On the Improvement of the Knuth's Redundancy Algorithm for Balancing Codes

Ebenezer Esenogho, Elie N. Mambou, and Hendrick C. Ferreira
Center for Telecommunication
Dept. of Electrical and Electronic Engineering Science, University of Johannesburg
P. O. Box 524, Auckland Park, 2006, South Africa
Email: {ebenezere, emambou, hcferreira}@uj.ac.za.

Abstract—A simple scheme was proposed by Knuth to generate balanced codewords from a random binary information sequence. However, this method presents a redundancy which is twice as that of the full sets of balanced codewords, that is the minimal achievable redundancy. The gap between the Knuth's algorithm generated redundancy and the minimal one is significantly considerable and can be reduced. This paper attempts to achieve this goal through a method based on information sequence candidates.

Index Terms—Balanced code, inversion point, redundancy, running digital sum (RDS), running digital sum from left (RDSL), running digital sum from right (RDSR), information sequence candidates.

I. Introduction

A binary codeword of length k is said to be balanced if the number of zeros and ones within that sequence equals $k/2$, for even k. Balanced codes are very useful for digital recording of data on optical and magnetic storage disks. They can also be used to correct or detect errors within channels.

Donald Knuth proposed a simple and efficient scheme to generate balanced codewords [1]. This approach stipulates that any binary unbalanced codeword, $\boldsymbol{x}$ of length k can always be encoded into a balanced one denoted as $\boldsymbol{x'}$, by inverting the first e bits of $\boldsymbol{x}$ where $1 \leq e \leq k$. The index e is encoded as a prefix, $\boldsymbol{p}$ that is appended to $\boldsymbol{x}'$ and send through a channel. At the receiver side, the decoder receives the codeword $\boldsymbol{px}'$, read off first the prefix and then, is able to recover the original information sequence $\boldsymbol{x}$ by inverting back the e first bits of $\boldsymbol{x}'$. This algorithm is very suitable for long sequences as it does make use of any lookup tables either at the encoder or the decoder.

The redundancy of Knuth's algorithm $\boldsymbol{p}$, is approximately evaluated as

$$p = log_2 k \; for \; m \ll 1 \tag{1}$$

Since then, numerous works were published to reduce the redundancy presented in (1).

In [2], an attempt to improve Knuth's balancing algorithm was presented based on the distribution of the transmitted prefix index. The basic Knuth scheme uses the first balanced point at position e to encode it as the prefix, therefore the encoder is set to choose smaller values for the position index. It has been shown that the distribution of the index for equiprobable information sequences, is not uniform and presents a redundancy of slightly less than (1); this scheme uses a variable length prefix of the chosen index which only makes a minor improvement on the Knuth's algorithm redundancy.

A major contribution in reducing the Knuth's algorithm redundancy was shown by Immink and Weber in [3]. This new scheme does not make use of look-up tables and presents a very efficient encoding of the index prefix; this requires at most $log_2(\,k/2+1\,)$ bits to represent the index. Furthermore, the distribution of the prefix length was discussed as well as the average efficiency of this construction.

Another attempt to reduce the Knuth algorithm redundancy was presented in [4]. This new method is called bit recycling for Knuth's algorithm (BRKA); since there is a high probability to have more than one balance point given an information sequence, this scheme uses the multiplicity of encodings to close the gap between the lower bound redundancy and the Knuth's one.

In this paper, we describe some tools to find all possible inversion points within a sequence. Furthermore, we will present an efficient and simple method to significantly improve Knuth's algorithm redundancy; every random binary information sequence is associated to a unique balanced codeword by following Knuth's scheme.

The rest of this paper is organized as follows: we will be exploring ways of finding balanced points in Section II. And then, encoding method description based on information sequence candidates is presented in Section III. Section IV describe the decoding methodology, while Section V gave a study of sparseness. Section VI present some performance analysis as well as discussions. Finally, the paper is concluded in Section VII.

II. Finding Inversion Points

There are various ways of determining inversion points given a random binary sequence, $\boldsymbol{x}$ of length k.

A. *Exhaustive Search*

This is done by following the Knuth's algorithm, that is inverting bits sequentially from the first index on the

Manuscript received April 30, 2019; revised November 8, 2019.
doi:10.12720/jcm.14.12.1147-1154

left till the last one and record all the balanced codewords. However, inversion points from left direction might be different from those from the right. Given that $\boldsymbol{x} = (x_1, \dots, x_k)$ with $x_i \in \{0,1\}$; $\boldsymbol{x}$ is referred to as balanced sequence if and only if $\sum_{i=1}^{k} x_i = k/2$ (for even k). Let us denote by e, the least index at which the sequence is balanced and $\boldsymbol{x}^{(e)}$ the codeword obtained after inverting the e first bits of $\boldsymbol{x}$, where $1 \le e \le k$.

Example 1 Consider the binary sequence $\boldsymbol{x}$ = 01000110 of length k=8. By performing an exhaustive search, we find:
$\boldsymbol{x}^{(1)}$= 11000110, $\boldsymbol{x}^{(2)}$= 10000110, $\boldsymbol{x}^{(3)}$= 10100110,
$\boldsymbol{x}^{(4)}$= 10110110, $\boldsymbol{x}^{(5)}$= $\boldsymbol{x}^{(6)}$= 10111110, $\boldsymbol{x}^{(7)}$= 10111000,
$\boldsymbol{x}^{(8)}$= 10111001.

z therefore, the balanced codewords are $\boldsymbol{x}^{(1)}$, $\boldsymbol{x}^{(3)}$ and $\boldsymbol{x}^{(7)}$. However, this approach is not efficient as it is long and only finds inversion points from the left direction; this leads to a complexity of Ok^2digit operations.

