

Very Efficient Balanced Codes

Kees A. Schouhamer Immink and Jos H. Weber

Abstract—The prior art construction of sets of balanced codewords by Knuth is attractive for its simplicity and absence of look-up tables, but the redundancy of the balanced codes generated by Knuth’s algorithm falls a factor of two short with respect to the minimum required. We present a new construction, which is simple, does not use look-up tables, and is less redundant than Knuth’s construction. In the new construction, the user word is modified in the same way as in Knuth’s construction, that is by inverting a segment of user symbols. The prefix that indicates which segment has been inverted, however, is encoded in a different, more efficient, way.

Index Terms—Magnetic recording, optical recording, channel capacity, constrained code, dc-free code, balanced code.

I. INTRODUCTION

SETS of bipolar codewords that have equal numbers of 1 ’s and -1 ’s are usually called *balanced codes*. Balanced codes have been widely used in storage and communication channels. A survey of properties and methods for constructing balanced codes can be found in [1]. There is a trend towards high rate codes, which are made possible by codes with longer codewords. The implementation of such a code is not a simple task as look-up tables for translating user words into channel words and *vice versa* are impractically large. Knuth published a simple algorithm for constructing balanced codes [2], which is very well suited for use with long codewords, since look-up tables are absent. Modifications and improvements of the generic scheme are discussed by Alon *et al.* [3], Al-Bassam & Bose [4], Tallini, Capocelli & Bose [5], and Weber & Immink [6]. The redundancy of a balanced code generated by Knuth’s algorithm falls a factor of two short with respect to the minimum required, i.e. the redundancy of a code that uses all balanced codewords of a given length [1]. An attempt by Weber & Immink [6] to compress the fixed length prefix to a variable length prefix with less redundancy was futile.

In this paper, we will study simple balanced code designs that require minimum redundancy, while, as in Knuth’s construction, the balanced codeword is obtained by a simple, algorithmic modification of the user word, and a prefix is added that carries sufficient information for the recipient to uniquely retrieve the user word. The way the user word is modified in the new construction is the same as in Knuth’s construction, that is by inverting a segment of user symbols. The prefix, however, is encoded and decoded in a different, more efficient,

way. We start, in Section II, with a brief description of the prior art code construction by Knuth, followed, in Section III, by a description of the new construction. In Section IV we will compute some statistics, which will enable us, in V, to compute the redundancy of the new code construction. In Section VI, we will present some details regarding the implementation of the new algorithm. Section VII concludes the paper.

II. KNUTH’S CODE CONSTRUCTION

The conventional Knuth algorithm runs as follows. The user data is arranged as a bipolar m -tuple $\mathbf{u} = (u_1, \dots, u_m)$, $u_i \in \{-1, 1\}$, m even. (Knuth also presented code constructions for odd m , but they will not be discussed here.) We define for $1 \leq j \leq m$, the bipolar m -tuple $\mathbf{u}^j = (-u_1, -u_2, \dots, -u_j, u_{j+1}, u_{j+2}, \dots, u_m)$. Knuth showed that for any user data \mathbf{u} an index j can be found such that the codeword \mathbf{u}^j is balanced, that is

$$-\sum_{i=1}^j u_i + \sum_{i=j+1}^m u_i = 0.$$

The index j is not necessarily unique as in general there are more positions where balance can be obtained [6]. Let the *smallest* index j , where \mathbf{u}^j is balanced, be denoted by $I(\mathbf{u})$. In other words, $I(\mathbf{u})$ is the smallest index, $1 \leq i \leq m$, for which \mathbf{u}^i is balanced. The balanced codeword $\mathbf{x} = \mathbf{u}^{I(\mathbf{u})}$ plus a prefix, which suitably represents the index $I(\mathbf{u})$, is transmitted. The receiver receives the codeword \mathbf{x} plus prefix $I(\mathbf{u})$, and can thus uniquely undo the encoding step by forming $\mathbf{u} = \mathbf{x}^{I(\mathbf{u})}$. For an efficient code, the redundant prefix should be as small as possible. Knuth showed that in his best construction the redundancy p is roughly equal to [2]

$$\log_2 m, \quad m \gg 1. \quad (1)$$

The redundancy of a full set of balanced codewords of length m , H_0 , equals

$$H_0 = m - \log_2 \binom{m}{m/2}, \quad (2)$$

and can be approximated by [2]

$$H_0 \approx \frac{1}{2} \log_2 m + 0.326, \quad m \gg 1. \quad (3)$$

We notice that, for large values of the codeword length, m , the redundancy of Knuth-based codes is twice as high as that of codes that uses ‘full’ sets of balanced codewords. It has been a continuing desideratum in data communication and storage systems to increase the capacity by using more efficient coding methods. In the next section, we will present the new coding technique.

