Efficient encoding of constrained block codes

Kees A. Schouhamer Immink, Fellow, IEEE, and Kui Cai, Senior Member, IEEE

***Abstract*—We present coding methods for generating ℓ-symbol constrained codewords taken from a set, $\mathcal{S}$, of allowed codewords. In standard practice, the size of the set $\mathcal{S}$, denoted by $M = |\mathcal{S}|$, is truncated to an integer power of two, which may lead to a serious waste of capacity. We present an efficient and low-complexity coding method for avoiding the truncation loss, where the encoding is accomplished in two steps: first, a series of binary input (user) data is translated into a series of M-ary symbols in the alphabet $\mathbb{M} = \{0, \ldots, M-1\}$. Then, in the second step, the M-ary symbols are translated into a series of admissible ℓ-symbol words in $\mathcal{S}$ by using a small look-up table. The presented construction of Pearson codes and fixed-weight codes offers a rate close to capacity. For example, the presented 255B320B balanced code, where 255 source bits are translated into 32 10-bit balanced codewords, has a rate 0.1 % below capacity.**

Keywords– constrained code, code design, binary block code, balanced code, Pearson code.

I. Introduction

Constrained codes have found widespread application in a large variety of communication systems such as cable transmission [1, 2], vehicular communications systems, visible light communications (VLC) systems [3], and data storage products ranging from magnetic, optical, solid-state (Flash), and DNA. *Runlength limited* codes [4] (RLL) use codewords with restrictions on the minimum and maximum runlength (that is, the number of consecutive like symbols) of the encoded sequence. RLL codes are ubiquitous in optical disc and magnetic recording products [5], VLC [3, 6], and DNA-based storage media [7, 8]. Balanced and almost balanced codes employ codewords with equal, or almost equal, numbers of 1's and 0's [5]. A typical example of an almost balanced code is the 8B10B code. The 8B10B code has many embodiments [2, 9], and is widely used in gigabit telecommunication systems and data storage media. Combinations of RLL and balanced codes can be found in data storage, energy harvesting, and communications codes [10, 11, 12, 13].

The codewords of a constrained block code are taken from a selected repertoire, $\mathcal{S}$, $\mathcal{S} \subseteq \mathbb{Q}^\ell$, of admissible codewords $\boldsymbol{x} = (x_1, x_2, \ldots, x_\ell)$, $x_i \in \mathbb{Q}$. The number of admissible codewords is denoted by $M = |\mathcal{S}|$, the size of $\mathcal{S}$. We do not concern ourselves with the selection or pairing of codewords. Each admissible codeword will be uniquely represented by an integer symbol taken from the alphabet $\mathbb{M} = \{0, \ldots, M-1\}$. The information capacity, denoted by C, of the channel using constrained codewords equals

$$C = \log_2 M. \tag{1}$$

In standard practice, the size of the original set $\mathcal{S}$ of admissible codewords is truncated to the nearest integer power of two, $2^{\lfloor C \rfloor}$, by judiciously deleting the surplus, $M - 2^{\lfloor C \rfloor}$, words, which may lead to a serious waste of capacity. A notorious example is the binary *Pearson code*, where only one word, the all-1 or all-0 word, is excluded [14]. With prior art fixed-block codes the rate equals $(\ell - 1)/\ell$, which entails a significant redundancy for small ℓ. Another well-known example is $M = 252$ (the number of 10-bit balanced (binary) codewords with equal numbers of 1's and 0's), where the truncation to $2^7 = 128$ codewords leads to an information rate waste of around 10 %. By combining ten 10-bit balanced words, we can translate 79 bits into ten 10-bit balanced codewords. The redundancy is less than a percent, but the improvement comes at a higher complexity of the look-up translation tables. There is a need to improve the rate efficiency of constrained codes without complex look-up tables.

We present and investigate a new encoding procedure that aims to improve the code rate efficiency without the need to use large look-up tables. The new encoding method is accomplished in two steps. First, the key step, binary source data are efficiently translated into a series of integer symbols in the alphabet $\mathbb{M}$ that are conveniently represented by q-bit binary words, where q an integer satisfying $q = \lceil C \rceil$. In the second step, the series of q-bit words is translated into a series of admissible codewords in $\mathcal{S}$. The first encoding step scales linearly with the number of symbols in the codeword.