B. *Using RDS*

The sequence $\boldsymbol{x}$ is converted into bipolar form through the following mapping: $0 \longrightarrow -1$ and $1 \longrightarrow +1$. That is $\boldsymbol{x} = x_1, \dots, x_k$, with $x_i \in \{-1, +1\}$. The running digital sum at index *i* is the cumulative sum of sequence elements until index *i*. We define the running digital sum from left $RDSL_i$ and the running digital sum from right $RDSR_i$ as follow:

$$RDSL_i = \sum_{j=1}^{i} x_j = RDSL_{i-1} + x_i \qquad (2)$$

$$RDSR_i = \sum_{j=1}^{i} x_j = RDSR_{i-1} + x_i \qquad (3)$$

Remark 1 $RDSL_i = RDSR_i$. However, the sequence $\boldsymbol{x}$ is balanced when $RDSL_i = RDSR_i = 0$.

Theorem 1 *For a non-balanced sequence, inversion points are found at indexes i if $RDSL_i$ = $RDSR_{i+1}$, for inversion starting from left (Knuth's algorithm case) and $RDSR_i$ = $RDSL_{i\ 1}$ for inversion starting from the right of the sequence.*

Proof: *This can be proved by observing that the difference between the number of symbols ` 1' and ` + 1' before and after the inversion point index i are equal. This is case when inverting from left or right direction. Therefore $RDSL_i$ = $RDSR_{i+1}$ (inversion from left) and $RDSR_i$ = $RDSL_{i\ 1}$ (inversion from right).*

Lemma 1 *From an already balanced sequence, inversion points are found √at indexes i if and only $RDSL_i$ = $RDSR_{i+1}$ = 0 for left inversion and $RDSR_i$ = $RDSL_{i\ 1}$ =0, for right one.*

Lemma 2 *An already balanced sequence always has at least one inversion point located at the last index k.*

Let us consider the same sequence as in Example 1, $\boldsymbol{x}$ = 01000110.

Index (i)	***1***	***2***	***3***	***4***	***5***	***6***	***7***	***8***
$x_i \in \{-1, +1\}$.	*-1*	*+1*	*-1*	*-1*	*-1*	*+1*	*+1*	*-1*
$RDSL_i$	*-1*	*+0*	*-1*	*-2*	*-3*	*-2*	*-1*	*-2*
$RDSR_{i+1}$	*-1*	*-2*	*-1*	*+0*	*+1*	*+0*	*-1*	*X*
$RDSR_i$	*-2*	*-1*	*-2*	*-1*	*+0*	*+1*	*+0*	*-1*
$RDSL_{i-1}$	*X*	*-1*	*+0*	*-1*	*-2*	*-3*	*-2-1*	
Left balance	√		√				√	
Right balance		√		√				√

(4)

Using the RDS approach as described in Theorem 1, the balanced codewords with inversion performed from the left are $\boldsymbol{x}^{(1)}$, $\boldsymbol{x}^{(3)}$, and $\boldsymbol{x}^{(3)}$, while balanced codewords from the right are $\boldsymbol{x}^{(2)}$, $\boldsymbol{x}^{(4)}$, and $\boldsymbol{x}^{(8)}$, as presented in (4). This is an efficient way of finding inversion points from both left and right directions; this approach presents a linear complexity of Ok^2operation digits.

This RDS approach to find inversion points can easily be interpreted using graphical representation. A (γ, τ)-random walk is a path with increases of and decreases of . Any RDS walk always generate a $\{-1;+1\}, \{-1;\ +1\}$-random walk; that is a walk with increases and decreases of either +1 or 1. This is simply due to the bipolar nature of a binary sequence.

Graphically, inversion points from the left are the intersection dots between the random walks of $RDSL_i$ and $RDSL_{i+1}$ (obtained by a horizontal shift of $RDSR_i$ walk to the left). Similarly, those from the right are intersection points between the random walks of $RDSR_i$ and $RDSL_{i\ 1}$ (obtained by a horizontal shift of $RDSL_i$ walk to the right).

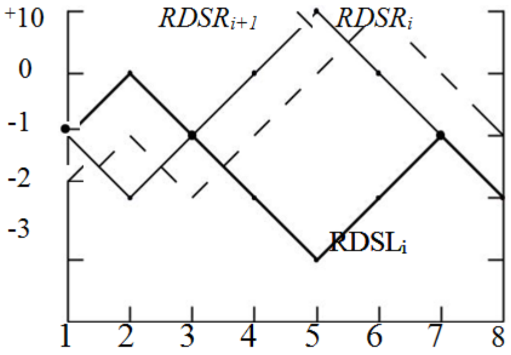

Fig. 1. Balanced points found at indexes 1, 3 and 7 from left for the sequence 01000110.