Manuscript received 12 January 2009; revised 17 July 2009. This project was supported by grant *Theory and Practice of Coding and Cryptography*, Award Number: NRF-CRP2-2007-03

Kees A. Schouhamer Immink is with Turing Machines Inc., Willemskade 15b-d, 3016 DK Rotterdam, The Netherlands, National Technological University of Singapore, Singapore (e-mail: immink@turing-machines.com).

Jos H. Weber is with TU Delft, IRCTR/CWPC, Mekelweg 4, 2628 CD Delft The Netherlands, (e-mail: J.H.Weber@ewi.tudelft.nl).

Digital Object Identifier 10.1109/JSAC.2010.1002xx.

III. NEW PREFIX CODING SCHEME FOR BALANCING CODEWORDS

Let $\mathbf{x} = \mathbf{u}^{I(\mathbf{u})}$ be a codeword balanced by Knuth's method described above. Clearly, we have $\mathbf{u} = \mathbf{x}^{I(\mathbf{u})} \in \{\mathbf{x}^1, \dots, \mathbf{x}^m\}$, and the index received makes it possible to uniquely single out the right member from the m possible ones. The new encoder is based upon the observation that not all m members of the set $\{\mathbf{x}^1, \dots, \mathbf{x}^m\}$ can be legally associated with \mathbf{x} , since, by definition, Knuth's encoder takes the smallest index for balancing a user word. An m -tuple \mathbf{x}^j with $j > I(\mathbf{x}^j)$, is not a *bona fide* word under Knuth's rules. We now define the set of user words legally associated with \mathbf{x} by $\sigma_{\mathbf{x}} = \{\mathbf{x}^j : j = I(\mathbf{x}^j)\}$. The cardinality of $\sigma_{\mathbf{x}}$ will be denoted by $d(\mathbf{x}) = |\sigma_{\mathbf{x}}|$.

Example: Let $m = 6$. Then it can be verified that $\sigma_{'000111'}$ = $\{'100111', '110111', '111111', '111000'\}$, where a '0' denotes the symbol value '-1'. In a similar way, $\sigma_{'010101'}$ = $\{'110101', '100101'\}$. Hence $d('000111')$ = 4 and $d('010101')$ = 2.

An efficient encoder will transmit the balanced word $\mathbf{x} = \mathbf{u}^{I(\mathbf{u})}$ plus an index that resolves the ambiguity about which word in $\sigma_{\mathbf{x}}$ is meant. The average number of bits required to represent the index depends on the way the code construction is implemented. For a fixed-length-prefix construction it depends on the maximum size of $\sigma_{\mathbf{x}}$, while in a variable-length-prefix construction it will depend on the average size of $\sigma_{\mathbf{x}}$. Before formulating the key theorem, we offer some definitions. Let z_k be the *running sum* of the first k , $k \leq m$, symbols of \mathbf{x} , or

$$z_k = \sum_{i=1}^k x_i,$$

and let $z_{\max} = \max\{z_k\}$ and $z_{\min} = \min\{z_k\}$.

Theorem 1:

$$d(\mathbf{x}) = z_{\max} - z_{\min} + 1.$$

Proof. We have $\mathbf{x}^i \notin \sigma_{\mathbf{x}}$ if there is a j , where $(\mathbf{x}^i)^j$, $1 \leq j < i \leq m$, is balanced. Since $(\mathbf{x}^i)^j = (x_1, \dots, x_j, -x_{j+1}, \dots, -x_i, x_{i+1}, \dots, x_m)$ we conclude that the sum of the elements of $(\mathbf{x}^i)^j$ equals