The paper is organized as follows. In Section II, we start a survey of properties of the radix conversion scheme. Section III presents results of the new coding technique. An alternative scheme, the *variable length to fixed length* scheme, is investigated in Section IV. Applications to binary Pearson codes and balanced codes are given in Section V. We present a high-rate 255B320B balanced code, where 255 source bits are translated into 32 10-bit balanced codewords. Our conclusions are presented in Section VI.

II. Radix conversion scheme

A straightforward method for translating an n-bit source file into a series of integer symbols in the alphabet $\mathbb{M}$ is *base* or *radix conversion*. The binary n-bit input word, considered as a number in radix 2, is converted into L_o integer symbols in $\mathbb{M}$, where L_o is a user-defined positive integer [15]. The L_o radix-M integer, in turn, is translated, using a look-up table, into the corresponding admissible word. The number of distinct integers that can be addressed with L_o symbols in an

Kees A. Schouhamer Immink is with Turing Machines Inc, Willemskade 15d, 3016 DK Rotterdam, The Netherlands. E-mail: immink@turing-machines.com.

Kui Cai is with Singapore University of Technology and Design (SUTD), Science, Mathematics, and Technology Cluster, 8 Somapah Rd, 487372, Singapore. E-mail: cai_kui@sutd.edu.sg.

This work is supported by Singapore Ministry of Education Academic Research Fund Tier 2 MOE2019-T2-2-123 and RIE2020 Advanced Manufacturing and Engineering (AME) programmatic grant A18A6b0057.

M-radix system equals M^{L_o}, so that for a code to exist we have $2^n \leq M^{L_o}$, or

$$n = \lfloor L_o C \rfloor. \tag{2}$$

An integer symbol in the alphabet $\mathbb{M}$ can carry at most C bits, so it is natural to define the *rate efficiency* of an encoder as the quotient of the (average) number of bits that are translated per symbol and the capacity C. Then, the rate efficiency of the radix-2-to-M conversion, denoted by $R_o(L_o)$, equals

$$R_o(L_o) = \frac{n}{L_o C} = \frac{\lfloor L_o C \rfloor}{L_o C}. \tag{3}$$

The rate efficiency of a simple look-up table, $L_o = 1$, equals $R_o(1) = \lfloor C \rfloor / C$. The best case rate efficiency is obtained when M is an integer power of two, then $R_o(L_o) = 1$, and the coding step is lossless.

Example *1:* Let $M = 252$. Then, we can transmit at most $C = \log_2(252) = 7.977$ bits per symbol. A simple binary encoder, using a look-up table, has a rate $R_o(1) = 7/7.977$, which implies a 10 % relative rate loss. Let, for example, $L_o = 10$, then, $n = 79$, so that the (relative) code redundancy of the radix conversion scheme equals $1 - R_o(10) = 1 - 79/79.77 \approx 0.0097$. ■

A drawback of the radix conversion scheme, unless M is an integer power of two, is the increasing complexity with growing codeword length n as we require n-bit addition/subtraction units and storage of the $(M-1) \times L_o$ n-bit wide coefficients. Let M be close to an integer power of two, say $M = 2^u + v$, where u, v are two integers, $u > 0$, $|v| \ll 2^u$, then $C \approx u + \frac{v}{\ln(2) 2^u}$. Let $v > 0$, then $R_o(1) = u/C$, and we simply find that $R_o(L_o) > R_o(1)$ for $L_o \gtrapprox 2^u \ln(2)/v \approx 0.693\, 2^u/v$. In other words, in order to improve the rate with respect to that of a simple look-up table, $R_o(1)$, we must increase L_o. For example, let $M = 33$, then $C = \log_2(33) \approx 5.044394118$ and $R_o(1) \approx 5/C = 0.9911$. We easily find that $L_o' = 23$ is the smallest L_o that can increase the rate to $R_o(23) = \lfloor C L_o' \rfloor / C L_o' = 116/115\, R_o(1)$. In the next section, we describe a simple method for efficiently generating codewords that has less concerns with respect to complexity.