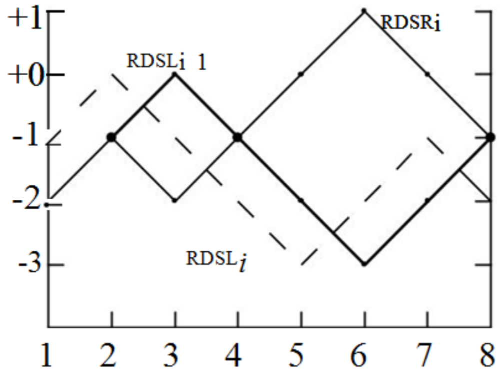

Fig. 2. Balanced points found at indexes 2, 4 and 8 from right for the sequence 01000110.

C. *Using Weights*

Let us consider the binary sequence $\boldsymbol{x}$ with $x_i \in \{0, 1\}$. Considering the left direction, we define $L_i(0)$ and $L_i(1)$

respectively as the weight of ‘0’ and the weight of ‘1’ within $\boldsymbol{x}$ from index 1 to *i*; similarly, *R_i(0)* and *R_i(1)* as the weight of ‘0’ and the weight of ‘1’ within $\boldsymbol{x}$ from index *i* + 1 to *k*. However for the right direction, *R_i(0)* and *R_i(1)* would denote respectively, the weight of ‘0’ and the weight of ‘1’ within $\boldsymbol{x}$ from index *i* to *k*; while *L_i(0)* and *L_i(1)*, the weight of ‘0’ and the weight of ‘1’ from index 1 to *i-1*.

Lemma 3 For any binary sequence, inversion points are found at indexes i either from left or right direction, if

$$|L_i(0) - L_I(1)| = |R_i(0) - R_i(1)|$$

Let us consider once more the sequence from Example 1,

$\boldsymbol{x}$ = 01000110.

Index (i)	***1***	***2***	***3***	***4***	***5***	***6***	***7***	***8***
$x_i \in \{0,1\}$.	*0*	*1*	*0*	*0*	*0*	*1*	*1*	*0*
$\lvert L_i(0) - L_I(1)\rvert$	***1***	*0*	***1***	*2*	*3*	*2*	***1***	*2*
$\lvert R_i(0) - R(1)\rvert$	***1***	*2*	***1***	*0*	*1*	*0*	***1***	*X*

(5)

Index (i)	***1***	***2***	***3***	***4***	***5***	***6***	***7***	***8***
$x_i \in \{0,1\}$.	*0*	*1*	*0*	*0*	*0*	*1*	*1*	*0*
$\lvert R_i(0) - R(1)\rvert$	*1*	***1***	*3*	***1***	*0*	*1*	*0*	***1***
$\lvert L_i(0) - L_I(1)\rvert$	*X*	***1***	*0*	***1***	*2*	*3*	*2*	***1***

(6)

Using the weight approach described in Lemma 2, (5) presents all possible inversion points from the left which are found are indexes 1, 3 and 7; while (6) shows inversion points from the right at indexes 2, 4 and 8.

The weight approach to find inversion points is as efficient as the method based on RDS presented in Section II.B with a linear complexity of $\mathcal{O}k^2$. However, the weight method might be more appropriate as it does not require the bipolar representation of the initial binary sequence.

III. Encoding Method Based on Information Sequence Candidates

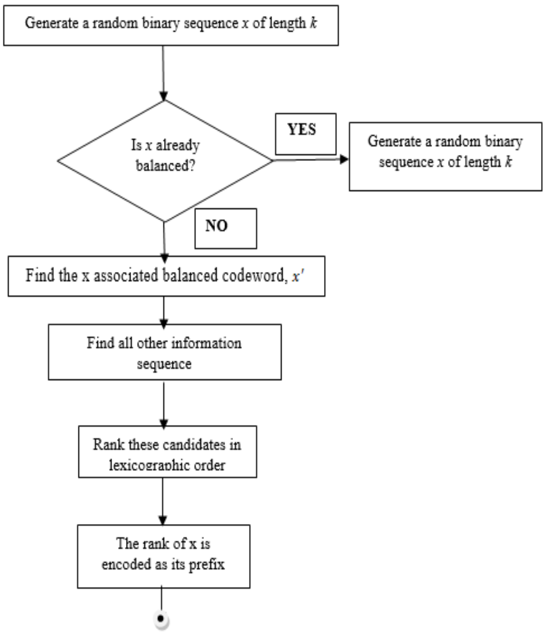
Generate a random binary sequence x of length k
Is x already balanced?
YES
Generate a random binary sequence x of length k
NO
Find the x associated balanced codeword, x'
Find all other information sequence
Rank these candidates in lexicographic order
The rank of x is encoded as its prefix

Fig. 3. Encoding scheme

The idea behind this encoding scheme is to associate every information sequence of length *k*, to a balanced codeword within the cardinality of 2^k as presented in Fig. 3.

Given a random binary sequence $\boldsymbol{x}$ to be encoded, if $\boldsymbol{x}$ is already balanced, a protocol can be adopted between transmitter and receiver to have a prefix-less codeword; otherwise , $\boldsymbol{x}$ is balanced following the Knuth’s algorithm, then the associated balanced codeword is obtained and denoted as $\boldsymbol{x}$ from the least inversion point index. All other information sequence candidates associated to $\boldsymbol{x'}$ are captured and listed in the lexicographic order. The prefix of $\boldsymbol{x}$ corresponds to its rank amongst the information sequence candidates.

