RAIRO-Inf. Theor. Appl. **39** (2005) 263-278
DOI: 10.1051/ita:2005015

PARITY CODES [*]

Paulo E. D. Pinto[1], Fábio Protti[2] and
Jayme L. Szwarcfiter[3]

Abstract. Motivated by a problem posed by Hamming in 1980, we define even codes. They are Huffman type prefix codes with the additional property of being able to detect the occurrence of an odd number of 1-bit errors in the message. We characterize optimal even codes and describe a simple method for constructing the optimal codes. Further, we compare optimal even codes with Huffman codes for equal frequencies. We show that the maximum encoding in an optimal even code is at most two bits larger than the maximum encoding in a Huffman tree. Moreover, it is always possible to choose an optimal even code such that this difference drops to 1 bit. We compare average sizes and show that the average size of an encoding in a optimal even tree is at least 1/3 and at most 1/2 of a bit larger than that of a Huffman tree. These values represent the overhead in the encoding sizes for having the ability to detect an odd number of errors in the message. Finally, we discuss the case of arbitrary frequencies and describe some results for this situation.

Mathematics Subject Classification. 94B35, 94B65.

1. Introduction

Huffman codes [4] are among the most traditional methods of encoding. One of the important aspects of these codes is the possibility of handling encodings of variable sizes. In the literature, we can find a great number of extensions and

[*] *Partially supported by the Conselho Nacional de Desenvolvimento Científico e Tecnológico – CNPq, and Fundação de Amparo à Pesquisa do Estado do Rio de Janeiro – FAPERJ, Brasil.*

[1] Universidade Estadual do Rio de Janeiro, Instituto de Matemática e Estatística, RJ, Brasil; `pauloedp@ime.uerj.br`

[2] Universidade Federal do Rio de Janeiro, Instituto de Matemática and NCE, Caixa Postal 2324, 20001-970, Rio de Janeiro, RJ, Brasil; `fabiop@nce.ufrj.br`

[3] Universidade Federal do Rio de Janeiro, Instituto de Matemática, NCE and COPPE, Caixa Postal 2324, 20001-970, Rio de Janeiro, RJ, Brasil; `jayme@nce.ufrj.br`

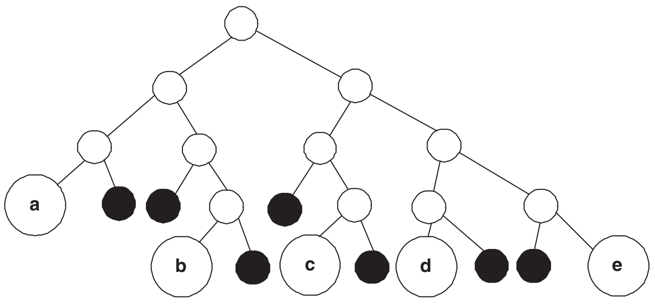

Figure 1. A Hamming-Huffman tree.

variations of the classical Huffman code. For instance, Faller [1], Gallager [2], Knuth [6] and Milidiú, Laber and Pessoa [9] addressed adaptive methods for the construction of Huffman trees. Huffman trees with minimum height were described by Schwartz [14]. The construction of Huffman type trees with length constraints was considered by Turpin and Moffat [12], Larmore and Hirschberg [8] and Milidiú and Laber [7, 10, 11]. On the other hand, Hamming formulated methods for the construction of error detecting codes [3]. Further, Hamming [3] posed the problem of describing a method that combines the advantages of Huffman codes with the noise protection of Hamming codes. The idea is to define a prefix code in which the encoding would contain redundancies that allow the detection of certain kinds of errors. This is equivalent to forbid some encodings which, when present in the reception, would signal an error. Such a code is called a *Hamming-Huffman code* and its representing binary tree a *Hamming-Huffman tree.* In a Huffman tree, all leaves correspond to encodings. In a Hamming-Huffman tree, there are encoding leaves and error leaves. Hitting an error leaf in the decoding process indicates the existence of an error. The problem posed by Hamming is to detect the occurrence of an error of one bit, as illustrated in the following example in [3], p. 76. Table 1 shows the symbols and their corresponding encodings. Figure 1 depicts the corresponding Hamming-Huffman tree. Error leaves are represented by black nodes. An error of one bit in the above encodings would lead to an error leaf.

Table 1. Example of a Hamming-Huffman Code.

Symbol	Encoding
a	000
b	0110
c	1010
d	1100
e	1111

Hamming-Huffman codes have not received further developments to our knowledge, and remained open. Motivated by the above problem, we propose a code, called *even code*, that is able to detect the occurrence of an odd number of 1-bit errors in the message. The detection may occur at the current symbol, or it may be delayed.