III. Description of the New Coding Method

A. Basic encoder

An integer in $\mathbb{M}$ is represented by a q-bit word, $q = \lceil \log_2 M \rceil$, taken from a constrained set, $\mathcal{C}$, where $|\mathcal{C}| = M$. The aim of the new coding method is to efficiently translate binary source data into a series of q-bit words in $\mathcal{C}$. The binary source data are assumed to be represented by $(n-1)$-bit words, denoted by $(a_1, \ldots, a_{n-1})$, $a_i \in \mathbb{B} = \{0, 1\}$, where n is a conveniently chosen integer. The $(n-1)$-bit source word is translated, using the new algorithm, into L q-bit words, where $qL = n$. The q-bit words are denoted by $\boldsymbol{u}_i$, $1 \leq i \leq L$, where the first word, $\boldsymbol{u}_1 = (p, a_1, \ldots, a_{q-1})$, called *pivot word*, contains $q-1$ user bits plus a redundant bit called *pivot bit*, denoted by p, $p \in \mathbb{B}$. The value of the pivot bit, p, is governed by the encoder, see later. The remaining $(L-1)$ q-bit words are defined by a shuffled input: $\boldsymbol{u}_i = (a_{(i-1)q}, \ldots, a_{iq-1})$, $2 \leq i \leq L$.

B. Description of the encoding and decoding algorithms

For clerical convenience we define two functions: $\mathrm{dec}(\boldsymbol{y})$ denotes the decimal representation of the q-bit word $\boldsymbol{y} = (y_1, \ldots, y_q)$, and, *vice versa*, $\boldsymbol{y} = \mathrm{bin}_q(z)$ denotes the q-bit binary representation, $\boldsymbol{y}$, of the integer z, $0 \leq z \leq 2^q - 1$. Clearly, $\mathrm{dec}(\boldsymbol{y}) = z$. All variables are integers, the bold face variables denote q-bit words. Let $w = 2^q - M$ denote the number of inadmissible words, $\boldsymbol{u}$, $\mathrm{dec}(\boldsymbol{u}) < w$. At the conclusion of the algorithm, all inadmissible q-bit words, $\boldsymbol{u}_i$, $\mathrm{dec}(\boldsymbol{u}_i) < w$, are eliminated and replaced by admissible q-bit words, so that $\mathrm{dec}(\boldsymbol{u}_i) \geq w, \forall i$.

Encoding routine

Input: The integers q, $w = 2^q - M$, L, and the binary $(Lq-1)$-bit source data $(a_1, \ldots, a_{Lq-1})$.
Output: Series of encoded q-bit words $\boldsymbol{u}_i$, where $\mathrm{dec}(\boldsymbol{u}_i) \geq w$, $1 \leq i \leq L$.
Initialize: Define the L q-bit words $\boldsymbol{u}_1 = (1, a_1, \ldots, a_{q-1})$ and $\boldsymbol{u}_i = (a_{(i-1)q}, \ldots, a_{iq-1})$, $2 \leq i \leq L$.
Set $v = 1$.
Replacing inadmissible words:
for $i = 2 : L$
  if $\mathrm{dec}(\boldsymbol{u}_i) < w$
  $\boldsymbol{u}_i = \boldsymbol{u}_v$; $\boldsymbol{u}_v = \mathrm{bin}_q((i-1)w + \mathrm{dec}(\boldsymbol{u}_i))$; $v = i$
  end
end.

Note that the pivot bit equals '1' in case the user data is sent unmodified, or it equals '0' in case at least one word has been modified. As a result, the receiver can easily detect that modifications have been effected. Decoding of the received codeword can be accomplished in a straightforward way by recursively undoing the replacements.