$$\sum_{k=1}^m x_k - 2 \sum_{k=j+1}^i x_k.$$

Then, as $\sum_{k=1}^m x_k = 0$, we notice that $\mathbf{x}^i \notin \sigma_{\mathbf{x}}$ if there is a j , $1 \leq j < i$, such that

$$\sum_{k=j+1}^i x_k = 0.$$

Or, in other words, $\mathbf{x}^i \notin \sigma_{\mathbf{x}}$ if there is a j , $1 \leq j < i$ such that $z_i = z_j$. Let Z denote the set of sum values z_i , $1 \leq i \leq m$, and let $V(Z)$ denote the number of distinct values of Z . Then it is immediate that $d(\mathbf{x}) = V(Z)$. Since all possible values of z_i between z_{\min} and z_{\max} are in Z , we conclude that

$$d(\mathbf{x}) = V(Z) = z_{\max} - z_{\min} + 1.$$

This concludes the proof. \blacksquare

The following Theorem gives a lower and upper bound on the size $d(\mathbf{x})$.

Theorem 2:

$$2 \leq d(\mathbf{x}) \leq \frac{m}{2} + 1.$$

Proof. Clearly, $z_1 = x_1 \neq 0$ and $z_m = 0$. Thus $d(\mathbf{x}) = z_{\max} - z_{\min} + 1 \geq 2$. We now proceed with the upper bound $d(\mathbf{x}) \leq \frac{m}{2} + 1$. Note that, since $z_0 = z_m = 0$, and $z_i = z_{i-1} \pm 1$ for all i , all values in Z occur at least twice with the exception of a single maximum z_{\max} and/or a single minimum z_{\min} . Thus

$$d(\mathbf{x}) \leq 2 + \frac{m-2}{2} = \frac{m}{2} + 1.$$

This concludes the proof. \blacksquare

The conventional Knuth scheme, which requires a prefix of length $\log_2(m)$ bits, is less efficient than a new scheme based on Theorem 2, which shows that the maximum number of bits required to represent the index equals $\log_2(m/2 + 1)$. The average number of bits required to represent the index will be computed in the next section.

IV. COMPUTATION OF THE DISTRIBUTION OF $d(\mathbf{x})$

In this section, we will compute the distribution of $d(\mathbf{x})$, so that we can, in Section V, compute the redundancy of the new schemes. Let $P(u, m)$ denote the number of binary balanced words \mathbf{x} of length m with $d(\mathbf{x}) = u$. By definition and invoking Theorem 2, we have

$$\sum_{u=2}^{m/2+1} P(u, m) = \binom{m}{m/2}.$$

The computation of the distribution of $d(\mathbf{x})$ is related to the problem of computing the number of sequences (random walks) whose running sum remains within given limits, a problem first studied by Chien [7]. Chien studied bipolar sequences $\{x_i\}$, $x_i \in \{-1, 1\}$, where the running sum $z_i = z_{i-1} + x_i$, for any i , remains within the limits N_1 and N_2 , where N_1 and N_2 are two (finite) constants, $N_2 > N_1$. The range of sum values a sequence may assume, denoted by

$$N = N_2 - N_1 + 1, \quad (4)$$

is often called the *digital sum variation*. Taking z_i at any instant i as the state of the stream $\{x_i\}$, then the bounds to z_i define a set of N allowable states. For the N -state source, an $N \times N$ connection matrix, D_N , is defined by $D_N(i, j) = 1$ if a transition from state σ_i to state σ_j is allowable and $D_N(i, j) = 0$ otherwise. The connection matrix D_N for the channel having a bound to the number of assumed sum values is given by

$$\begin{aligned} D_N(i+1, i) &= D_N(i, i+1) = 1, \quad i = 1, 2, \dots, N-1, \\ D_N(i, j) &= 0, \quad \text{otherwise.} \end{aligned} \quad (5)$$

The (i, j) -th entry of the m -th power of D_N will be denoted by $D_N^m(i, j)$. The following Theorem will be helpful in computing $P(u, m)$.

