Efficient Balanced Codes

DONALD E. KNUTH, HONORARY MEMBER, IEEE

Abstract—Coding schemes in which each codeword contains equally many zeros and ones are constructed in such a way that they can be efficiently encoded and decoded.

A BINARY word of length \( m \) can be called balanced if it contains exactly \( \lfloor m/2 \rfloor \) ones and \( \lfloor m/2 \rfloor \) zeros. Let us say that a balanced code with \( n \) information bits and \( p \) parity bits is a set of \( 2^n \) balanced binary words, each of length \( n + p \).

Balanced codes have the property that no codeword is "contained" in another; that is, the positions of the ones in one codeword will never be a subset of the positions of the ones in a different codeword. This property makes balanced codes attractive for certain applications, such as the encoding of unchangeable data on a laser disk [2]. Conversely, if we wish to form as many binary words of length \( m \) as possible with the property that no word is contained in another, Sperner's lemma [3] tells us that we can do no better than to construct the set of all balanced words of length \( m \).

A balanced code is efficient if there is a very simple way to encode and decode \( n \)-bit numbers. In other words, we want to find a one-to-one correspondence between the set of all \( n \)-bit binary words and the set of all \( (n + p) \)-bit codewords such that, if \( w \) corresponds to \( w' \), we can rapidly compute \( w' \) from \( w \) and vice versa. Furthermore, we want \( p \) to be very small compared with \( n \), so that the code is efficient in its use of space as well as time. For example, it is trivial to construct a balanced code with \( n \) information bits and \( n \) parity bits by simply letting the binary word \( w \) correspond to the codeword \( w = w \), where \( w \) is the complement of \( w \). Encoding and decoding is clearly efficient in this case, but memory space is being wasted.

Let \( M(m) = \left( \begin{array}{c} m \\ \lfloor m/2 \rfloor \end{array} \right) \) be the total number of balanced binary words of length \( m \). To have a balanced code with \( n \) information bits and \( n \) parity bits by simply letting the binary word \( w \) correspond to the codeword \( w' = w \), where \( w \) is the complement of \( w \). Encoding and decoding is clearly efficient in this case, but memory space is being wasted.

Let \( M(m) = \left( \begin{array}{c} m \\ \lfloor m/2 \rfloor \end{array} \right) \) be the total number of balanced binary words of length \( m \). To have a balanced code with \( n \) information bits, we clearly need to have enough parity bits \( p \) so that \( M(n + p) \geq 2^n \). Stirling's approximation tells us that

\[
\log M(m) = m - \frac{1}{2} \log m - \frac{1}{2} \log \pi - \frac{\epsilon(m)}{m},
\]

where \( 0 \leq \epsilon(m) \leq 1.25/\ln 2 \approx 0.61; \) all logarithms here have radix 2, and the constant \( 1/2 \log \pi/2 \) is approximately 0.326. Therefore, in particular, we must have \( p > 1/2 \log n + 0.326 \) in any balanced code.

The purpose of this correspondence is to describe a balanced code with \( 2^n \) information bits and \( p \) parity bits, for which serial encoding and decoding is especially simple. This means, for example, that 256-b words can be encoded efficiently with only eight parity bits, obtaining 264-b balanced words; thus the percentage of memory devoted to overhead in order to satisfy the balance constraint is only \( 8/264 = 3.03 \) percent.

A similar scheme that allows efficient parallel decoding and efficient serial encoding is also described. The parallel method for \( n \) information bits takes roughly \( \log n + 1/2 \log \log n \) parity bits in its simplest form, and the \( 1/2 \log \log n \) term can be replaced by 1 at the expense of additional complexity. For example, a balanced code with 256 information bits and nine parity bits will be constructed explicitly. This code has the property that the 256-b word \( w \) corresponds to a balanced 265-b codeword \( w' = uw^{(k)} \), where \( w^{(k)} \) denotes \( w \) with its first \( k \) bits complemented and where the 9-b prefix \( u \) determines \( k \). It is clearly possible to determine \( w \) quickly from \( w' \) in such a code.