Decoding routine

Input: The integers q, $w = 2^q - M$, L, and L q-bit words $\boldsymbol{u}_i$, $1 \leq i \leq L$, encoded by the above routine.
Output: Series of decoded q-bit words, denoted by $\hat{\boldsymbol{u}}_i$, $1 \leq i \leq L$. The $(Lq-1)$-bit source data $(a_1, \ldots, a_{Lq-1})$ are found after a reshuffling of $\hat{\boldsymbol{u}}_i$.
Restoring the source data:
for $i = 1 : L$ $\hat{\boldsymbol{u}}_i = \boldsymbol{u}_i$ end
if $\mathrm{dec}(\boldsymbol{u}_1) < 2^{q-1}$
  $v = 1; c = 0;$
  while $(c < 2^{q-1})$
    $v_o = v$; $c = \mathrm{dec}(\boldsymbol{u}_v)$;
    $v = 1 + c \div w$; $\hat{\boldsymbol{u}}_v = \mathrm{bin}_q(\mathrm{dec}(\boldsymbol{u}_{v_o}) - (v-1)w)$;
  end
  $\hat{\boldsymbol{u}}_1 = \mathrm{bin}_q(c - 2^{q-1})$;
end.

The arithmetic of the algorithms is easily embodied in a look-up table. Two worked encoding examples will be helpful to understand the encoding algorithm.

Example *2:* Let $q = 4$, $L = 4$, $w = 2$ ('0000' and '0001' are forbidden words). Let the user data be '000 0001 0000 1111'. After prepending the pivot bit '1', we obtain the sequence '1000 0001 0000 1111'. Set at the start $v = 1$. We find the first inadmissible word at position $i_1 = 2$, and we replace it by the pivot word, that is, $\boldsymbol{u}_2 = \boldsymbol{u}_1$ = '1000'. Then $w_{i_1} = \text{dec}(\boldsymbol{u}_2) = 1$, so that the pivot word becomes $\boldsymbol{u}_1 = \text{bin}_4((i_1 - 1)w + w_{i_1}) = \text{bin}_4((2-1)2+1) = \text{bin}_4(3) =$ '0011'. We obtain the intermediate result '0011 1000 0000 1111'. Set $v = i_1 = 2$. The second inadmissible word is found at index position $i_2 = 3$. We now set $\boldsymbol{u}_3 = \boldsymbol{u}_v = \boldsymbol{u}_2 =$ '1000' and $\boldsymbol{u}_2 = \text{bin}_4((i_2 - 1)w + w_{i_2}) = \text{bin}_4(4+0) =$ '0100', and we obtain the final result '0011 0100 1000 1111'. ■

Example *3:* Let, as above, $q = 4$, $L = 4$, $w = 2$. Let the user data be '000 0000 0000 0000'. Without much ado, we write down the intermediate results '0010 1000 0000 0000', '0010 0100 1000 0000', and '0010 0100 0110 1000'. The sent codeword is '0010 0100 0110 1000'. In case the source word is '001 0001 0001 0001', we obtain the codeword '0011 0101 0111 1001'. ■

Unique decoding is possible if the number, L, of q-bit words that can maximally be translated, equals [16]

$$L = \left\lfloor \frac{2^{q-1}}{2^q - M} \right\rfloor. \tag{4}$$

so that the rate efficiency of the code, denoted by $R_1(M)$, is at most

$$R_1(M) = \frac{Lq - 1}{LC}. \tag{5}$$

If $2^q - M = 2^t$ is a power of two, $1 \le t \le q-1$, we simply find

$$R_1(M) = \frac{q - 2^{t+1-q}}{C}. \tag{6}$$

In case $L = 1$, when M is in the range

$$2^{q-1} < M \le \frac{3}{2}2^{q-1} - 1, \tag{7}$$

the algorithm does not improve the rate efficiency with respect to a simple look-up table. The worst case rate efficiency equals

$$R_1\left(\frac{3}{2}2^{q-1} - 1\right) \approx \frac{q-1}{q - 2 + \log_2(3)}. \tag{8}$$

We may, in some instances, improve the rate efficiency by combining r, $r \ge 1$, M-ary symbols into a single r-symbol word.