The plan of the paper is as follows. Section 2 defines even codes. Section 3 presents a method for finding optimal even codes, while a characterization of them is described in Section 4. Section 5 is dedicated to comparing classical Huffman codes for equal frequencies with optimal even codes. We show that the maximum size of an encoding in an optimal even code is at most two bits larger than the corresponding encoding given by a Huffman code. Moreover, it is always possible to choose an optimal even code in which this difference is reduced to one bit. We compare average sizes and prove that the difference between the average encoding size of an optimal even code and the corresponding average encoding size in a Huffman code lies between 1/3 and 1/2 of a bit. These values represent the overhead in terms of encoding sizes of providing some error detection capability to a Huffman code. Finally, in Section 6, we discuss the case of arbitrary frequencies, and present some results for this situation.

The following definitions are of interest.

Let $\mathscr{S}$ be a set of elements, called *symbols.* An *encoding* e for a symbol $s \in \mathscr{S}$ is a finite sequence of 0's and 1's, associated to s. Each 0 and 1 is a *bit* of e. The bits of the encoding e are labeled $1, 2, \ldots, l$, and l is the *size* of e. A *segment* $e(i, j)$ is the subsequence of e, starting at i and ending at j. A segment $e(1, j)$ is a *prefix* of e. The *parity* of e is the parity of the quantity of 1's contained in e. The set of encodings for all symbols of $\mathscr{S}$ is a *code* $\mathscr{C}$ for $\mathscr{S}$. Two codes $\mathscr{C}$, $\mathscr{C}'$ for $\mathscr{S}$ are *equivalent* when there exists a bijection from $\mathscr{C}$ to $\mathscr{C}'$ such that corresponding encodings have the same size. The *cost* of a code is the sum of the sizes of its encodings. A code is *optimal* when its cost is minimum. A code in which every encoding does not coincide with a prefix of any other encoding is a *prefix code.*

A *message* M is a sequence of symbols. The *encoded message* of M is the corresponding sequence of encodings. The *parity* of an encoded message is the number of 1's contained in the message.

A *binary tree* is a rooted tree T in which every node z, other than the root, is labeled *left* or *right* in such a way that any two siblings have different labels. A *path* in T is a sequence of nodes $z_1, \ldots, z_t$, such that z_q is the parent of z_{q+1}. The value $t-1$ is the *size* of the path, whereas all the z_i's are *descendants* of z_1. If z_1 is the root then $z_1, \ldots, z_t$ is a *root path* and, in addition, if z_t is a leaf, then $z_1, \ldots, z_t$ is a *root-leaf path* of T. The *depth* of a node is the size of the root path to it, while the *depth* of the tree is the largest depth among the nodes. For a node z of T, $T(z)$ denotes the *subtree of* T *rooted at* z, that is, the binary tree containing all descendants of z in T, including itself. The *left subtree* of z is the subtree $T(z')$, where z' is the left child of z. Similarly, define the *right subtree* of z. The left and right subtrees of the root of T are denoted by T_L and T_R, respectively. A *strictly binary tree* is one in which every node is a leaf or has two children. The edges of T leading to left children are labeled 0, whereas those leading to right children are labeled 1. The *parity* of a node z is the parity of the quantity of 1's among the edges forming the root path to z. A node is *even* or *odd* according to

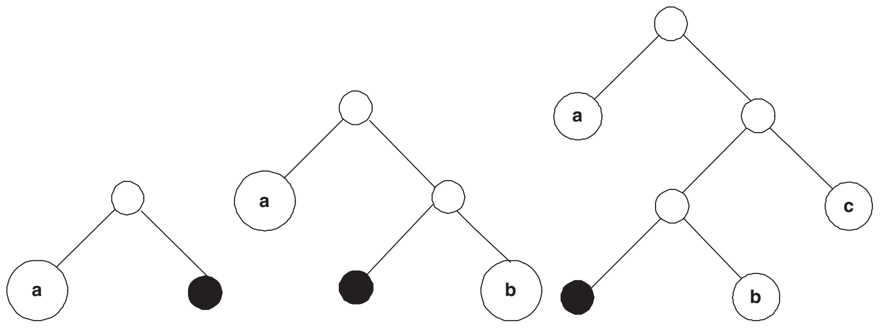

FIGURE 2. Examples of even trees.

its parity. The *path length* $P_T(n)$ of a tree T with n leaves is the sum of the sizes of all root-leaf paths of T. When there is no ambiguity we write simply $P(n)$.