Theorem 3: The number of balanced words \mathbf{x} of length m with $d(\mathbf{x}) = u$, $P(u, m)$, $2 \leq u \leq m/2 + 1$, is given by

$$P(u, m) = \sum_{i=1}^u D_u^m(i, i) - 2 \sum_{i=1}^{u-1} D_{u-1}^m(i, i) + \sum_{i=1}^{u-2} D_{u-2}^m(i, i), 2 \leq u \leq \frac{m}{2} + 1.$$

Proof. In order to calculate $P(u, m)$, we must count the number of balanced sequences \mathbf{x} of length m , whose running sum span equals u . The matrix entries $D_N^m(i, i)$ give the number of balanced sequences of length m , whose running sum span is *at most* N that start and end with a given sum value i . Thus the count $D_N^m(i, i)$ includes words \mathbf{x} with $d(\mathbf{x}) < N$. We may resolve this difficulty by observing that a balanced word \mathbf{x} with $d(\mathbf{x}) = N$ has a unique starting (and ending) state. Namely, assume a word \mathbf{x} with the property $d(\mathbf{x}) = N$, and let

$$z_k = z_0 + \sum_{i=1}^k x_i,$$

where $1 \leq z_0 \leq N$ denotes the initial value of the running sum. Then, by definition, $\max\{z_i\} - \min\{z_i\} + 1 = N$. The limiting values z_{\max} and z_{\min} are by definition the maximum and minimum sum values allowed within the N -state machine. Other values of z_0 are not allowed as they will lead to too high or too low a value of the running sum. We conclude that there is a unique starting (state) value z_0 for a sequence having the maximum running sum span N . Similarly, a word \mathbf{x} with $d(\mathbf{x}) = N - 1$ may have two possible starting states, and a word \mathbf{x} with $d(\mathbf{x}) = N - 2$ has three possible starting states, etc. As a result, we find

$$\begin{aligned} \sum_{i=1}^u D_u^m(i, i) &= P(u, m) + 2P(u-1, m) + \\ &+ 3P(u-2, m) + 4P(u-3, m) + \dots \\ &= \sum_{k=0}^{u-2} (k+1)P(u-k, m). \end{aligned}$$

Then, after a simple manipulation, we find

$$P(u, m) = \sum_{i=1}^u D_u^m(i, i) - 2 \sum_{i=1}^{u-1} D_{u-1}^m(i, i) + \sum_{i=1}^{u-2} D_{u-2}^m(i, i), 2 \leq u \leq \frac{m}{2} + 1. \quad (6)$$

This proves the theorem. \blacksquare

A useful property to compute powers of D_N was derived by Salkuyeh [8], namely

$$D_N^m(i, j) = \frac{2}{N+1} \sum_{k=1}^N \lambda_k^m \sin \frac{ik\pi}{N+1} \sin \frac{jk\pi}{N+1}, \quad (7)$$

where

$$\lambda_i = 2 \cos \frac{\pi i}{N+1}, 1 \leq i \leq N$$

TABLE I
 $P(u, m)$ VERSUS u .

u	$P(u, m)$
2	2
3	$2(2^{m/2} - 2)$
$m/2$	$m(m-4), m > 4$
$m/2 + 1$	m

are the eigenvalues of D_N [1]. The number $\sum_{i=1}^N D_N^m(i, i)$ can be calculated by invoking relation (7). After rearranging some terms, we find

$$\begin{aligned} \sum_{i=1}^N D_N^m(i, i) &= \frac{2}{N+1} \sum_{k=1}^N \lambda_k^m \sum_{i=1}^N \sin^2 \frac{ik\pi}{N+1} \\ &= \sum_{i=1}^N \lambda_i^m = 2^m \sum_{i=1}^N \cos^m \frac{\pi i}{N+1}. \quad (8) \end{aligned}$$

With the above relations it is now straightforward to compute $P(u, m)$. For special values of u , we could derive simple relations that offer more insight. The first two cases were discussed previously.