C. Combining words

The number of combined admissible r-symbol words equals M^r, so that we obtain

$$q' = \lceil r \log_2 M \rceil, \tag{9}$$

$$L' = \left\lfloor \frac{2^{q'-1}}{2^{q'} - M^r} \right\rfloor. \tag{10}$$

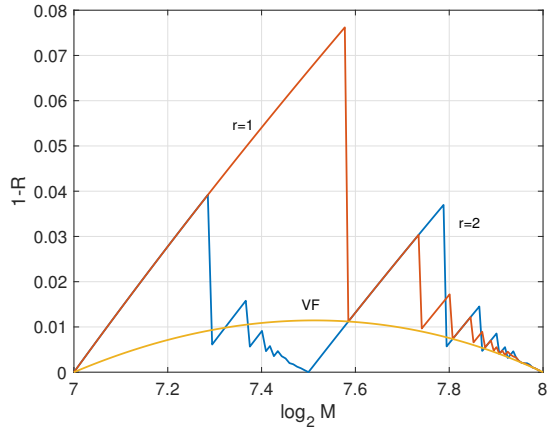

Fig. 1. Redundancy $1 - R_r(M)$ versus $C = \log_2 M$ for $r = 1$ and 2. The curve denoted by VF is discussed in Section IV.

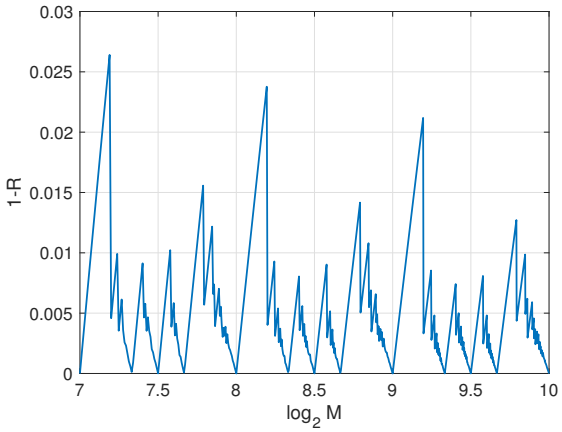

Fig. 2. Redundancy $1 - \hat{R}_3(M)$ versus $C = \log_2 M$, where $\hat{R}_3(M)$ denotes the largest of $R_1(M), R_2(M)$, and $R_3(M)$.

Let $R_r(M)$ denote the rate efficiency of the encoder, then

$$R_r(M) = \frac{L'q' - 1}{L'rC} = R_1(M^r). \tag{11}$$

Figure 1 shows the redundancy $1 - R_r(M)$ versus $C = \log_2 M$ for $r = 1$ and 2. The encoder cannot improve its rate efficiency with respect to an encoder using a simple look-up table if $L' = 1$, that is, when M is in the range $2^{q'-1} < M < \sqrt[r]{\frac{3}{2}2^{q'-1}}$, which gets smaller with increasing r.

We may further observe that not for all M we have $R_r(M) > R_1(M), r > 1$. Let $\hat{R}_m(M)$ denote the highest $R_r(M)$ achievable for a selected $r = 1, \ldots, m$, that is, $\hat{R}_m(M) = \max\{R_1(M), R_2(M), \ldots, R_m(M)\}$. Figure 2 shows the redundancy $1 - \hat{R}_3(M)$ as a function of $\log_2 M$.

D. Encoder complexity

The complexity of the encoder scales with L. If the source data are random, then the probability of t replacements, $0 \le$

TABLE I
ENCODER TABLE OF VF CODE FOR $M = 5$.

input	*output*
000	0
001	1
01	2
10	3
11	4

$t \leq L$, in the L q-bit words follows a binomial distribution. The average number of replacements, denoted by μ, is

$$\mu = L\frac{w}{2^q}. \tag{12}$$

As (4), we have $\mu \leq 1/2$. The probability that no word is altered equals $(1 - \frac{w}{2^q})^L$.

IV. VARIABLE-TO-FIXED (VF) LENGTH ENCODING

Cao and Fair have investigated the application of variable-to-fixed length codewords (VF code) for constrained systems [17, 18]. Again, let $q = \lceil \log_2 M \rceil$. In the VF scheme, the M source words may have two lengths, namely $q-1$ and q. We define $v_1 = 2^q - M$ source words of length $q-1$ and $v_2 = M - v_1 = 2M - 2^q$ source words of length q. The M source words are assigned to the M integers taken from $\mathbb{M}$.

