Constant Weight Codes: An Approach Based on Knuth's Balancing Method

Vitaly Skachek [1,2]
Coordinated Science Laboratory
University of Illinois, Urbana-Champaign
1308 W. Main Street
Urbana, IL 61801, USA
E-mail: vitalys@illinois.edu

Kees A. Schouhamer Immink [1]
Turing Machines Inc, Willemskade 15b-d
3016 DK Rotterdam, The Netherlands
School of Physical and Mathematical Sciences
Nanyang Technological University
21 Nanyang Link, Singapore 637371
E-mail: immink@turing-machines.com

***Abstract*—In this article, we study properties and algorithms for constructing sets of 'constant weight' codewords with bipolar symbols, where the sum of the symbols is a constant q, $q \neq 0$. We show various code constructions that extend Knuth's balancing vector scheme, $q = 0$, to the case where $q > 0$. We compute the redundancy of the new coding methods.**

***Index Terms*—Balanced code, channel capacity, constrained code, magnetic recording, optical recording.**

I. Introduction

Let q be an integer. A set $\mathcal{C}$, which is a subset of

$$\left\{ \boldsymbol{w} = (w_1, w_2, \ldots, w_n) \in \{-1, +1\}^n \; : \; \sum_{i=1}^{n} w_i = q \right\} ,$$

is called a *constant weight code* of length n. If $q = 0$, the code $\mathcal{C}$ is called a *balanced code*. A vector $\boldsymbol{w} \in \mathcal{C}$ is called a *code word*. The *weight* of a word equals the number of $+1$'s in it. An *encoder* for the code $\mathcal{C}$ is an invertible mapping from the set of all vectors $\boldsymbol{w} \in \{-1, +1\}^m$ onto the vectors in $\mathcal{C} \subseteq \{-1, +1\}^n$, for some integer $m \geq 1$. The *redundancy* produced by the encoder is $n - m$. For practical purposes, we are interested in encoders which are simple and computationally efficient, and whose inverse, *decoder*, is also computationally efficient.

Balanced codes have found application in cable transmission, optical and magnetic recording. A survey of properties and methods for constructing balanced codes can be found in [3]. A simple encoding technique for generating balanced codewords, which is capable of handling (very) large blocks was described by Knuth [5] in 1986.

For a general value of q, constant weight codes are useful in a variety of applications such as fault-tolerant circuit design and computing, pattern generation for circuit testing, identification coding, and optical overlay networks [1], [2]. However, a simple encoding algorithm for a given imbalance $q \neq 0$ has not been known. The publications dealing with constructions of unbalanced codes that the authors are aware of are the geometric approach by Tian *et al.* [1] and the enumerative method by Schalkwijk [4] (see also [2]). Below, we briefly discuss these two methods.

The time complexity of the method in [1] is $\Theta(\mathsf{w}^2)$, where $\mathsf{w} = (n + q)/2$ is the weight of the encoded word. If $\mathsf{w} = \Theta(n)$, the complexity of this method behaves as $\Theta(n^2)$. Moreover, the rate of the produced code is not optimal. The authors of [1] mention that they lose about $1 - 3$ bits per each block in their construction, where the number of blocks is proportional to w. Therefore, if q is small, the rate loss is proportional to m, and the produced redundancy is also proportional to m.

By contrast, the redundancy, produced by the method in [4], is essentially optimal. However, this method has some drawbacks. First, it requires large registers. Second, it suffers from error propagation. Third, this method has higher complexity, at least $\Omega(\mathsf{w}^2)$ operations over integers. If q is small, the complexity of this method is too high.

In this article we study various simple and efficient methods of generating bipolar codewords of imbalance q, $q \neq 0$, where mounting size of hardware and massive error propagation can be avoided. Without loss of generality we assume $q > 0$, since, for $q < 0$, codewords can simply be obtained by inverting all m symbols of a codeword with $q > 0$.

II. Background

Let the word be $\boldsymbol{w} = (w_1, w_2, \ldots, w_n)$, $w_i \in \{-1, +1\}$, and let q be the imbalance of the user symbols defined by

$$q = \sum_{i=1}^{n} w_i . \tag{1}$$

The cardinality of the set of all q-balanced words of length n, n and q even, can be approximated by [6]

$$\binom{n}{\frac{n+q}{2}} \approx \frac{2^n}{\sqrt{\frac{n\pi}{2}}} e^{-\frac{q^2}{2n}}, \quad n >> 1.$$

[1]This project was supported by the National Research Foundation of Singapore under the grant *Theory and Practice of Coding and Cryptography*, award number: NRF-CRP2-2007-03.