Example 2 Let us consider all sequences of length *k* = 4. The tabular below shows all information sequence candidates associated to every balanced codeword.

$\boldsymbol{x'}$	0011	0101	0110	1001	1010	1100
$\boldsymbol{x}$	1011	1101	1000	0001	0010	0000
	1111	**1010**	1110	0111	**0101**	0100
	1100		**1001**	**0110**		**0011**

(7)

(7) shows the encoding process described in [3], whereby balanced codewords (marked in bold) are part of the information sequence candidates.

$\boldsymbol{x'}$	0011	0101	0110	1001	1010	1100
$\boldsymbol{x}$	1011	1101	1000	0001	0010	0000
	1111		1110	0111		0100

(8)

However, in our scheme, balanced codewords are excluded from the cardinalities of information sequence candidates as shown in (8) based on Theorem 2.

Theorem 2 *Any balanced codeword of length k is always associated to another balanced one*.

Proof: By applying Knuth’s inversion algorithm on any already balanced codeword, another balanced codeword is generated; at the worst-case scenario, it is found by inverting all bits as stated in Lemma 2.

Let us denote by $\boldsymbol{c(x')}$, the cardinality of information sequence candidates associated to a balanced codeword. In Example 2, $1 \leq c(\boldsymbol{x'}) \leq 2$.

The inclusion of balanced sequences within the set of information sequence candidates as presented in [3], adds an extra rank in the ranking process. As we described in Lemma 2, an already balanced sequence always leads to at least one another balanced sequence obtained by inverting all bits which might or not be the associate one.

Let us denote by max{RDSLi} and min {RDSLi}, the maximum and minimum of RDSLi respectively performed on any sequence.

Theorem 3

$$c(\boldsymbol{x'}) = max\{\, RDSLi\} - min\{\,\mathrm{RDSL}i\}.$$

Proof: It was proved in [3] that $c(\boldsymbol{x'}) = max\{\, RDSLi\} - min\{\,\mathrm{RDSL}i\} + 1$. the balanced codeword was removed out of every set of information sequence candidate. Therefore the new $c(\boldsymbol{x'})$ is subtracted by 1, that is $c(\boldsymbol{x'}) = max\{RDSLi\} - min\{\text{RDSL}i\}$.

Theorem 4

$$1 \le c(\boldsymbol{x'}) \le k/2$$

Proof: It was established in [3] that $2 \le c(\boldsymbol{x'}) \le k/2 + 1$; then after removing the balanced codeword out of every set of information sequence candidate, it follows that $1 \le c(\boldsymbol{x'}) \le k/2$ .Therefore, the required prefix redundancy for this scheme is $log_2\, k/2$; this is a significant improvement on the Knuth's algorithm with a redundancy of $log_2\, k/2$. The prefix is obtained from ranking the information sequence candidates associated to a balanced codeword from 0 to $\frac{k}{2} - 1$.

IV. Decoding

The decoding process is illustrated in Fig. 4. The process is as follow: The prefix is extracted from the overall received codeword of length $n = k + p$ as the first $k/2$ bits; then all the $k/2$ information sequence candidates associated to $\boldsymbol{x'}$ are listed and ordered lexicographically from 0 to $k/2\ -1$. Finally, the prefix is mapped to the rank of the right original information sequence.

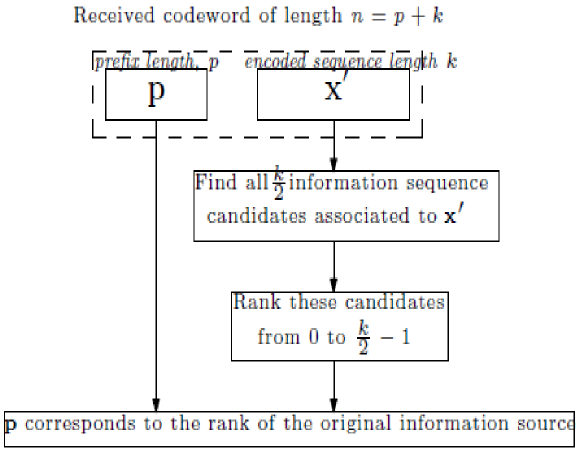
Received codeword of length $n = p + k$
prefix length, p
encoded sequence length k
p
x'
Find all $\frac{k}{2}$ information sequence candidates associated to $\mathbf{x}'$
Rank these candidates from 0 to $\frac{k}{2} - 1$
p corresponds to the rank of the original information source

Fig. 4. Flow chart of the decoding process

Example 3 We want to decode the received codeword, **11**11000011, where the bold and underline word represents the prefix.

Info. seq. candidates	Prefix rank
01000011	0 (00)
00000011	1 (01)
00110011	x(Not ranked because
00111011	2 (10)
00111111	3 (11)

(9)

(9) shows all information sequence candidates associated to the balanced codeword 11000011 with their correspondent prefix ranks.

Therefore, the received codeword **11**11000011 is mapped to the original information sequence, 00111111.

One can notice that the proposed scheme requires a redundancy of $\log_2$ (8/2)=2 to encode any 8 bits sequence as in Example 3, while the Knuth's one is $\log_2(8) = 3$ and the Immnink & Weber's one as in [3] is $\log_2$ (8/2 + 1) = 2.32.