- There are two codewords that achieve the minimum bound $d(\mathbf{x}) = 2$, namely $\mathbf{x} = (+1, -1, +1, -1, \dots)$ and its inverse.
- There are m codewords that achieve the upper bound, $d(\mathbf{x}) = m/2 + 1$, namely the codewords starting with the maximum runlength of $m/2 + 1$'s followed by $m/2 - 1$'s, and the $m - 1$ circular shifts of that codeword.
- There are $2(2^{m/2} - 2)$ codewords \mathbf{x} with $d(\mathbf{x}) = 3$, namely the $2^{m/2}$ codewords formed of combinations of the 2-bit words $(+1, -1)$ and $(-1, +1)$ minus the two codewords with $d(\mathbf{x}) = 2$ plus the one-bit circular shifts of those codewords.
- There are $m(m-4)$ codewords \mathbf{x} with $d(\mathbf{x}) = m/2$, $m > 4$, namely the $m/2 - 2$ codewords starting with a runlength of $m/2 - 1$ '1's followed by i '1's, a '1', and $m/2 - i$ '1's, $1 \leq i \leq m/2 - 2$, their inverse, and the $m - 1$ circular shifts of those codewords.

A survey of the above findings is shown in Table I.

V. PERFORMANCE COMPUTATIONS

We first compute the average number of bits, H , required to represent the index. The quantity, H , sets a theoretical limit as it is not assumed that the prefix is balanced or has an integer number of bits. There are $d(\mathbf{x})$ different user words that are transformed into the balanced word \mathbf{x} , so that we conclude

$$\sum_{u=2}^{m/2+1} uP(u, m) = 2^m. \quad (9)$$

The average number of bits, H , required to represent the index is given by

$$H = 2^{-m} \sum_{u=2}^{m/2+1} uP(u, m) \log_2 u. \quad (10)$$

Results of computations of the average prefix length, H , versus user word length m are listed in Table II. As a reference

TABLE II
AVERAGE PREFIX LENGTH, H , AND MINIMUM REDUNDANCY,
 $H_0 = m - \log_2 \binom{m}{m/2}$, VERSUS USER WORD LENGTH m .

m	H	H_0
64	3.3641	3.3314
128	3.8616	3.8286
256	4.3603	4.3272
512	4.8597	4.8265
1024	5.3594	5.3261
2048	5.8592	5.8259
4096	6.3591	6.3258
8192	6.8591	6.8258

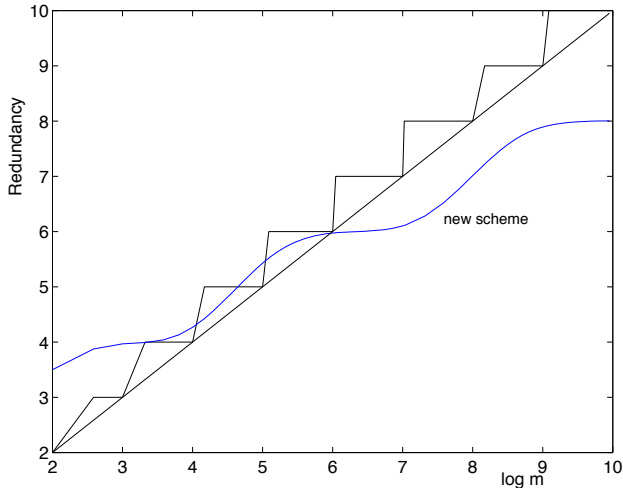


Fig. 1. Average prefix length as a function of $\log_2 m$ of the VL prefix Knuth scheme with balanced prefix. As a reference we plotted the minimum redundancy of Knuth's construction $\log_2(m)$ and $\lceil \log_2(m) \rceil$.

we listed the base-line redundancy of full sets of balanced codewords, H_0 . The difference between the average prefix length H and H_0 is less than 1 percent. The redundancy of the variable-length (VL) balanced code, H , as shown in Table II, is a theoretical minimum. As in Knuth's prior art construction, the VL prefix should be balanced (or should compensate the unbalance of the codeword). To that end, for every integer p , $p > 0$, we define the integer function $B(p)$ as the smallest even integer q such that

$$\binom{q}{q/2} \geq p.$$

Then, assuming that the VL index is mapped onto a balanced prefix, we find with a slight modification of (10) that

$$\hat{H} = 2^{-m} \sum_{u=2}^{m/2+1} uP(u, m)B(u), \quad (11)$$

where \hat{H} denotes the redundancy of the new construction having balanced prefixes. Figure 1 shows results of computations.