[2]The work was done while Vitaly Skachek was with the School of Physical and Mathematical Sciences, Nanyang Technological University, 21 Nanyang Link, Singapore 637371.

Then the redundancy of a full set of q-balanced codewords is

$$\frac{1}{2}\log_2 n + \frac{q^2}{2n}\log_2 e + 0.326, \quad n >> 1. \tag{2}$$

We notice that the redundancy consists of two terms, namely the term $\frac{1}{2}\log_2 n + 0.326$, the redundancy of the set of balanced words, and the q-dependent term $\frac{q^2}{2n}\log_2 e$. Observe that the q-dependent term decreases with increasing n (equivalently, increasing m).

Knuth's encoding scheme

Knuth published an algorithm for generating sets of bipolar codewords with equal numbers of '$+1$'s and '-1's. In Knuth's method, an m-bit user word, m even, consisting of bipolar symbols valued ± 1, is forwarded to the encoder. The encoder inverts the first k bits of the user word, where k is chosen in such a way that the modified word has equal numbers of '$+1$'s and '-1's. Such an index k can always be found. This k is represented by a balanced (prefix) word $\boldsymbol{u}$ of length p. The p-bit prefix word followed by the modified m-bit user word are both transmitted, so that the rate of the code is $m/(m+p)$. The receiver can easily undo the inversion of the first k bits received by decoding the prefix. Both encoder and decoder do not require large look-up tables, and Knuth's algorithm is therefore very attractive for constructing long balanced codewords. Knuth showed that the redundancy p is roughly equal to $\log_2 m$, for $m >> 1$ [5], [7]. Modifications of the generic scheme are discussed in Knuth [5], Alon *et al.* [8], Al-Bassam & Bose [9], Tallini, Capocelli & Bose [10], and Weber & Immink [7].

We now try to answer the question how to construct a constant weight code with codewords, where $q \neq 0$.

III. Methods for constructing constant weight codes

A. General setup

Let $\boldsymbol{w} = (w_1, w_2, \ldots, w_m)$, $w_i \in \{-1, +1\}$, be an arbitrary input word. We use the notation

$$d(\boldsymbol{w}) = \sum_{i=1}^{m} w_i. \tag{3}$$

Let $\boldsymbol{w}^{(k)}$ be the word $\boldsymbol{w}$ with its first k bits inverted, and let $\sigma_k(\boldsymbol{w})$ stand for $d(\boldsymbol{w}^{(k)})$. We also denote $q' = d(\boldsymbol{w})$.

By far the simplest method for generating q-balanced codewords first generates a balanced codeword by using Knuth's method, and appends q '$+1$' symbols to the balanced codeword. The method is attractive as it is simple to implement, and as it avoids mass error propagation. The redundancy of this scheme is $p+q$, where $p \approx \log_2 m$ is the redundancy of Knuth's base scheme. From (2) it follows that this simple method is far from optimum since the redundancy does not decrease with increasing m.

Alternative methods, to be discussed below, first use a slightly modified version of Knuth's algorithm for generating q-balanced words. These method may fail to generate such words and, in a second step, the 'failed' words are modified. Information regarding the modification made is, as in Knuth's scheme, carried by the prefix.

Let us assume that we apply Knuth's algorithm to $\boldsymbol{w}$ to generate a q-balanced codeword, $0 \le q \le m$. That is, we scan $\boldsymbol{w}$ and seek an index k, $0 \le k \le m$, such that

$$\sigma_k(\boldsymbol{w}) = -2\sum_{i=1}^{k} w_i + d(\boldsymbol{w}) = q.$$

For $q = 0$, such an index k can always be found. For $q \neq 0$, however, the encoder may fail to find such an index k. Since, from the above, $\sigma_0(\boldsymbol{w}) = d(\boldsymbol{w}) = q'$, (no symbols inverted) and $\sigma_m(\boldsymbol{w}) = -d(\boldsymbol{w}) = -q'$ (all m symbols inverted), we conclude that words $\boldsymbol{w}$ with q' in the range $-q < q' < q$ and $\sigma_k(\boldsymbol{w}) < q$, $0 \le k \le m$, cannot be translated into a q-balanced codeword. User words, called *delinquent* words, with a disparity within the range $|q'| \le q-2$ cannot be encoded by Knuth's algorithm, and some alternative has to be found. In the sequel, we present various code constructions that encode the delinquent words.