V. Study of the Sparseness of $c(\boldsymbol{x'})$

Let N(λ, k) be the number of possible balanced codewords $\boldsymbol{x'}$ of length k such that $c(\boldsymbol{x'}) = \lambda$. The following equation holds from Theorem 4;

$$\sum_{\lambda=1}^{k/2} \text{N}(\lambda, k) = \binom{k}{k/2} \tag{10}$$

The value N(λ, k) has been evaluated in [3] for $/2\ -1$. This was done using the computation of the number of bipolar sequences whose running sum lies within two finite bounds B1 and B2 (with B2 > B1), as proposed by Chien [5].

The interval of running sum values that a sequence may reach, also referred to as the digital sum variation (*DSV*) is given by B = B2−B1 + 1. Each iteration in the random walk of a sequence defines an entry of a BXB connection matrix, M_B .

M_B is such that, $M_B\,(i,j) = 1$, if there is a path in the random walk from state s_i to state s_j ; and $M_B\,(i,j) = 0$ if no path can be established. For each iteration, a random walk of the running sum can only move one state up or down. Therefore, $M_B\,(i+1,i) = M_B\,(i,i+1) = 1$ and $M_B\,(i,i) = 0$, where $i,j = 1,2, \ldots B-1$ as presented in (11).

$$M_B = \begin{bmatrix} 0 & 1 & 0 & \ldots & 0 & 0 \\ 1 & 0 & 1 & \ldots & 0 & 0 \\ 0 & 1 & 0 & & 0 & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 0 & 1 \\ 0 & 0 & 0 & \cdots & 1 & 0 \end{bmatrix} \tag{11}$$

${M_B}^{\boldsymbol{k}}(i,j)$ denotes the $(i.j)^{\boldsymbol{th}}$ entry of the kth power of M_B .

Theorem 5 The number of balanced codewords $\boldsymbol{x'}$ of length k and $c(\boldsymbol{x'}) = \lambda$, $N(\lambda,k)$ for $1 \le \lambda \le k/2\ -1$is such that

$$N(\lambda,k) = \sum_{i=1}^{\lambda+1} M_{\lambda+1}^{k}\,(i.i) - 2\sum_{i=1}^{\lambda} M_{\lambda}^{k}\,(i.i) + \sum_{i=1}^{\lambda-1} M_{\lambda-1}^{k}\,(i.i)$$

Proof: The number of balanced codewords such that $c(\boldsymbol{x'}) = \lambda$ for $2 \le \lambda \le \log_2 k/2$ +1 in [3] was as follow,

$$N(\lambda,k) = \sum_{i=1}^{\lambda} M_{\lambda}^{k}\,(i.i) - 2\sum_{i=1}^{\lambda-1} M_{\lambda-1}^{k}\,(i.i) + \sum_{i=1}^{\lambda-2} M_{\lambda-2}^{k}\,(i.i)$$

However, there is one starting state where a sequence has the maximum RDS spanning B + 1. Similarly, all sequences with $c(\,\boldsymbol{x'}) =$ B have two starting states;

sequences with $c(\boldsymbol{x}') = B - 1$ have three starting states. This implies that

$$\sum_{i=1}^{\lambda} M_{\lambda}^{k}(i.i) = N(\lambda,k) + 2N(\lambda-1,k) + 3N(\lambda-2,k) + 4N(\lambda-3,k) + 5N(\lambda-4,k) \dots = \sum_{i=0}^{\lambda-1}(I+1)\,N(\lambda-i,k).$$

This leads to the following

$$N(\lambda,k) = \sum_{i=1}^{\lambda+1} M_{\lambda+1}^{k}(i.i) - 2\sum_{i=1}^{\lambda} M_{\lambda}^{k}(i.i) + \sum_{i=1}^{\lambda-1} M_{\lambda-1}^{k}(i.i)$$

A simplified expression of MB was provided in [3] based on a formula to compute powers of MB derived by Salkuyeh [6] as follow:

$$\sum_{i=1}^{B} M_{B}^{k}(i.i) = 2^{k}\sum_{i=1}^{B} cos^{k}(i.i).\frac{\pi i}{B+1} \qquad (12)$$

This makes the computation of $N(\lambda,k)$ much simpler as follow:

$$N(\lambda,k) = 2^{k}\left(\sum_{i=1}^{\lambda+1} cos^{k}\frac{\pi i}{\lambda+2} - 2\sum_{i=1}^{\lambda} cos^{k}\frac{\pi i}{\lambda+1} + \sum_{i=1}^{1} cos^{k}\frac{\pi i}{\lambda}\right) \qquad (13)$$

The computation of $N(\lambda,k)$ as presented in (13) becomes obvious for special values of as shown in (14). The enumeration of sequences corresponding to these values of as well as the pseudo code for computing $c(\boldsymbol{x}')$, for generating the ordered set of information sequence candidates and for determining the prefix index were provided in [3].