A SIMPLE PARALLEL SCHEME

Let \( \nu(w) \) be the total number of ones in the binary word \( w \), let \( \nu_k(w) \) be the number of ones in the first \( k \) bits of \( w \), and let \( \nu(w^{(k)}) \) be the word \( w \) with its first \( k \) bits complemented. For example, if \( w = 0111010110 \), we have \( \nu(w) = 6, \nu_2(w) = 3, \) and \( \nu(w^{(k)}) = 100010110 \). Since \( k - \nu_k(w) \) of the first \( k \) bits of \( w \) are zeros, we have

\[
\nu(w^{(k)}) = \nu(w) + k - 2\nu_k(w).
\]

This relation is the key to all the coding schemes that will be described in the following.

If \( w \) has length \( n \) and if we let \( \sigma_k(w) \) stand for \( \nu(w^{(k)}) \), the quantity \( \sigma_k(w) \) changes by \( \pm 1 \) when \( k \) increases by one, so it describes a "random walk" from \( \sigma_0(w) = \nu(w) \) to \( \sigma_n(w) = n - \nu(w) \).

Now comes the point: the value \( \lfloor n/2 \rfloor \) lies in the closed interval between \( \nu \) and \( n - \nu \) for all integers \( \nu \); hence a \( k \) always exists such that \( \sigma_k(w) = \lfloor n/2 \rfloor \). In other words, every word \( w \) can be associated with at least one \( k \) such that \( w^{(k)} \) is balanced. If we encode \( k \) in a balanced word \( u \) of length \( p \), and if \( n \) and \( p \) are not both odd, we can let \( w \) correspond to the balanced codeword \( uw^{(k)} \). If \( n \) and \( p \) are both odd, we can use a similar construction, but the
value of \( k \) should be chosen so that \( \sigma_k(w) = \lceil n/2 \rceil \); then again \( uw^{(k)} \) will be balanced.

For example, suppose that we want a balanced code of this sort having eight information bits. Every 8-b word \( w \) defines at least one value of \( k \) such that \( w^{(k)} \) is balanced; we never need to use \( k = 8 \), so we can assume that \( 0 \leq k < 8 \). If we arbitrarily choose eight balanced words \( (u_0, \ldots, u_7) \) of length five, we can represent \( w \) by the balanced word \( u_k w^{(k)} \). (Such a choice of \( u \)'s is possible since \( M(5) = 10 \times 8 \).) This gives us a code with eight information bits and five parity bits. Parallel decoding is easy, because \( k \) is determined from \( u \) by table-lookup; then \( w \) is \( w^{(k)} u_k \). Serial encoding is also easy, because we can determine \( k \) by computing \( \sigma_k(w) \) for \( k = 0, 1, \ldots \) until finding \( \sigma_k(w) = 4 \).

A similar scheme gives a balanced code with 256 information bits and 11 parity bits, because \( M(11) > 256 \). In general, this approach works with \( n \) information bits and \( p \) parity bits whenever \( M(p) \geq 2^{n/2} \).

**A Simple Serial Scheme**

We can decrease the number of parity bits in the previous construction by using all the bits of \( u \). The idea is to encode \( w \) as \( uw^{(k)} \) for some \( u \) and \( k \), as before, but \( u \) does not have to be balanced; any imbalance in \( u \) will be compensated by a corresponding imbalance in \( w^{(k)} \). For example, when \( n = 4 \) and \( p = 2 \) we can simply let \( k = 0 \) when \( 0 < v(w) < 4 \); if \( v(w) = 1, 2, 3 \) we can let \( u = 11, 01, 00 \), respectively. The remaining two cases \( w = 0000 \) and \( w = 1111 \) are handled by letting \( k = 2 \) and \( u = 10 \).

