) - b(\gamma) - c(\gamma)$, both starting and ending with a "1." The result now follows by observing that the map ϕ defined by

$$\phi(\gamma) = \begin{cases} 0^{a(\gamma)-1}10^{c(\gamma)}\tau(\gamma)0^{b(\gamma)} & \text{if } a(\gamma) > 0 \\ 0^{c(\gamma)}\tau(\gamma)0^{b(\gamma)} & \text{if } a(\gamma) = 0 \text{ and } c(\gamma) \leq l \\ 0^{b(\gamma)+\lfloor d/2 \rfloor}\tau(\gamma)0^{c(\gamma)-\lfloor d/2 \rfloor} & \text{if } a(\gamma) = 0 \text{ and } c(\gamma) > l \end{cases}$$

is a one-to-one map from $\hat{\mathcal{M}}_{dkdd}(n+1)$ into $\hat{\mathcal{M}}_{dklr}(n)$. $\square$

Theorem 8: For $0 < d < k < 2d$, $1 \leq i \leq \lceil d/(k-d) \rceil - 2$, $l = \lfloor i(k-d)/2 \rfloor$, $r = \lfloor i(k-d)/2 \rfloor$, $l' = \lfloor (i+1)(k-d)/2 \rfloor$, $r' = \lfloor (i+1)(k-d)/2 \rfloor$, and $n \geq 1$,

$$\hat{M}_{dkl'r'}(n) \leq \hat{M}_{dklr}(n+d+1).$$

Proof: Let $\gamma \in \hat{\mathcal{M}}_{dkl'r'}(n)$. The result follows by observing that the map ϕ defined by

$$\phi(\gamma) = \begin{cases} 0^{a(\gamma)}\delta(\gamma)0^{d-10^{b(\gamma)}} & \text{if } a(\gamma) < l \text{ and } b(\gamma) < r \\ 0^{a(\gamma)}\delta(\gamma)0^{d+b(\gamma)-r}10^r & \text{if } a(\gamma) < l \text{ and } b(\gamma) \geq r \\ 0\delta(\gamma)0^{d+a(\gamma)-l}10^{b(\gamma)} & \text{if } a(\gamma) \geq l \text{ and } b(\gamma) < r \\ 0\delta(\gamma)0^{d+a(\gamma)+b(\gamma)-l-r}10^r & \text{if } a(\gamma) \geq l \text{ and } b(\gamma) \geq r \end{cases}$$

is a one-to-one map from $\hat{\mathcal{M}}_{dkl'r'}(n)$ into $\hat{\mathcal{M}}_{dklr}(n+d+1)$. $\square$

From Theorems 5–8, we can draw the following conclusions.

- If $k \geq \lfloor 5d/2 \rfloor$, which is the case in almost all practical applications, e.g., $(d, k) = (1, 7)$ and $(2, 7)$, the most efficient coding is achieved by the $j = 3$ option, i.e., by the method of Beenker and Immink [1] which they call “Construction 2.”
- If $2d < k < \lfloor 5d/2 \rfloor$, the most efficient coding is achieved by one (or both) of the options with $j \in \{2, 3\}$.
- If $k = 2d$, the most efficient coding is achieved by the $j = 2$ option for sufficiently large codes.
- If $d < k < 2d$, the most efficient coding is achieved by one (or more) of the options with $j \in \{2, 3, 4\}$.

From the above-mentioned conclusions, it follows that in the $d < k < \lfloor 5d/2 \rfloor$ range, the most efficient method has not been completely determined yet, except for $k = 2d$. In the latter case, the $j = 2$ option maximizes the code rate. Here, we find situations for which Beenker and Immink’s method ($j = 3$) is not optimal. For example, if $d = 2$, $k = 4$, and $A = 2^5 = 32$, the $j = 2$ option gives a rate $R = 5/(11 + 3) = 0.357$, while the $j = 3$ option gives only $R = 5/(13 + 2) = 0.333$. As candidate options for maximum efficiency, we should compare $j = 2$ and $j = 3$ if $2d < k < \lfloor 5d/2 \rfloor$, and $j = 2$, $j = 3$, and $j = 4$ if $d < k < 2d$. For (d, k) satisfying $d < k < \lfloor 5d/2 \rfloor$ and $2 \leq d \leq 10$, we have computed $M_{dklr}(n-b)$, with b, l , and r chosen according to Table VII, starting from $n = k + l + b + 2$ until one option was at least as good as the other(s) for $k + 1$ consecutive values of n . This option will then also be optimal for larger lengths because of (6). The results are shown in Table IX. For example, if $(d, k) = (5, 6)$, the $j = 4$ option gives maximal efficiency. The results need not be unique. For example, if $(d, k) = (2, 3)$, the $j = 2$ option, and even the $j = 1$ option provide equally good efficiency as the indicated $j = 3$ option.