Info. seq. candidates	Prefix rank
1	2
2	$2(2^{(k/2)-1})$
$k/2$	$k(k-4),(k>4)$
$k/2$	k

(14)

VI. Analysis and Discussions

We would like to compute the average number of bits denoted as $H(k)$ required to encode the prefix index of a sequence of length k. The number of information sequence candidates associated to a balanced codeword x' is $c(x)$ out of the $2^{k} - \binom{k}{k/2}$ possible information sequence candidates.

$$= \sum_{i=1}^{\lambda/2} \lambda N(\lambda,k) = 2^{k} - \binom{k}{\frac{k}{2}} \qquad (15)$$

It follows that

$$\sum_{i=1}^{\frac{\lambda}{2}} \lambda N(\lambda,k)\log_2 k \Big/ 2^{k} - \binom{k}{\frac{k}{2}} \qquad (16)$$

The minimum redundancy for the full set of balanced code-words is given in [2] by:

$$H_o(k) = k - \log_2\binom{k}{\frac{k}{2}} \approx \frac{1}{2}\log_2 k + 0.326 \qquad (17)$$

The average number of bits for the construction in [3] is as follow:

$$H_1(k) = 2^{-k}\sum_{i=2}^{\frac{\lambda}{2}+1} \lambda N(\lambda,k)\log_2\lambda \qquad (18)$$

The average number of bits for the method in [4] is given-by

$$H_2(k) = \sum_{c=1}^{\frac{\lambda}{2}} P(c)\,A(c) \qquad (19)$$

where $P(c) = 2^{c+1-k}\binom{k-1-c}{\frac{k}{2}-c}, 1 \le c \le k/2,$

$$d = c - 2^{\lfloor \log_2 c \rfloor}, \text{and AV(c)} = (\mathrm{c} - 2\mathrm{d}).\lfloor \log_2 c \rfloor.\frac{1}{2^{\lfloor \log_2 c \rfloor}} + 2d.\frac{1}{2^{\lfloor \log_2 c \rfloor}}.\lceil \log_2 c \rceil$$

Table I presents the comparison of the average number of bits necessary to encode the prefix from various schemes. Let d_{H_a}, H_b be the difference between the average prefix length H_a and H_b ; we observed that $d_H, H_o \le 0.61$, $d_H.H_1 \le$0:64 and $d_{H_2}, H \le 1.23$.

TABLE I: COMPARISON OF THE PREFIX AVERAGE NUMBER OF BITS

k	H_0	H	H_1	H_2
4	1.4150	0.8000	1.4387	0.5000
8	1.8707	1.4632	1.8985	0.9375
16	2.3483	2.0806	2.3790	1.3706
32	2.8370	2.6629	2.8691	1.8082
64	3.3314	3.2207	3.3641	2.2516
128	3.8286	3.7615	3.8616	2.7039
256	4.3272	4.2902	4.3603	3.1647
512	4.8265	4.8104	4.8597	3.6330
1024	5.3261	5.3246	5.3594	4.1082

Fig. 5 presents the average number of bits for prefix encoding for various schemes. The proposed scheme's average redundancy given by (16), performed better than the average minimum redundancy for the full set of balanced codewords as in (17) and the Immink & Weber average redundancy as in (18). However, the difference in length between the proposed (14) scheme and the Al-rababa's et al average redundancy as in (19) is less than 1.23.

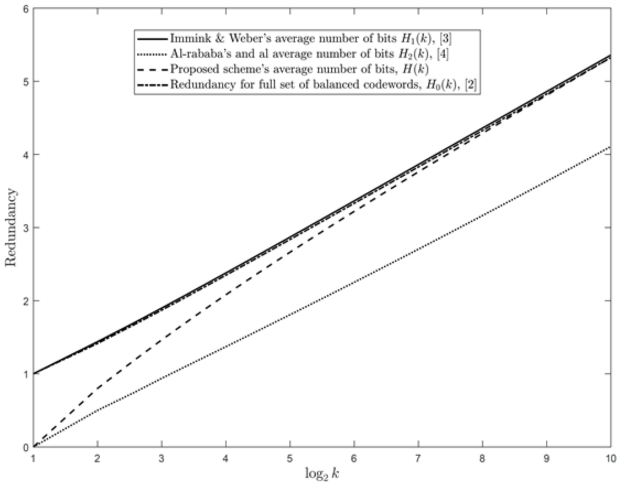

Fig. 5. $H_o(k)$, $H(k)$, $H_1(k)$ and $H_2(k)$ vs $\log_2 k$.

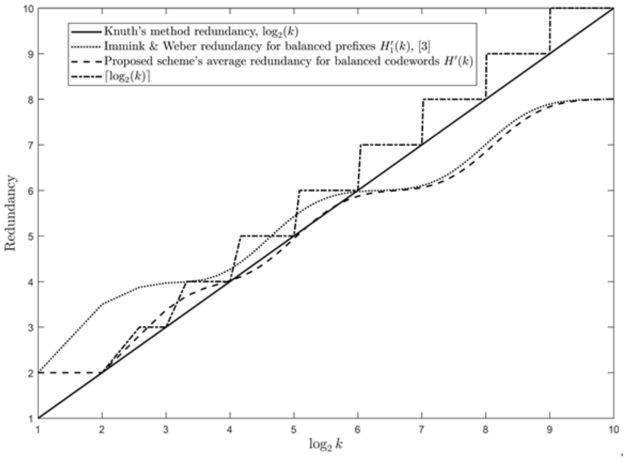

Fig. 6. $H'(k)$, $H_1'(k)$ $\log_2 k$ and $\lceil\log_2 k\rceil$

Fig. 6 shows the comparison between the average redundancy for balanced prefixes for $H(k)$, $H_1(k)$ denoted as $H'(k)$ and $H_1'(k)$respectively as well as $\log_2 k$ and $\lceil\log_2(k)\rceil$. $H'(k)$ is obtained from a simple modification of $H(k)$ provided in (16) as follow

$$H'(k)=\sum_{i=1}^{\frac{\lambda}{2}}\lambda N(\lambda,k)(\Delta\lambda)\Big/2^k-\binom{k}{\frac{k}{2}} \qquad (20)$$