Example *4:* Let $M = 5$, then $q = 3$, $v_1 = 2^q - M = 3$, and $v_2 = 2$. Define the five source words 000, 001, 01, 10, 11 of length 3 and 2, respectively. The $M = 5$ source words are arbitrarily assigned to the integers $0, \ldots, 4$. Table I shows a possible assignment. Let the input string be 001011000011. The encoder parses the input string into the words 001, 01, 10, 000, 11 and translates them, using Table I, into the output string 1, 2, 3, 0, and 4. ■

Assuming independent and identically distributed random input data, the *average* rate efficiency of the VL code, denoted by $R_{\rm vl}(M)$, equals

$$R_{\rm vl}(M) = \frac{1}{C}\left(\frac{q-1}{2^{q-1}}v_1 + \frac{q}{2^q}v_2\right) = \frac{1}{C}\left(q - 2 + \frac{M}{2^{q-1}}\right). \tag{13}$$

The *worst case* rate efficiency equals $\lfloor C \rfloor / C$, which is the same as that of a simple block code. Note that, see (6), for $M = 2^q - 2^t$ we have $R_{\rm vl}(M) = R_1(M)$. Figure 3 shows the redundancy $1 - R_{\rm vl}(M)$ versus $\log_2 M$. A survey plotted in Figure 1 shows the redundancy $1 - R_{\rm vl}(M)$ for the variable-to-fixed length (VF) encoding and $1 - R_r(M)$ versus $C = \log_2 M$, $r = 1$, $r = 2$, for the new method.

V. APPLICATIONS

In this section, we present applications to Pearson codes and fixed-weight codes.

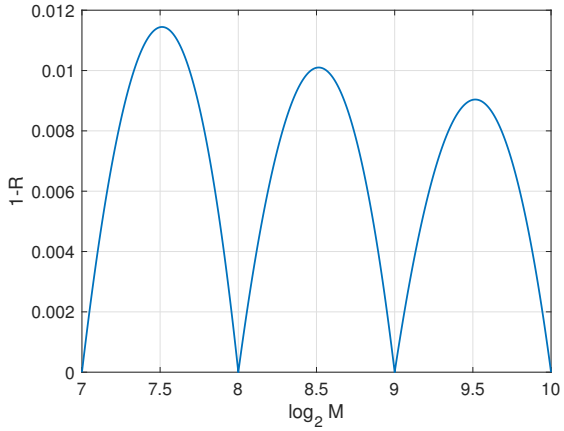

Fig. 3. Redundancy $1 - R_{\rm vl}(M)$ of the VL scheme versus $C = \log_2 M$.

A. Binary Pearson codes

Pearson codes have been advocated for channels whose gain and/or offset are unknown [19]. For binary channels, $\mathbb{Q} = \{0, 1\}$, with unknown offset, the *offset channel,* it suffices to forbid the all-0 word (or the all-1 word), and for channels with both unknown gain and offset, the *offset/gain channel,* we forbid both the all-0 and all-1 words. Let the codeword size be q, then we simply have $M = 2^q - 1$, $M' = 2^q - 2$, and $C = \log_2 M$, $C' = \log_2 M'$ for the offset and offset/gain channel, respectively. Although only one or two codewords are barred, prior art block codes face a serious loss for small q. By invoking the new coding method, we are able to improve the rate efficiency to $R_1(2^q-1) = (q-2^{-q+1})/C$ or $R'_1(2^q-2) = (q - 2^{-q+2})/C'$, see (6). For the VF code we find the same rate efficiency results, namely $R_{\rm vl}(2^q - 1) = R_1(2^q - 1)$ and $R'_{\rm vl}(2^q - 2) = R'_1(2^q - 2)$, respectively, which accords with the results presented in [17, 18, 20].

B. Fixed-weight codes

The weight of a binary codeword is the number of its symbols equal to '1'. A *balanced* code is a fixed-weight code whose codewords have equal numbers of 1's and 0's. The 6B8B [21] and 4B6B [3] codes are examples of balanced codes, where the short-hand notation mBℓB refers to codes that translate an m-bit input word into an ℓ-bit codeword. Balanced codes, such as the 6B8B and 5B10B codes, have a minimum *Hamming distance* of at least two. The 5B10B features a minimum Hamming distance 4 [22], which offers a greater noise resilience at the cost of a higher redundancy. Note that the 8B10B [2] code is not balanced as it uses codewords of weight 4, 5, and 6. The codewords with weight 4 or 6 are sent alternately for balancing the numbers of 0'1 and 1's, so that the concatenation of codewords is almost balanced [2]. The minimum Hamming distance of the 8B10B code is unity.