As a reference we plotted the curves $\log_2(m)$ and $\lceil \log_2(m) \rceil$, which show the minimum redundancy and that of integer valued redundancy of Knuth's construction. We may observe that for $m < 64$ the redundancy of the fixed-prefix Knuth scheme and that of the VL scheme do not significantly differ. For $m > 64$, we notice that the average redundancy of

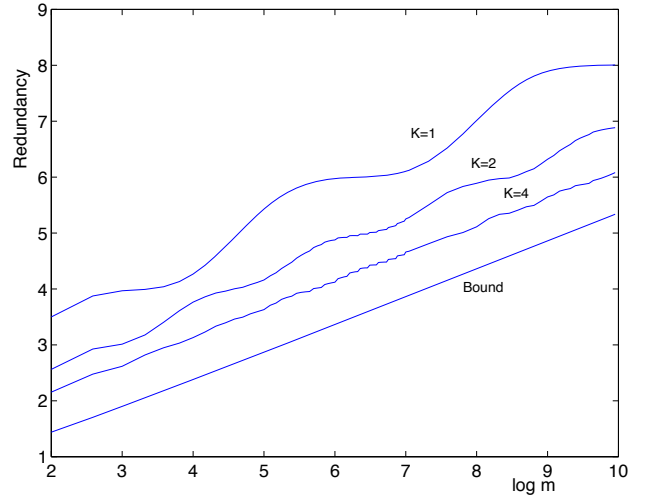


Fig. 2. Average prefix length as a function of $\log_2 m$ of the VL prefix Knuth scheme combining $K = 1, 2$, or 4 m -bit codewords; all schemes have a balanced prefix.

the new scheme is less than that of the classic Knuth scheme. We may approach the theoretical minimum by combining a plurality of codewords, say K . Each of the K m -bit user words is balanced as discussed above, and the K prefixes are combined into one 'super prefix'. The super prefix is balanced by a look-up table or in case the super prefix is too long for table look-up, we apply Knuth's method. Figure 2 shows some results of computations, where it is assumed that the prefixes of $K = 2$ and $K = 4$ words are combined. As a reference we plotted the curve of minimum redundancy as defined by (10). We note that the curve showing the average redundancy of $K = 4$ combined prefixes is only one bit away from the bound. We will now take a look at the implementation of both the encoder and decoder that exploits the findings of Theorem 2.

VI. IMPLEMENTATION ISSUES

The two coding schemes, which are based on Theorem 2, may use a) a fixed length or b) a variable length (VL) prefix. In the first scheme, the prefix length is fixed as in the conventional scheme. Then the prefix must be able to uniquely encode at most $m/2 + 1$ indices requiring less than $\log_2 m$ bits, so that it is less redundant than the conventional method. In the second scheme, where the prefix length depends on the user data, the prefix length varies between 1 and $\log_2(m/2 + 1)$ bits. On the average, the VL coding scheme will be more efficient than the first scheme. We will first describe the implementation of the encoder and decoder.

Encoder description: Assume the user data \mathbf{u} enters the encoder. The encoder computes, as in the classic Knuth method, the balancing position $I(\mathbf{u})$, and transmits the balanced word $\mathbf{x} = \mathbf{u}^{I(\mathbf{u})}$. The computation of the prefix is more involved. To that end, we first specify an order relation on $\sigma_{\mathbf{x}}$. Then we compute the rank, $I_{\mathbf{u}}$, $0 \leq I_{\mathbf{u}} \leq d(\mathbf{x}) - 1$, of \mathbf{u} in the ordered set $\sigma_{\mathbf{x}}$, and uniquely translate the rank $I_{\mathbf{u}}$ into a (preferably) balanced prefix. In a scheme with a fixed prefix

length, the prefix must accommodate in the worst case $m/2+1$ values of $I_{\mathbf{u}}$. For the scheme with variable prefix length, the prefix length depends on \mathbf{x} , and must accommodate the $d(\mathbf{x})$ possible values of $I_{\mathbf{u}}$. The (balanced) m -tuple \mathbf{x} and the prefix are transmitted to the receiver.