B. Unbalanced prefix

It is assumed that the index k in Knuth's algorithm is represented by a p-bit balanced prefix $\boldsymbol{u}$, p even. Delinquent words can be balanced by the introduction of prefixes of imbalance q_p, $2 \le q_p \le p$. If the imbalance of a delinquent word, $q' < 0$, we invert all symbols of that user word. Then a delinquent word or its inverted version may be balanced by choosing a prefix with imbalance $q_p = q - |q'|$. An inversion made can be signalled by a unique prefix, thus requiring two distinct prefixes of imbalance q_p, $2 \le q_p < q$. For $q_p = q$, since $q' = 0$, we only need one prefix, the all-one prefix. The unbalanced prefix is an indication to the decoding side that the user word $\boldsymbol{w}$ did not undergo changes. Thus, if $q \le p$ it is quite straightforward to generate a codeword having imbalance q using a simple modification of Knuth's method.

Observe that this method is efficient (with respect to produced redundancy) when p is at most approximately the length of representation of the index k, i.e. when $p = O(\log m)$.

C. Flipping tail patterns

1) Method decription: In an alternative way of encoding, the encoder finds $(q-q')/2$ '-1' symbols in the delinquent $\boldsymbol{w}$ that are inverted into '$+1$' symbols. Note that, since $|q'| \le q-2$, at most $q-1$ symbols '-1' have to be identified and inverted. The positional information of the $(q-q')/2$ inverted symbols is conveyed to the receiver by the prefix, which adds to the redundancy of the code. The prefix conveys either information on the index k or the information on the positions of the inverted symbols. The index $k \in \{0, \ldots, m\}$ requires $m+1$ combinations of the prefix, while the positional information of the inverted symbols requires N_p combinations, totaling $m+1+N_p$ prefix combinations. We identify a segment of a delinquent $\boldsymbol{w}$, where the $(q-q')/2$ symbols '-1' can be found, so that

N_p and the redundancy of the new method can be calculated. We exemplify the above with a simple example, where $q = 2$.

Example: For $q = 2$, we find that delinquent words $\boldsymbol{w}$ are characterized by a) $q' = 0$, b) $w_1 = +1$, and c) $w_m = -1$. From $q' = 0$, we conclude that only one '-1' has to be located and inverted. As the tail symbol of all delinquent words is $w_m = -1$, we conclude that by inverting the tail symbol, we can translate any input word $\boldsymbol{w}$ into a codeword with imbalance $q = 2$. Clearly, we require $m + 2$ different prefixes to uniquely identify the modification made at the transmitter's site.

Theorem 1: Let $\boldsymbol{w}$ be a delinquent word, $q \geq 2$ and q is even, then there are at least $(q-q')/2$ symbols valued '-1' in the $\ell = (3q-q')/2-2$ tail bits $(w_{m-\ell+1}, w_{m-\ell+2}, \cdots, w_m)$.
Proof: In case the Knuth encoder fails, we have $\sigma_k(\boldsymbol{w}) < q$ and $\sigma_k(\boldsymbol{w})$ is even, and thus

$$\sigma_k(\boldsymbol{w}) = -2\sum_{i=1}^{k} w_i + q' \leq q-2, \quad 0 \leq k \leq m\ , \tag{4}$$

or

$$2\sum_{i=k+1}^{m} w_i \leq q + q' - 2, \quad 0 \leq k \leq m\ . \tag{5}$$

Let ℓ^+ be the number of '$+1$'s in the last ℓ positions of $\boldsymbol{w}$ and ℓ^- be the number of '-1's in these ℓ positions. We have that

$$\ell^+ + \ell^- = \ell\ . \tag{6}$$

From (5) (with $k = m - \ell$), we also have

$$2\ell^+ - 2\ell^- \leq q + q' - 2\ . \tag{7}$$

We conclude from (6) and (7) that

$$\ell^- \geq \ell/2 - (q+q')/4 + 1/2\ .$$

By the substitution $\ell = (3q - q')/2 - 2$, we obtain

$$\ell^- \geq (q-q')/2 - 1/2\ .$$

Since q and q' are both even, and ℓ^- is integer, we have $\ell^- \geq (q-q')/2$, as required. ∎

Since for a delinquent word we have $|q'| \geq q-2$, it follows from Theorem 1 that in the $q - 2 + i$ tail bits there are at least i symbols valued '-1'. We therefore conclude that only a limited number of inversion combinations needs to be taken into account.