We have applied the new coding method to constructing balanced codes. Table II shows the rate efficiency of the new construction versus the codeword length ℓ, where $M = \binom{\ell}{\ell/2}$.

TABLE II
PERFORMANCE OF ℓ-BIT BALANCED CODES.

ℓ	$M = \binom{\ell}{\ell/2}$	$\lfloor C \rfloor / C$	$R_1(M)$	$R_{\text{vl}}(M)$
8	70	0.979	0.979	0.994
10	252	0.877	0.999	0.999
12	924	0.914	0.995	0.995
14	3432	0.937	0.993	0.994
16	12870	0.952	0.989	0.994

Except for the case $\ell = 8$, the resulting rate efficiency is close to capacity, in most cases less than half a percent. Example 5, $\ell = 10$, details the construction of a rate 255/320, 255B320B balanced code with minimum Hamming distance two, whose rate is 0.3 % lower than that of an 8B10B code with a minimum Hamming distance of unity.

Example *5:* There are $M = 252$ 10-bit balanced words. A straightforward implementation of a block code translates 7 source bits into 10 channel bits. We may improve the efficiency by combining codewords, see Example 1, but its implementation requires impracticably large look-up tables. With the new scheme, we find $q = \lceil \log_2 M \rceil = 8$, and $w = 2^8 - 252 = 4$. Then $Lq - 1 = 255$ source bits can be encoded into $L = 32 (= 2^{q-1}/w)$ 10-bit balanced words. The rate efficiency is 0.999, see Table II. The new encoding method requires data storage of 32 bytes, the execution of the encoding algorithm, and a small look-up table for translating an 8-bit wide word into a 10-bit balanced word. ■

VI. CONCLUSIONS

We have presented an encoding method for efficiently translating binary source data into a series of integer symbols in the alphabet $\{0, \ldots, M-1\}$. The series of integer symbols is translated, using a second encoder, into a series of constrained codewords. We have compared the rate efficiency of the new scheme with that of variable-to-fixed (VF) length codes. As an application example, we have presented constructions of Pearson codes and fixed-weight and balanced codes that offer a rate close to capacity. We have presented a high-rate 255B320B balanced code, where 255 source bits are translated into 32 10-bit balanced codewords, has a rate 0.1 % below capacity, and a minimum Hamming distance between the 10-bit words being two.

REFERENCES

[1] K. Balasubramanian, S. S. Agili and A. Morales, "Encoding and compensation schemes using improved pre-equalization for the 64B/66B Encoder," 2012 IEEE International Conference on Consumer Electronics (ICCE), Las Vegas, NV, pp. 361-363, 2012, doi: 10.1109/ICCE.2012.6161902.

[2] A. X. Widmer and P. A. Franaszek, "A Dc-balanced, Partitioned-Block, 8B/10B Transmission Code," *IBM J. Res. Develop.,* vol. 27, no. 5, pp. 440-451, Sept. 1983, doi: 10.1147/rd.275.0440.

[3] "IEEE standard for local and metropolitan area networks, part 15.7-2011: Short range wireless optical communication using visible light," *IEEE Std 802.15.7-2011,* pp. 1-309, Sept 2011, doi: 10.1109/IEEESTD.2011.6016195.

[4] B. H. Marcus, P. H. Siegel, and J. K. Wolf, "Finite-state Modulation Codes for Data Storage," *IEEE Journal on Selected Areas in Communications,* vol. 10, no. 1, pp. 5-37, Jan. 1992, doi: 10.1109/49.124467.

[5] P. H. Siegel, "Recording Codes for Digital Magnetic Storage," *IEEE Transactions on Magnetics,* vol. MAG-21, no. 5, pp. 1344-1349, Sept. 1985, doi: 10.1109/TMAG.1985.1063972.

[6] Z. Wang, Q. Wang, W. Huang, and Z. Xu, *V*isible Light Communications: Modulation and Signal Processing, Wiley-IEEE Press, Jan 2018.