A *(binary tree) representation* of a code $\mathscr{C}$ is a binary tree T such that there exists a one-to-one correspondence between encodings $e \in \mathscr{C}$ and root-leaf paths p of T in such a way that e is precisely the sequence of labels, 0 or 1, of the edges forming p. A code admits a binary tree representation if and only if it is a prefix code. A *full representation tree* of $\mathscr{C}$ is a binary tree T^* obtained from the representation tree T of $\mathscr{C}$, by adding a new leaf as the second child of every node having exactly one child. The original leaves of T are the *encoding leaves*, whereas the new introduced leaves are the *error leaves*. Clearly, in case of Huffman trees, there are no error leaves.

2. Even codes

In this section, we describe even codes and show how they can detect an odd number of errors in a message.

A *parity code* is a prefix code in which all encodings of the same size have identical parities. An *even code* is a code in which all encodings are even, and an *even tree* is a tree representation of an even code. Similarly, a code in which all encodings are odd is an *odd code* and its tree representation is an *odd tree*. For a code $\mathscr{C}$, the symmetric of $\mathscr{C}$ is the code obtained from $\mathscr{C}$ by changing the first bit in every encoding. Clearly, a code is even if and only if its symmetric code is odd.

Figure 2 illustrates examples of even trees.

The next proposition relates parity codes and even codes.

Theorem 1. *Let $\mathscr{C}$ be a parity code for a set of symbols $\mathscr{S}$. Then there exists an even code $\mathscr{C}'$ for $\mathscr{S}$ such that $\mathscr{C}$,$\mathscr{C}'$ are equivalent.*

Proof. Let $\mathscr{C}$ be a parity code for $\mathscr{S}$. Construct $\mathscr{C}'$ from $\mathscr{C}$, as follows. Order the encodings of $\mathscr{C}$ according to their sizes and perform the following procedure: Let $d_1, d_2 \ldots, d_k$ be the distinct encoding sizes appearing in $\mathscr{C}$. For $i = 1, \ldots, k$, take the encodings with size d_i and, in case that those encodings have odd parity, change the ith bit in all encodings with size equal to or greater than d_i. Let $\mathscr{C}'$ be the code obtained after the kth iteration of the procedure. It is clear that all encodings of $\mathscr{C}'$ are even. In addition, since $\mathscr{C}$ is a prefix code, $\mathscr{C}'$ is also a prefix code. In order to verify the latter assertion, let e', e'' be two encodings of $\mathscr{C}$, $|e'| \leq |e''|$. Then whenever bit j of e' changes in the above procedure, bit j of e'' also changes. Consequently, $e' \neq e''(1, |e'|)$, implying that the corresponding encodings of $\mathscr{C}'$ are also distinct. That is, $\mathscr{C}'$ is indeed a prefix code, and consequently an even code. □

The above proposition implies that, for our purposes, we can restrict attention to even codes, while handling the larger class of parity codes.

It is easy to conclude that even codes detect the occurrence of an odd number of 1-bit errors in a message. We know that all encodings are even, so the encoded message is also even. By introducing an odd number of errors, the encoded message becomes odd. Since the encodings are even, in the full tree representation of the code, an error leaf would be hit during the decoding process, or otherwise it terminates at some odd node of the tree. It is worth noting that odd codes do not have this property. For example, consider the code $\mathscr{C} = \{1, 01\}$ and the message 01; if the first bit is changed, resulting 11, that wrong message would be decoded with no error detection.

The tree representation of an even code is an *even tree*. A code in which all encodings are odd is an *odd code* and its tree representation an *odd tree*. For a code $\mathscr{C}$, the symmetric of $\mathscr{C}$ is the code obtained from $\mathscr{C}$ by changing the first bit in every encoding. Clearly, a code is even if and only if its symmetric code is odd.

3. Optimal even codes

In this section, we describe a method for finding optimal even codes. The cost of an optimal even code $\mathscr{C}$ for n symbols is the minimal path length $P(n)$ among the corresponding even trees. The following theorem determines $P(n)$.