When \( n = 8 \) and \( p = 3 \), an exhaustive analysis shows that no similar scheme exists in which \( k \) is determined by \( u \); however, we can construct a code in which \( u \) is determined by \( v(w) \) as follows:

\[
\begin{array}{cccccc}
0 & 001 & 4 & 3 & 101 & 3 \vspace{1pt} \\
1 & 011 & 3 & 4 & 100 & 4 \vspace{1pt} \\
2 & 010 & 4 & 5 & 000 & 5 \vspace{1pt} \\
\end{array}
\]

The word \( uw^{(k)} \) will be balanced in this case if and only if \( v(uw^{(k)}) = v(u) + \sigma_k(w) = \lceil n/2 \rceil \); this happens if and only if \( \sigma_k(w) = s \), where the values of \( s \) have been tabulated. Since \( \sigma_k(w) \) runs from \( v(n) \) to \( n - v(n) \), it is easy to verify in each case that some value of \( k \) will make \( \sigma_k(w) = s \). The code is defined by choosing the smallest \( k \) such that \( \sigma_k(w) = v(w) \).

One complication exists, however: two different values of \( v(w) \) correspond to the same value of \( u \), namely, \( u = 001 \) has both \( v(w) = 0 \) and \( v(w) = 8 \). This is not really a difficulty, because it arises only for the two words \( w = 00000000 \) and \( 11111111 \) (when we know that \( k = 4 \)); but it is an annoying anomaly. The best way to avoid it is to consider only the values of \( \sigma_k(v) \) modulo 8 when decoding. We know \( v(w) \) mod 8, so we choose the smallest \( k \) such that \( \sigma_k(v) = v(w) \) (modulo 8).

Incidentally, there is no balanced code with \( n = 8 \) and \( p = 2 \), since \( M(10) = 252 \) is less than 256. Therefore, the balanced code just defined is optimum for \( n = 8 \).

A similar balanced code can be constructed with \( p \) parity bits and \( n = 2^p \) information bits, for all \( p \geq 3 \), as follows. For \( 0 \leq l < n \), let \( u_l \) be a \( p \)-bit word such that the number

\[
s_l = n/2 + \lceil p/2 \rceil \cdot r(u_l)
\]

lies between \( l \) and \( n - l \), inclusive. This should be a permutation of the \( p \)-bit words; that is, \( l \neq l' \) should imply that \( u_l \neq u_{l'} \). An \( n \)-bit word \( w \) is then encoded as \( u_w w^{(k)} \), where \( l = v(w) \) mod \( n \) and where \( k \) is minimal such that \( \sigma_k(w) \equiv s_l \) (modulo \( n \)). An \( (n + p) \)-bit word \( w' = uw \) is decoded as \( v^{(k)} \), where \( k \) is minimal such that \( \sigma_k(v) = l \) (modulo \( n \)) and where \( l \) is determined by the condition \( u = u_l \).

It remains to specify the correspondence between \( l \) and \( u_l \). Since \( p \) is much smaller than \( n \), the choice is delicate only when \( l \) is near \( n/2 \). It is not difficult to find a mapping that assigns the balanced words to values of \( l \) near \( n/2 \); the rest of the codes are essentially arbitrary.

For example, let \( p = 8 \) and \( n = 256 \). We want to permute the 8-b words \( u_{128+l} \) for \( 128 \leq l < 128 \) in such a way that \( 0 \leq l + v(u_{128+l}) - 4 \leq 2^l \) when \( l \geq 0 \) and \( 0 \leq l + v(u_{128+l}) - 4 \leq 2^l \) when \( l < 0 \). The inequalities are always valid when \( |l| \geq 4 \), so the choice of \( u_l \) is important only when \( 124 < l < 132 \). A suitable mapping is obtained by letting \( u_l = a_l b_l \), where \( a_l \) and \( b_l \) are the 4-b binary representations of \( (l + 8) \mod 16 \) and \( b_l \) is the 4-b binary representation of \( (l + 8) \mod 16 \).

\[
\l = (120 + a + 16((b - a) \mod 16)) \mod 256
\]

**An Optimized Parallel Scheme**

We have now constructed two balanced codes with \( n = 256 \); one has \( p = 11 \) parity bits to allow parallel decoding, and the other has \( p = 8 \) parity bits to allow serial decoding. The author has been unable to construct a parallel decoder for such schemes when \( n = 256 \) and \( p = 8 \), but the following method gives parallel decoding when \( p = 9 \) and in general whenever \( n = 2^p - 1 \).