[7] M. Blawat, K. Gaedke, I. Hutter, X. Cheng, B. Turczyk, S. Inverso, B. W. Pruitt, and G. M. Church, "Forward Error Correction for DNA Data Storage," International Conference on Computational Science (ICCS 2016), vol. 80, pp. 1011-1022, 2016, doi.org/10.1016/j.procs.2016.05.398.

[8] K. A. S. Immink and K. Cai, "Properties and Constructions of Constrained Codes for DNA-based Data Storage," *IEEE Access,* vol. 8, pp. 49523-49531, 2020, doi: 10.1109/ACCESS.2020.2980036.

[9] S. Fukuda, Y. Kojima, Y. Shimpuku, and K. Odaka, "8/10 Modulation Codes for Digital Magnetic Recording," *IEEE Transactions on Magnetics,* vol. MAG-22, no. 5, pp. 1194-1196, Sept. 1986, doi: 10.1109/TMAG.1986.1064445.

[10] V. Braun and K. A. S. Immink, "An Enumerative Coding Technique for DC-free Runlength-Limited Sequences," *IEEE Transactions on Communications,* vol. 48, no. 12, pp. 2024-2031, Dec. 2000, doi: 10.1109/26.891213.

[11] K. A. S. Immink and K. Cai, "Properties and constructions of energy-harvesting sliding-window constrained codes," *IEEE Communications Letters,* vol. 24, no. 9, pp. 1890-1893, Sept. 2020, doi: 10.1109/LCOMM.2020.2993467.

[12] K. A. S. Immink, "A New DC-free Runlength Limited Coding Method for Data Transmission and Recording," *IEEE Transactions on Consumer Electronics,* vol. CE-65, no. 4, pp. 502-505, Nov. 2019, doi: 10.1109/TCE.2019.2932795.

[13] K. A. S. Immink and K. Cai, "Spectral Shaping Codes," *IEEE Transactions on Consumer Electronics,* vol CE-67, no. 2, pp. 158-165, May 2021, 10.1109/TCE.2021.3073199.

[14] J. H. Weber, K. A. S. Immink, and S. R. Blackburn, "Pearson Codes," *IEEE Transactions on Information Theory,* vol. IT-62, no. 1, pp. 131-135, Jan. 2016, doi: 10.1109/TIT.2015.2490219.

[15] D. E. Knuth, "Positional Number Systems," *The Art of Computer Programming,* vol. 2: Semi-numerical Algorithms, 3rd ed. Reading, MA: Addison-Wesley, pp. 195-213, 1998.

[16] K. A. S. Immink, "High-Rate Maximum Runlength Constrained Coding Schemes Using Nibble Replacement," *IEEE Transactions on Information Theory,* pp. 6572-6580, vol. IT-58, no. 10, Oct. 2012, doi: 10.1109/TIT.2012.2204034.

[17] C. Cao and I. Fair, "Construction of Multi-State Capacity-Approaching Variable-Length Constrained Sequence Codes With State-Independent Decoding," *IEEE Access,* vol. 7, pp. 54746-54759, 2019, doi: 10.1109/ACCESS.2019.2913339.

[18] C. Cao and I. Fair, "Capacity-Approaching Variable-Length Pearson Codes," *IEEE Communications Letters*, vol. 22, no. 7, pp. 1310-1313, July 2018, doi: 10.1109/LCOMM.2018.2829706.

[19] K. A. S. Immink and J. H. Weber, "Minimum Pearson Distance Detection for Multi-Level Channels with Gain and/or Offset Mismatch," *IEEE Transactions on Information Theory,* vol. IT-60, no. 10, pp. 5966-5974, Oct. 2014, doi: 10.1109/TIT.2014.2342744.

[20] J. H. Weber, T. G. Swart, and K. A. S. Immink, "Simple Systematic Pearson Coding," IEEE International Symposium on Information Theory, Barcelona, Spain, pp. 385-389, July 2016, doi: 10.1109/ISIT.2016.7541326.

[21] A. X. Widmer, "Dc-balanced 6B/8B Transmission Codes with Local Parity," US Patent 6,876,315, April 2005.

[22] V. A. Reguera, "New RLL code with improved error performance for visible light communication," arXiv preprint arXiv:1910.10079, 2019.