Theorem 2.

$$P(n) = \begin{cases} 1, & \textit{if } n = 1 \\ 3, & \textit{if } n = 2 \\ n + \min\{P(n-1), \min\limits_{1<i<n-1}\{P(i) + P(n-i)\}\}, & \textit{if } n > 2. \end{cases}$$

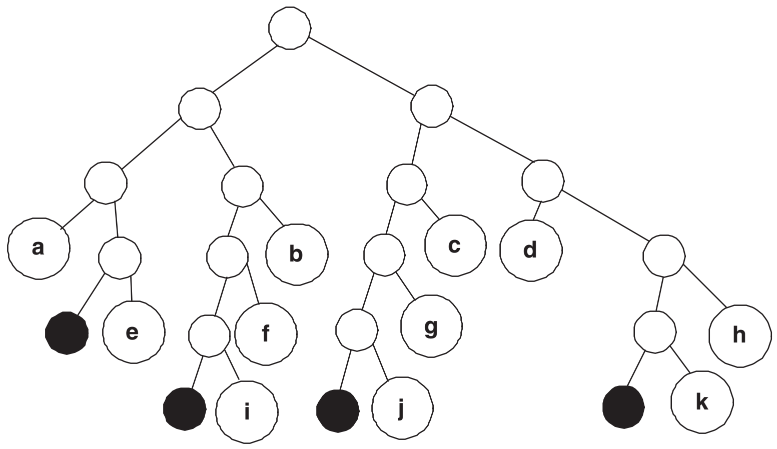

FIGURE 3. Equilibrated even tree for 11 symbols.

Proof. The results for $n = 1$ and $n = 2$ can be verified in Figure 2. If $n > 2$, the external minimization in the expression is taken over two cases:

Case 1. The left subtree has only one encoding. Then in this case we have $P(n) = 1 + (n - 1) + P(n - 1) = n + P(n - 1)$, since at the left side there is only one encoding, and at the right side there is also an optimal odd tree for $n - 1$ symbols. Observe now that all the encodings are one level deeper relatively to the root.

Case 2. The left subtree has more than one encoding. Let i be the number of encodings of the left subtree, which is an optimal even tree, $1 < i < n-1$. The right subtree is an optimal odd tree with $n - i$ encodings. All the encodings are one level deeper relatively to the root. The least possible value for $P(n)$ is then given by the internal minimization: $\min_{1<i<n-1}\{i + P(i) + (n - i) + P(n - i)\} = n + \min_{1<i<n-1}\{P(i) + P(n - i)\}$. □

Next, we define inductively a useful family of codes *via* its tree representation. An *equilibrated even(odd) tree* T for n symbols is an even(odd) tree T such that:

- if $n \leq 3$, the equilibrated even trees are those of Figure 2, while the odd ones are the symmetric of these even trees;
- if $n > 3$, the left subtree of T is an equilibrated even tree for $\lfloor n/2 \rfloor$ symbols, while the right subtree is an equilibrated odd tree for $\lceil n/2 \rceil$ symbols.

An example of an equilibrated even tree for 11 symbols is given in Figure 3.

We will prove that equilibrated even trees are optimal. First, we compute the path length of an equilibrated tree.

Theorem 3. *Let $P(n)$ be the path length of an equilibrated even tree for n symbols. Then:*

$$P(n) = \begin{cases} 1, & \text{if } n = 1 \\ 3, & \text{if } n = 2 \\ n\left(\left\lceil \log \frac{n}{3} \right\rceil + 3\right) - 3.2^{\lceil \log \frac{n}{3} \rceil}, & \text{if } n > 2. \end{cases}$$

Proof. If $n \le 5$, we can verify by inspection that the theorem is correct. For $n > 5$, use induction. Assume that the above expression is correct for $2 < j < n$. Applying the definition and the induction hypothesis, we have:

$$P(n) = n + P(\lfloor n/2 \rfloor) + P(\lceil n/2 \rceil)$$

where

$$P(\lfloor n/2 \rfloor) = \lfloor n/2 \rfloor \left(\left\lceil \log \frac{\lfloor n/2 \rfloor}{3} \right\rceil + 3 \right) - 3.2^{\lceil \log \frac{\lfloor n/2 \rfloor}{3} \rceil}$$

and

$$P(\lceil n/2 \rceil) = \lceil n/2 \rceil \left(\left\lceil \log \frac{\lceil n/2 \rceil}{3} \right\rceil + 3 \right) - 3.2^{\lceil \log \frac{\lceil n/2 \rceil}{3} \rceil}.$$

We then have two cases:

Case 1. n is of the form $n = 3.2^{q-1} + 1$. Then:

$$\left\lceil \log \frac{n}{3} \right\rceil = q, \left\lceil \log \frac{\lceil n/2 \rceil}{3} \right\rceil = q - 1, \left\lceil \log \frac{\lfloor n/2 \rfloor}{3} \right\rceil = q - 2, \lfloor n/2 \rfloor = 3.2^{q-2}.$$