Decoder description: The decoder receives the codeword $\mathbf{x} = \mathbf{u}^{I(\mathbf{u})}$ and a suitable representation of $I_{\mathbf{u}}$. In a scheme with fixed prefix length, the decoder can identify the prefix $I_{\mathbf{u}}$. Then, the decoder generates the ordered set $\sigma_{\mathbf{x}}$, and retrieves from $\sigma_{\mathbf{x}}$ the member with rank $I_{\mathbf{u}}$, and outputs that member. In a VL scheme, the decoder first receives \mathbf{x} , and computes $d(\mathbf{x})$. The value of $d(\mathbf{x})$ identifies the prefix length. Then we decode the index $I_{\mathbf{u}}$, generate the ordered set $\sigma_{\mathbf{x}}$, and retrieve the member of $\sigma_{\mathbf{x}}$ with index $I_{\mathbf{u}}$. Note that in the VL scheme, the 'prefix' must follow the codeword, and normal people would call it therefore a 'suffix'. For heritage reasons, we will use Knuth's term 'prefix', while it should be appreciated that in this context, the 'prefix' will be following the codeword.

Essential parts of the encoding and decoding operation are the computation of $d(\mathbf{x})$, the generation of the ordered set $\sigma_{\mathbf{x}}$, and the computation of the index $I_{\mathbf{u}}$. The complexity of a straightforward algorithm for computing $d(\mathbf{x})$ grows quadratically with increasing codeword length m . We will discuss some more efficient methods. Let $\mathbf{x} = \mathbf{u}^{I(\mathbf{u})}$ be a balanced word. The complexity of the following algorithm grows linearly with m . Define the set of running sum values as $Z = \{z_k\}, \forall k$, and define the binary vector $\mathbf{v} = (v_{-m/2}, \dots, v_{m/2})$ as $v_i = 1, i \in Z$ and $v_i = 0, i \notin Z$. As $d(\mathbf{x})$ equals the number of different values that z_i assumes, which equals the weight of \mathbf{v} , we find $d(\mathbf{x}) = \sum v_i$. The following code implements the above in MatLab notation:

Algorithm 1

Input: User input word $x(i), 1 \leq i \leq m$.

Output: $d(\mathbf{x})$.

$z(1)=x(1); v(z(1)+m/2+1)=1;$

for $i=2:m; z(i)=z(i-1)+x(i); v(z(i)+m/2+1)=1; end;$

$dx=sum(v);$

where $z(i)$ denotes the running sum, $x(i) \in \{-1, +1\}$ denotes the entries of codeword \mathbf{x} , and $v(i)$ are the entries of a binary vector counting the occurrence of the running sum values $z(i) + m/2 + 1$ ¹. The sum of the entries of \mathbf{v} equals $d(\mathbf{x})$. We now define the ordering of the members of $\sigma_{\mathbf{x}}$. Let \mathbf{x}^i and \mathbf{x}^j be elements of $\sigma_{\mathbf{x}}$. We call \mathbf{x}^i less than \mathbf{x}^j , in short $\mathbf{x}^i < \mathbf{x}^j$, if $i < j$. The rank of $\mathbf{v} \in \sigma_{\mathbf{x}}$, denoted by $I_{\mathbf{v}}$, is defined to be the position of \mathbf{v} in the ordered list of members of $\sigma_{\mathbf{x}}$, i.e., $I_{\mathbf{v}}$ is the number of all \mathbf{y} in $\sigma_{\mathbf{x}}$ with $\mathbf{y} < \mathbf{v}$. The following MatLab-algorithm finds the rank, $I_{\mathbf{u}}$, of the user word $\mathbf{u} = \mathbf{x}^{I(\mathbf{u})}$ in the ordered set $\sigma_{\mathbf{x}}$ by counting the words less than $\mathbf{x}^{I(\mathbf{u})}$.

Algorithm 2

Input: $I(\mathbf{u}), v(i), z(i), 1 \leq i \leq m$, as defined in Algorithm 1.

Output: $I_{\mathbf{u}} = p$.

$p=0;$

for $i=1:I(\mathbf{u})-1; if v(z(i)+m/2+1)=1 then p=p+1; v(z(i)+m/2+1)=0; end;end;$

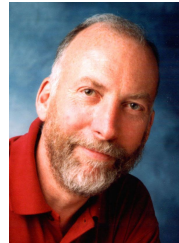
After execution of the routine, we find $I_{\mathbf{u}} = p$.