Similarly, $H'(k)$ is derived form $H_1(k)$ given in (18) as fellow:

$$H_1'(k)=2^{-k}\sum_{i=2}^{\frac{\lambda}{2}+1}\lambda N(\lambda,k)(\Delta\lambda) \qquad (21)$$

where $(\Delta\lambda)$ correspond to the smallest value of length k such that $\frac{k}{2}\leq\lambda$. The graphs of $\log_2(k)$ and $\lceil\log_2 k\rceil$represents the minimum redundancy and that of integer valued redundancy of the traditional Knuth's construction. We observe that, it is only from $k>64$ that the average redundancy of the scheme presented in [3] is less than that of the Knuth scheme; whereas

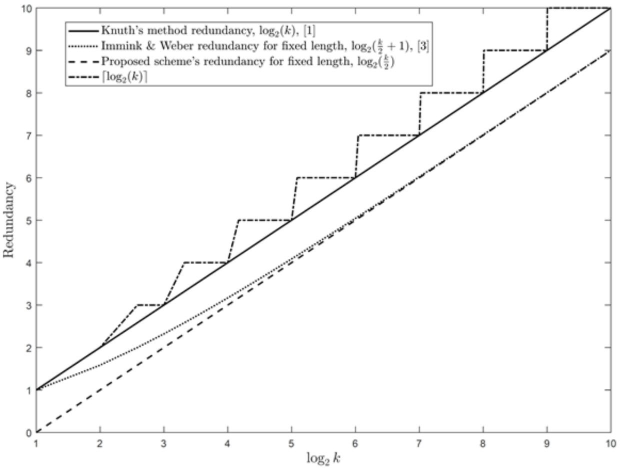

Fig. 7. Fixed length schemes

According to Theorem 4, the two coding schemes are applicable for the proposed scheme. For the fixed length prefix construction, the encoding of the prefix requires exactly $\log_2(k/2)$ bits; whereas for the variable length (VL) scheme, the prefix length varies between 1 and $\log_2(k/2)$ depending on the nature of the information sequence. However, the VL scheme is more efficient than the fixed length one on the average basis.

Fig. 7 presents the fixed length performance, we observed that the proposed scheme is more efficient than the classic Knuth scheme for any length and it performs better than the fixed length construction presented in [3] for k < 512. For practical systems purpose, a redundancy can only be a positive integer value. Fig 8 presents the rounded up fixed length schemes. This confirmed the previous assumption that the proposed fixed length scheme is more efficient than that of [3] for k < 512.

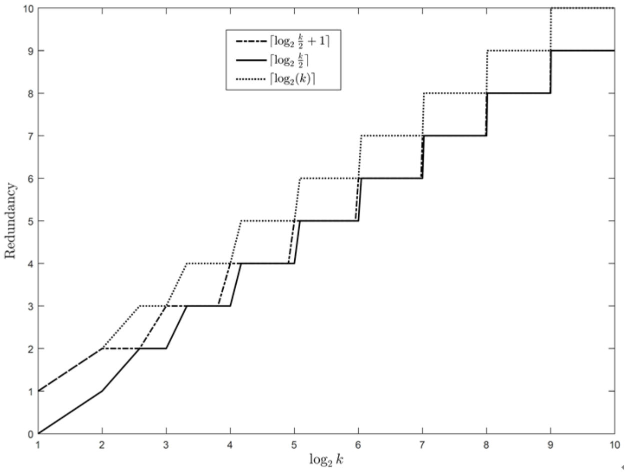

Fig. 8. Rounded up fixed length schemes

VII. Conclusion

We have presented a modification of the construction given in [6], for encoding and decoding of binary codewords. The proposed scheme requires exactly $\log_2(k/2)$ bits for the fixed length prefix and a prefix length between 0 and $\log_2(k/2)$ bits for VL scheme. The sparseness of the prefix length was analyzed, and the average efficiency of this scheme was discussed and compared to existing ones. The proposed construction is very advantageous compared to some prior schemes as look-up tables are not used and it is less redundant.

Furthermore, this scheme can be featured with the construction provided in [7] to provide the overall codeword balancing (information with prefix). As future work, we intend to apply the proposed scheme on the overall codeword length to close the remaining gap from the lower redundancy bound.

ACKNOWLEDGEMENT

The authors would like to acknowledge Jos Weber for proofreading this article and for constructive discussions. This work is supported by the center for Telecommunication and the Global Excellence Stature program, University of Johannesburg.

REFERENCES

[1] D. E. Knuth, “Efficient balanced codes,” *IEEE Transactions on Information Theory*, vol. 32, no. 1, pp. 51–53, Jan. 1986.

[2] J. H. Weber and K. A. S. Immink, “Knuth’s balanced codes revisited,” *IEEE Transactions on Information Theory*, vol. 56, no. 4, pp. 1673-1679, April 2010.