Then, in case Knuth's algorithm fails to produce an unbalanced codeword, we can take care of that 'failed' word $\boldsymbol{w}$ by inverting $(q-q')/2$ symbols '-1' in the $(3q-q')/2-2$ tail bits of $\boldsymbol{w}$. In the worst case, $q-1$ '-1' symbols have to be inverted into a '$+1$' in $2q-3$ tail symbols.

In the next section, we compute the number of modifications, N_p, that have to be made to delinquent words in order to generate a q-balanced codeword.

TABLE I
SET OF SHORTEST TAIL STRINGS FOR $q = 4$.

$q' = 2$	$q' = 0$	$q' = -2$
-1	$-1-1$	$-1-1-1$
$-1+1$	$-1+1-1$	$-1+1-1-1$
$-1+1+1$	$-1-1+1$	$-1-1+1-1$
	$-1+1+1-1$	$-1+1+1-1-1$
	$-1+1-1+1$	$-1+1-1+1-1$

TABLE II
NUMBER OF PREFIXES, N_p, VERSUS q.

q	N_p	N'_p
2	1	1
4	13	12
6	131	111
8	1429	1183
10	16795	13362

2) Redundancy computation: The computation of N_p is directly related to the computation of the redundancy of the new method. Note that N_p is independent of the word length m. The number, N_p, of prefixes required to identify the $(q-q')/2$ symbol inversions made in the $\ell = (3q-q')/2-2$ tail bits is upper bounded by

$$N_p < \sum_{\substack{q'=-q+2 \\ q'\ even}}^{q-2} \binom{\frac{3q-q'}{2}-2}{\frac{q-q'}{2}},$$

or after a change of variable, we obtain

$$N_p < \sum_{i=1}^{q-1} \binom{2q-2-i}{q-i}.$$

The counting of N_p can be accomplished by setting up a forest of $q-1$ binary trees for all possible values of q' $\in \{-q+2, -q+4, \ldots, q-2\}$. Starting from the root with w_m, we generate all possible valid tail strings $\{w_{m-k}, \ldots, w_m\}$, where a string is valid if $2\sum_{i=m-k}^{m} w_i \leq q + q' - 2$. We terminate a string $\{\ldots, w_{m-2}, w_{m-1}, w_m\}$ when the string contains $(q-q')/2$ symbols '-1'. Theorem 1 guarantees that the length of a string is at most $2q-3$. N_p is computed by summing all strings of all $q-1$ trees. For $q = 4$, the 13 shortest tail strings are given in Table I, where the less significant symbol of the strings is associated with the tail symbol w_m of a codeword and the tree root. Appending the above ℓ-bit tail strings to $m-\ell$ leading '$+1$' symbols yields the m-bit vectors, required to "unbalance" the delinquent words. Thus, from the above 13 m-bit vectors plus Knuth's $m+1$ balancing vectors we can select at least one vector such that the inner product of the selected and an input word equals $q = 4$ for any even value of $m > 4$. We have computed N_p for various values of q. Table II shows results of our computations. An alternative computation of N_p is based on a finite-state machine model of the running sum $\sum w_k$. The number of sequences, whose running sum $\sum w_k$ remains within given limits can be computed by a model introduced by Chien [11]. The sum sequence

$$z_k = \sum_{i=1}^{k} w_i, \qquad k = m - \ell + 1, \ldots, m$$

remains within the bounds $N_1 = (q' - q)/2 + 1$ and $N_2 = q' + \ell$ (there are at most ℓ consecutive tail symbols '+1', where $\ell = (3q - q')/2 - 2$, see Theorem 1). The number of values that z_k can take within the (tail) interval $k = m - \ell + 1, \ldots, m$ is at most $N_2 - N_1 + 1 = 2q - 2$. The stream w_k at any instant $k = m - \ell + 1, \ldots, m$ can be seen as the output of a Moore-type finite-state machine with $N = 2q - 2$ states, whose $N \times N$ connection matrix, D_N, is defined by $D_N(i,j) = 1$ if a transition is allowable and $D_N(i,j) = 0$ otherwise [11]. We have

$$\begin{cases} D_N(i+1,i) = D_N(i,i+1) = 1, & \text{if } i = 1, 2, \ldots, N-1, \\ D_N(i,j) = 0, & \text{otherwise.} \end{cases}$$