The idea is to choose \( l \) words \( (u_l, \ldots, u_{l+p-1}) \) of \( p \) bits each and to choose \( l \) values \( (k_1, \ldots, k_l) \) in the range \( 0 \leq k_j \leq n \).
such that every random walk

\((0, \sigma_0(w)), (1, \sigma_1(w)), \ldots, (n, \sigma_n(w))\) \((\ast)\)

is guaranteed to pass through one of the points

\[ P_j = (k_j, \lfloor (n + p)/2 \rfloor - v(u_j)) \]

for some \(j\). We can then encode \(w\) as the balanced word \(u_w(w)\). Parallel decoding is possible since the \(p\)-bit parity word \(u\) determines the extent of complementation.

We shall choose the \(u\)'s and \(k\)'s in such a way that \(v(u_{j+1}) - v(u_j) = 0\) or \(1\) and \(k_{j+1} - k_j = 1 - (v(u_{j+1}) - v(u_j))\). This means that \(P_{j+1} - P_j\) is always either \((1, 0)\) or \((0, -1)\). For example, when \(p = 3\) and \(n = 4\), we can let the pairs \((k_j, u_j)\) be

\[
(0,001) \quad (1,010) \quad (2,100) \quad (2,011) \quad (3,101) \quad (4,110)
\]

so that the points \(P_j\) are

\[
(0,2) \quad (1,2) \quad (2,2) \quad (2,1) \quad (3,1) \quad (4,11).
\]

We shall also choose \(k_1 = 0\) and \(k_n = n\), so that any random walk \((\ast)\) must lie entirely "above" or "below" the set of \(P\)'s.

Let \(P_0 = (0, M)\) and \(P_1 = (n, m)\) be the extreme points. If \((\ast)\) does not intersect the set \(\{P_1, \ldots, P_1\}\), we must have either \(\sigma_0(w) > M\) and \(\sigma_n(w) > m\) or \(\sigma_0(w) < M\) and \(\sigma_n(w) < m\). Since \(\sigma_0(w) + \sigma_n(w) = n\), this cannot happen unless \(n > M + m + 2\) or \(n \leq M + m - 2\). Therefore, it suffices to design the construction so that \(|M + m - n| \leq 1\).

A moment's thought now makes it clear what to do: we list all \(p\)-bit numbers \(u\) in any order such that the weights \(v(u)\) are nondecreasing, then we choose \(l - n + h + 1\) of these near the "middle" of the sequence such that \(v(u_i) = h\) for some \(h\). For example, the case \(p = 3\) worked out earlier has \(h = 1\) and \(l = 6\). When \(p = 9\) there are 126 \(u\)'s of weight four and 126 of weight five; we can take \(h = 3\), \(l = 260\), starting with any four words \((u_1, \ldots, u_4)\) of weight three, then \((u_2, \ldots, u_{130})\) of weight four, then \((u_{131}, \ldots, u_{256})\) of weight five, and \((u_{257}, \ldots, u_{300})\) of weight six. In this case \(n = 256\), \(M = \lfloor 265/2 \rfloor - 3 = 129\), \(m = \lfloor 256/2 \rfloor - 6 - 126\); hence \(M + m - n - 1\) and we have achieved our objective. It is not difficult to verify that the method works for all \(p \geq 3\): when \(p\) is odd, \(h\) will be odd, and we will have \(M = (n + h - 1)/2\), \(m = (n - h - 1)/2\), but when \(p\) is even, \(h\) will be even and we will have \(M = (n + h)/2\), \(m = (n - h)/2\).

The method just described does not depend in any essential way on the assumption that \(n\) is a power of two. We can use it, in fact, to transmit as many as \(2^p - p - 1\) information bits if we let \(l = 2^p\).

ACKNOWLEDGMENT

The author wishes to thank an anonymous referee for several penetrating observations that substantially improved this correspondence.

REFERENCES