[3] K. A. S. Immink and J. H. Weber, “Very efficient balanced codes,” *Journal on Selected Areas in Communications*, vol. 28, no. 2, pp. 188-192, Feb. 2010.

[4] A. Al-rababa’a, D. Dube, and J. Y. Chouinard, “Using bit recycling to reduce Knuth’s balanced codes redundancy,” in *Proc. Canadian Workshop on Information Theory*, Toronto, ON, Canada, Jun. 18-21, 2013, pp. 6-11.

[5] T. M. Chien, “Upper bound on the efficiency of Dc-constrained codes,” *Bell Syst. Tech. J.*, vol. 49, pp. 2267–2287, Nov. 1970.

[6] D. K. Salkuyeh, “Positive integer powers of the tri-diagonal Toeplitz matrices,” *International Mathematical Forum*, vol. 1, no. 22, pp. 1601–1065, 2006.

[7] E. N. Mambou and T. G. Swart, “Encoding and decoding of balanced q-ary sequences using a Gray code prefix,” in *Proc. IEEE International Symposium on Information Theory*, Barcelona, Spain, Jul. 10–15, 2016, pp. 380–384.

Ebenezer Esenogho received the B.Eng. degree (2nd Class Hons Upper Div.) in Computer Engineering in 2008, the M.Eng. degree in Electronic/Telecomm Engineering from the University of Benin in 2012. He previously lectured at the University of Benin before embarking on his Ph.D. program in 2013. Prior to now, he was involved in research/lecturing with the Centre for Radio Access and Rural Technology, University of KwaZulu-Natal, Centre of Excellence, where he rounded-up his Ph.D. degree in Electronic/Telecom (5G Cognitive Network) in 2017. Currently, he is admitted into the prestigious Global Excellence Stature Post-Doctoral Research Fellowship at the University of Johannesburg, Auckland Park under the Institute for Intelligent System (IIS), Center for Telecommunication Research (CfT). He is a recipient of several grants/scholarship/award including the CEPS/Eskom’s HVDC 2013, CEPS/Eskom’s HVDC 2014, J. W Nelson 2015 and GES 2017 and 2018. He is passionate about leadership and is evidence in his services as a postgraduate representative for student Engineering council in the management board of University of KwaZulu-Natal during his PhD days, 2016-2017 and presently serving as the first postdoctoral research fellowship representative in the University of Johannesburg Senate Council, 2018-date. He was a UJ/DST Research delegate to the H2020-ESASTAP EU-South Africa STI Cooperation on Strengthing Technology Research and Innovation in Vienna, Austria. He has authored/co-authored several peer-reviewed journals and conference papers and, reviewed for some notable ISI/Scopus journals in his field. His research interests are in the Fifth Generation (5G) Wireless Networks, Cognitive Radio Networks, Smart Grid Networks, IoT/IoE, SDN/SDR, Wireless Sensor Networks, Artificial Intelligence, Mobile Computing, Visible light communication and Coding/Information theory. He is a registered Engineer and corporate member of SAIEE/IEEE region 8

Mr. Elie Ngomseu Mambou graduated from the prestigious University of Johannesburg, South Africa. He holds in his academic portfolio, two bachelor’s degree in Information and Technology and Electrical and Engineering with distinctions in 2014. In 2016, he bagged two master’s degree in Telecommunication Engineering from Beijing Institute of Technology, China and University of Johannesburg in with distinction. He is a recipient of the Chancellor's award for being the best graduating student in the entire University of Johannesburg in 2017 academic session. He has a solid academic background and has been into active teaching, research and development in institutions of higher learning to date with various international conferences and journals publications couple with past industrial experience. Currently, he is rounding up his PhD degree at the University of Johannesburg, South Africa. His research interests are in coding, information theory, Fifth Generation (5G) Wireless Networks, Cognitive Radio Networks, IoT/IoE, Wireless Sensor Networks, Artificial Intelligence, Mobile Computing, Visible light communication. He is a registered as a graduate Engineer with ECSA and a member of SAIEE/IEEE region 8

Hendrik Christoffel Ferreira is a Professor in Digital Communications and Information Theory at the University of Johannesburg, Johannesburg, South Africa. He studied electrical engineering at the University of Pretoria, South Africa, where he obtained his Ph.D. in 1980. He worked as a visiting researcher at Likebait in San Diego. He joined the Rand Afrikaans University in 1983, where, in 1989, he was appointed full professor. In recognition of his excellence in research and educating post-graduate students, he has been appointed as a research professor at the University of Johannesburg in 2007. He is a Fellow of the SAIEE, the South African Institute of Electrical Engineers. He has published over to 300 research papers on topics such as digital communications, power line communications, vehicular communication systems. With his work he introduced and developed a new theme in Information Theory, namely coding techniques for constructing combined channel codes, where error correction and channel properties are considered jointly. Ferreira is a pioneering initiator and stimulator of the research fields of Information Theory and Power Line Communications in South Africa. He has also been an organizer for the IEEE Information Theory Society and Power Line Communications within South Africa and Africa. He is a member of the Technical Committee for Power Line Communications of the IEEE Communications Society, and he has served on the Technical Program Committee of several IEEE conferences, including the IEEE (ISIT) International Symposium on Information Theory, the IEEE (ISPLC) International Symposium on Power Line Communications, and the IEEE Africon and Chinacom conferences. He is a Fellow of SAIEE and Senior Member of IEEE region 8.