The (i,j)-th entry of the m-th power of D_N is denoted by $D_N^m(i,j)$. After a manipulation, we find that the number N_p, is given by

$$N_p = \sum_{\ell=q-1}^{2q-3} D_{2q-2}^{\ell}[\ell + 1, 2q - 3], \tag{8}$$

where we count the number of distinct sequences ending in state $2q - 3$ of length $q - 2 + i$, $i = 1, \ldots, q - 1$ having i '-1' symbols. Note that there are at least i symbols valued '-1' in the $q - 2 + i$ tail bits.

We can improve the redundancy by a small modification of the tree search: in case $q' < 0$, we invert many symbols in a large tail. This costs a lot of bits of redundancy. Thus, in the worst case, $q' = -q + 2$, we invert $q - 1$ symbols in $2q - 3$ tail symbols. If we assume that we invert all bits of that user word, then the imbalance q' becomes $-q'$. The worst case $q' = -q + 2$ becomes $q' = q - 2$, where only one symbol has to be inverted in $q - 1$ tail bits. For this inverted word, we search for the smallest tail string that contains $(q - q')/2$ '-1' symbols.

In a systematic way, we set up a forest of $q - 1$ binary trees as described above, where we follow a valid path until either we tallied $(q - q')/2$ '-1' symbols (as described above in the first method) or we tallied $(q - q')/2$ '+1' symbols. The m-bit balancing vectors are obtained as follows. The ℓ-bit tail strings with $(q - q')/2$ '+1' symbols are appended to $m - \ell$ '+1' symbols, while the ℓ-bit tail strings with $(q - q')/2$ '-1' symbols are appended to $m - \ell$ '-1' symbols. We tally the leaves of all trees, which yields N'_p, the number of tail strings found. Results of computations are collected in Table II.

We conclude that N_p is rapidly mounting with increasing imbalance q. As the binary logarithm of the sum $m + N_p$ determines the redundancy of the code, the method is either efficient for small q or very large m.

D. Different operational modes

In order to encode an arbitrary block of '± 1's of length m into a block of imbalance q, the methods discussed in Sections III-B and III-C produce $\approx q$ redundant bits in the worst case. Below, we introduce an alternative encoding method which encodes an arbitrary block of data of length m into a block with imbalance $q = \frac{3}{2}\log_2 m$ bits with only $\log_2 m + o(\log m)$ redundant bits.

This method can also be adopted for some other parameter combinations. For example, for the input word $\boldsymbol{w}$ of length m and desired imbalance q, such that $m > 2^{2q/3}$, we can use a combined decoder that applies Knuth's algorithm to the first $m - 2^{2q/3}$ input bits, and the proposed algorithm to the remaining $2^{2q/3}$ bits. The resulting encoder outputs words with imbalance q and redundancy $\log_2 m + \frac{2}{3}q + o(\log m)$.

Recall that if $q' < 0$, we first invert the word. Then, there are five different modes of operation: modes 1, 2, 3, 4 and 5. Mode 5 contains four different submodes, denoted by 5-1 – 5-4. The choice of mode depends on the value of q' and the structure of the input word $\boldsymbol{w}$. The selected mode and a possible inversion (or not) are denoted by a special prefix. There are eight different modes, for each mode there are also two possibilities for inversion. These 16 possibilities are represented by balanced prefixes with six bits.

In some operational modes we make use of a balanced suffix of length $\log_2 m + o(\log m)$. In Modes 1 and 5-1 this suffix denotes the number of bits flips. In Modes 2 and 5-2 the suffix carries additional weight, thus making the total imbalance of the word equal q. In Modes 3 and 5-3, the suffix serves as an index of an entry in the input word. Modes 4 and 5-4 do not use suffix. Instead, in these modes, the values of the last $\log_2 m - 1$ bits in the tail of the word are encoded by a *location* of a special string of length $2\log_2 m$, which is inserted into the input word. The tail is then discarded. Therefore, in these two modes, the resulting length of the word increases by $\log_2 m + o(\log m)$ bits as well.