VII. CONCLUSIONS

We have presented a new method for constructing sets of balanced bipolar codewords. The new construction presented is attractive as it does not use look-up tables and is less redundant than Knuth's prior art construction. We have presented simple algorithms for computing the prefix, encoding, and decoding. We have analyzed the distribution of the lengths of the prefix length, and determined the average efficiency of the new construction.

REFERENCES

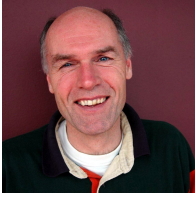
- [1] K.A.S. Immink, *Codes for Mass Data Storage Systems*, Second Edition, ISBN 90-74249-27-2, Shannon Foundation Publishers, Eindhoven, Netherlands, 2004.
- [2] D.E. Knuth, 'Efficient Balanced Codes', *IEEE Trans. Inform. Theory*, vol. IT-32, no. 1, pp. 51-53, Jan. 1986.
- [3] N. Alon, E.E. Bergmann, D. Coppersmith, and A.M. Odlyzko, 'Balancing Sets of Vectors', *IEEE Trans. Inform. Theory*, vol. IT-34, no. 1, pp. 128-130, Jan. 1988.
- [4] S. Al-Bassam and B. Bose, 'On Balanced Codes', *IEEE Trans. Inform. Theory*, vol. IT-36, no. 2, pp. 406-408, March 1990.
- [5] L.G. Tallini, R.M. Capocelli, and B. Bose, 'Design of some New Balanced Codes', *IEEE Trans. Inform. Theory*, vol. IT-42, pp. 790-802, May 1996.
- [6] J.H. Weber and K.A.S. Immink, 'Knuth's Balancing of Codewords Revisited', IEEE International Symposium on Information Theory, ISIT2008, pp. 1567-1571, Toronto, 6-11 July 2008.
- [7] T.M. Chien, 'Upper Bound on the Efficiency of De-constrained Codes', *Bell Syst. Tech. J.*, vol. 49, pp. 2267-2287, Nov. 1970.
- [8] D.K. Salkuyeh, 'Positive Integer Powers of the Tri-diagonal Toeplitz Matrices', *International Mathematical Forum*, 1, no. 22, pp. 1061 - 1065, 2006



Kees Schouhamer Immink received his PhD degree from the Eindhoven University of Technology. He founded and was named president of Turing Machines Inc. in 1998. He is, since 1994, an adjunct professor at the Institute for Experimental Mathematics, Essen University, Germany, and is affiliated with the Nanyang Technological University of Singapore. Immink designed coding techniques of a wealth of digital audio and video recording products, such as Compact Disc, CD-ROM, CD-Video, Digital Compact Cassette system, DCC, Digital Versatile

Disc, DVD, Video Disc Recorder, and Blu-ray Disc. He received a Knighthood in 2000, a personal Emmy award in 2004, the 1996 IEEE Masaru Ibuka Consumer Electronics Award, the 1998 IEEE Edison Medal, 1999 AES Gold and Silver Medals, and the 2004 SMPTE Progress Medal. He was named a fellow of the IEEE, AES, and SMPTE, and was inducted into the Consumer Electronics Hall of Fame, and elected into the Royal Netherlands Academy of Sciences and the US National Academy of Engineering. He served the profession as President of the Audio Engineering Society inc., New York, in 2003.

¹The term $m/2 + 1$ is added since MatLab does not allow non-positive array indices.



Jos H. Weber (S'87-M'90-SM'00) was born in Schiedam, The Netherlands, in 1961. He received the M.Sc. (in mathematics, with honors), Ph.D., and MBT (Master of Business Telecommunications) degrees from Delft University of Technology, Delft, The Netherlands, in 1985, 1989, and 1996, respectively.

Since 1985 he has been with the Faculty of Electrical Engineering, Mathematics, and Computer Science of Delft University of Technology. Cur-

rently, he is an associate professor at the Wireless and Mobile Communications Group. He is the chairman of the WIC (Werkgemeenschap voor Informatie- en Communicatietheorie in the Benelux) and the secretary of the IEEE Benelux Chapter on Information Theory. He was a Visiting Researcher at the University of California at Davis, USA, the University of Johannesburg, South Africa, and the Tokyo Institute of Technology, Japan. His main research interests are in the areas of channel and network coding.