VI. CONCLUSIONS

In this paper, we have given general merging rules for cascading two $dklr$ -sequences, i.e., (d, k) constrained sequences starting with at most l “0’s” and ending with at most r “0’s.” It has been proved that these rules are optimal in the sense that they minimize the (fixed) number of merging bits needed to avoid violating the (d, k) constraints. The merging rules have been applied to a (well-known) fixed-length coding scheme, in which a message is uniquely mapped to a $dklr$ -sequence of fixed length n . Different options for choosing the values of l and r in order to maximize the code rate have been considered. For the practical cases with $k \geq \lfloor 5d/2 \rfloor$, it has been shown that the scheme provides the best results when using the merging method of Beenker and Immink, which can be considered as a special case of the general merging rules.

TABLE IX
INDEXES J OF OPTIONS OPTIMIZING THE CODE RATE FOR LARGE
CODES FOR $D < K < \lfloor 5D/2 \rfloor$ AND $2 \leq D \leq 10$

k, d	2	3	4	5	6	7	8	9	10
3	3								
4	2	3							
5		3	3						
6		2	2	4					
7			3	4					
8			2	3	4	4			
9			3	4	2	4	4		
10				2	3	4	4	4	
11				3	4	3	4	4	4
12					2	4	2	4	4
13					3	4	4	4	4
14					3	2	4	4	4
15						2	4	4	2
16						3	2	4	4
17							2	4	4
18							3	2	4
19							3	2	4
20								3	2
21								3	2
22									3
23									3
24									3

Finally, we should point out that Fuja and Gu [2], as well as Tjalkens [9], have recently proposed a different approach for mapping messages into runlength-limited sequences and cascading these sequences while maintaining the runlength constraints. In their approach, no merging bits are needed, but the choice of the sequence representing a message depends on the sequence representing the previous message. In our approach, only the merging bits depend on the previous message. Therefore, the scheme described in this paper allows a somewhat simpler encoder than the scheme of [2] and [9], possibly at the cost of a slight reduction in code rate.

ACKNOWLEDGMENT

It is a pleasure to thank M. E. Jong Baw for her contributions to some of the results presented in this paper.

REFERENCES

- [1] G. F. M. Beenker and K. A. S. Immink, “A generalized method for encoding and decoding run-length-limited binary sequences,” *IEEE Trans. Inform. Theory*, vol. IT-29, pp. 751–754, Sept. 1983.
- [2] T. Fuja and J. Gu, “A new method for constructing block codes for runlength-limited channels,” in *Proc. 1992 IEEE Inform. Theory Workshop*, Salvador, Brazil, June 1992, pp. 101–102.
- [3] K. A. S. Immink, “Runlength-limited sequences,” *Proc. IEEE*, vol. 78, pp. 1745–1759, Nov. 1990.
- [4] —, *Coding Techniques for Digital Recorders*. Englewood Cliffs, NJ: Prentice-Hall, 1991.
- [5] P. Lee and J. K. Wolf, “A general error-correcting code construction for run-length limited binary channels,” *IEEE Trans. Inform. Theory*, vol. 35, pp. 1330–1335, Nov. 1989.
- [6] K. Norris and D. S. Bloomberg, “Channel capacity of charge-constrained runlength limited codes,” *IEEE Trans. Magn.*, vol. MAG-17, pp. 3452–3455, Nov. 1981.
- [7] P. H. Siegel, “Recording codes for digital magnetic storage,” *IEEE Trans. Magn.*, vol. MAG-21, pp. 1344–1349, Sept. 1985.
- [8] D. T. Tang and L. R. Bahl, “Block codes for a class of constrained noiseless channels,” *Inform. Contr.*, vol. 17, pp. 436–461, 1970.
- [9] Tj. Tjalkens, “Glueless runlength limited sequences,” in *Proc. 13th Symp. Inform. Theory in the Benelux*, Enschede, The Netherlands, June 1992, pp. 57–64.