Recall that $q' = \sum_{i=1}^{m} w_i$. Assume without loss of generality, $q' \geq 0$, otherwise flip all the bits, and mark this in the prefix of the encoded word. Denote by $\mu = \sum_{i=m-\log_2 m+2}^{m} w_i$ a sum of the last $\log_2 m - 1$ symbols in $\boldsymbol{w}$.

Mode 1. $q' > \frac{3}{2}\log_2 m$: In this case, we just apply the Knuth's algorithm. We append a balanced suffix of $\log_2 m + o(\log m)$ bits at the tail of the block, to denote the actual number of bit flips.

Mode 2. $\frac{1}{2}\log_2 m \leq q' \leq \frac{3}{2}\log_2 m$: In this case, we just append the suffix of the total imbalance $q_p = q - q' \leq \log_2 m$.

Mode 3. $0 \leq q' < \frac{1}{2}\log_2 m$ and the word $\boldsymbol{w}$ contains a subsequence "+1 +1 +1 ... +1" *of length $\geq \log_2 m$:* Then, the sum of all bits in $\boldsymbol{w}$, except for this subsequence, is less than $-\frac{1}{2}\log_2 m$. By flipping all the bits in $\boldsymbol{w}$, except for the above subsequence and one '-1' immediately after it, we make the total imbalance $\geq \frac{1}{2}\log_2 m + \log_2 m = \frac{3}{2}\log_2 m$. Then, we sequentially flip the bits in the subsequence "+1 +1 +1 ... +1", from the first to the penultimate, to make the total imbalance equal q. It is straight-forward to see that this imbalance can always be achieved. We need $\log_2 m + o(\log m)$ bits to encode the index of the first flipped bit in this subsequence by the balanced suffix. Observe, that from knowing that index, the flipped region can be uniquely determined, and so the whole operation is invertible.

Mode 4. $0 \leq q' < \frac{1}{2}\log_2 m$, the word $\boldsymbol{w}$ contains no subsequence "+1 +1 +1 … +1" *of length $\geq \log_2 m$ and $q' \geq \mu$:* Denote $\hat{\boldsymbol{w}} = (w_1, w_2, \ldots, w_{m-\log_2 m+1})$. We insert the following string between two consecutive symbols in $\hat{\boldsymbol{w}}$:

$$-1 \underbrace{+1+1+1 \ldots +1}_{s_1} \underbrace{-1-1-1 \ldots -1}_{s_2} +1 \ ,$$

$s_1 + s_2 = 2\log_2 m - 2$, $s_1 \geq \log_2 m$ and $s_2 \geq 1$. Denote the sum of the elements in this string by η. Under the given constraints, η can take any even value in $[2, 2\log_2 m - 4]$.

We aim to select the values of s_1 and s_2 such that after the insertion the total imbalance of the resulting word (i.e., $\hat{\boldsymbol{w}}$ with the inserted sequence) becomes $\frac{3}{2}\log_2 m$. Note, that after the insertion of the string, the imbalance becomes $q' - \mu + \eta$. Since $q' \geq \mu$, we obtain that

$$0 \leq q' - \mu < \frac{3}{2}\log_2 m - 1 \ ,$$

and so there exists a string as above such that $q' - \mu + \eta = \frac{3}{2}\log_2 m$.

Next, we decide on the location of this insertion. The location of the insertion $i \in \{0, 1, 2, \ldots, m - \log_2 m + 1\}$, represents the encoding of the last $\log_2 m - 1$ bits that were removed from $\boldsymbol{w}$. These bits can take

$$2^{\log_2 m - 1} = \frac{m}{2}$$

different values. Since $\frac{m}{2} < m - \log_2 m + 1$ (for all $m \geq 4$), we have enough different locations to encode the values of these bits.

There is no other substring "+1 +1 +1 … +1" of length $\geq \log_2 m$ in $\hat{\boldsymbol{w}}$, and so the inserted string can be uniquely identified. Therefore, all steps are invertible.

Mode 5. $0 \leq q' < \frac{1}{2}\log_2 m$, the word $\boldsymbol{w}$ contains no subsequence "+1 +1 +1 … +1" *of length $\geq \log_2 m$ and $q' < \mu$:* Consider the word $\tilde{\boldsymbol{w}} \triangleq \boldsymbol{w}^{(m-\log_2 m+1)}$. Since

$$\sum_{i=1}^{m-\log_2 m+1} w_i = q' - \mu < 0 \ ,$$

we obtain that $d(\tilde{\boldsymbol{w}}) = (\mu - q') + \mu > (q' - \mu) + \mu = q' \geq 0$ and also $d(\tilde{\boldsymbol{w}}) = (\mu - q') + \mu > \mu$. Thus, the resulting $\tilde{\boldsymbol{w}}$ falls under one of the cases considered in Modes 1–4. We apply the corresponding encoding method in Modes 1–4 to $\tilde{\boldsymbol{w}}$ (we refer to these cases as Modes 5-1 – 5-4, respectively).

As we can see, all encoding modes are invertible. Therefore, the whole encoding algorithm is invertible.

Time complexity: It can be verified that each mode requires up to $O(m)$ bit operations and $O(m)$ incerments/decrements of counters of size $O(\log m)$ bits.

IV. CONCLUSIONS

We have studied various constructions of simple codes that generate codewords with a given imbalance q.

We have shown that we can extend Knuth's balancing vector scheme, $q = 0$, to the case where $q > 0$. Part of the input words can be converted into a codeword of imbalance q by the original Knuth algorithm. The other 'delinquent' input words require a dedicated encoding step. We have presented a construction where delinquent words are balanced by an imbalance of the prefix. In a second construction, at most $(q - q')/2$ '-1' symbols of the delinquent word are inverted, where q' denotes the imbalance of the user word. The information regarding the changes made is carried by the prefix word. For $q = 2$, it suffices to invert the tail bit of the delinquent word. For larger values of q, more combinations of symbol inversions have to be defined which have to be carried by the prefix costing more redundancy. We have computed the redundancy of the new method for various values of m and q. We conclude that the method is either efficient for small q or very large m.

Finally, we presented a new method which allows to encode an arbitrary block of data of length m bits into a block of imbalance $q = \frac{3}{2}\log_2 m$ with only $\log_2 m + o(\log m)$ redundant bits. The time complexity of this method is $O(m)$ bit operations and $O(m)$ increments/decrements of counters of size $O(\log_2 m)$ bits. This method can be used in the algorithm that encodes arbitrary binary input words of length $m > 2^{2q/3}$ into words with imbalance q and redundancy $\log_2 m + \frac{2}{3}q + o(\log m)$.

REFERENCES

[1] C. Tian, V.A. Vaishampayan, and N. Sloane, A Coding Algorithm for Constant Weight Vectors: A Geometric Approach Based on Dissections', *IEEE Trans. Inform. Theory,* vol. IT-55, no. 3, pp. 1051-1060, March 2009.

[2] Y.M. Chee, C.J. Colbourn, and A.C.H. Ling, 'Optimal Memoryless Encoding for Low Power Off-Chip Data Buses', Proceedings of the 2006 IEEE/ACM international Conference on Computer-Aided Design, San Jose, California, November 05 - 09, 2006.

[3] K.A.S. Immink, *Codes for Mass Data Storage Systems,* Second Edition, ISBN 90-74249-27-2, Shannon Foundation Publishers, Eindhoven, Netherlands, 2004.

[4] J.P.M. Schalkwijk, 'An Algorithm for Source Coding', *IEEE Trans. Inform. Theory,* IT-18, pp. 395-399, 1972.

[5] D.E. Knuth, 'Efficient Balanced Codes', *IEEE Trans. Inform. Theory,* vol. IT-32, no. 1, pp. 51-53, Jan. 1986.

[6] P. Stanica, 'Good Lower and Upper Bounds on Binomial Coefficients', *Journal of Inequalities in Pure and Applied Mathematics,* vol. 2, Art. 30, 2001.

[7] K.A.S. Immink and J. Weber, 'Very Efficient Balancing of Codewords', *IEEE Journal on Selected Areas of Communications*, vol. 28, pp. 188-192, 2010.

[8] 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.

[9] S. Al-Bassam and B. Bose, 'On Balanced Codes', *IEEE Trans. Inform. Theory,* vol. IT-36, no. 2, pp. 406-408, March 1990.

[10] 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.

[11] T.M. Chien, 'Upper Bound on the Efficiency of Dc-constrained Codes', *Bell Syst. Tech. J.,* vol. 49, pp. 2267-2287, Nov. 1970.