Hence:

$$P(n) = n + \lfloor n/2 \rfloor (q - 2 + 3) - 3.2^{q-2} + \lceil n/2 \rceil (q - 1 + 3) - 3.2^{q-1}.$$

That is,

$$P(n) = n\left(\left\lceil log \frac{n}{3} \right\rceil + 3\right) - 3.2^{\lceil log \frac{n}{3} \rceil}.$$

Case 2. n is of the form $n = 3.2^q$ or $n = 3.2^{q-1} + p,\ 1 < p < 3.2^{q-1}$. Then:

$$\left\lceil \log \frac{n}{3} \right\rceil = q, \left\lceil \log \frac{\lceil n/2 \rceil}{3} \right\rceil = \left\lceil \log \frac{\lfloor n/2 \rfloor}{3} \right\rceil = q - 1.$$

Hence:

$$P(n) = n + \lfloor n/2 \rfloor (q - 1 + 3) - 3.2^{q-1} + \lceil n/2 \rceil (q - 1 + 3) - 3.2^{q-1}.$$

That is,

$$P(n) = n\left(\left\lceil \log \frac{n}{3} \right\rceil + 3\right) - 3.2^{\lceil \log \frac{n}{3} \rceil}.$$

In both cases, the result is still valid for n. □

Finally, we have:

Theorem 4. *Equilibrated even trees are optimal.*

Proof. Let T be an equilibrated even tree for n symbols. For $n \leq 6$, we can verify by inspection that the theorem is correct. For $n > 6$ use induction. From Theorem 3, $P(n) = n(\lceil \log \frac{n}{3} \rceil + 3) - 3.2^{\lceil \log \frac{n}{3} \rceil}$. The induction hypothesis assures that equilibrated even trees for less than n symbols are optimal. For the induction step, let T' be an optimal even tree for $n > 6$ symbols. As the subtrees are also optimal, by the induction hypothesis we can replace the subtrees T'_L and T'_R of T by equilibrated trees, while maintaining T' as optimal. Let i be the number of symbols on the left subtree of T'. Apply Theorem 2.

Let us prove that $i > 2$. Assume first that $i = 1$. Then:

$$P_{T'}(n) = n + (n-1)\left(\left\lceil \log \frac{n-1}{3} \right\rceil + 3\right) - 3.2^{\lceil \frac{n-1}{3} \rceil}.$$

Consider the following two cases:

Case 1. If $\lceil \log \frac{n}{3} \rceil = \lceil \log \frac{n-1}{3} \rceil = q$, then $P_{T'}(n) = P_T(n) + (n-q) - 3$. Since in this case $(n-q) > 3$, we have $P_{T'}(n) > P_T(n)$, a contradiction.

Case 2. If $\lceil \log \frac{n}{3} \rceil = q$ and $\lceil \log \frac{n-1}{3} \rceil = q-1$, then $P_{T'}(n) = P_T(n) + 3.2^{q-1} - (q+2)$. Since in this case $q > 1$ implies $3.2^{q-1} > (q+2)$, we have $P_{T'}(n) > P_T(n)$, a contradiction again. Consequently $i \neq 1$.

Assume now that $i = 2$. Then:

$$P_{T'}(n) = n + 3 + (n-2)\left(\left\lceil \log \frac{n-2}{3} \right\rceil + 3\right) - 3.2^{\lceil \frac{n-2}{3} \rceil}.$$

We examine again the possible cases:

Case 1'. If $\lceil \log \frac{n}{3} \rceil = \lceil \log \frac{n-2}{3} \rceil = q$, then $P_{T'}(n) = P_T(n) + n - (2q+3)$. Clearly, this can only occur for $n > 7$. Thus $n > (2q+3)$ and $P_{T'}(n) > P_T(n)$, a contradiction.

Case 2'. If $\lceil \log \frac{n}{3} \rceil = q$ and $\lceil \log \frac{n-2}{3} \rceil = q-1$, then $P_{T'}(n) = P_T(n) + 3.2^{q-1} - (2q+1)$. Since in this case $q > 1$ implies $3.2^{q-1} > 2q+1$, we have $P_{T'}(n) > P_T(n)$, again a contradiction. Consequently, $i \neq 2$.

From now on, we know that $i > 2$. Now we will write $P_{T'}(n)$ in terms of a function f defined in the interval $[3, \lfloor \frac{n}{2} \rfloor]$ and and prove that $f(i) \geq f(i+1)$ for

$i \in [3, \lfloor \frac{n}{2} \rfloor - 1]$. Consequently, $f(i)$ attains the minimum at $i = \lfloor \frac{n}{2} \rfloor$, and T' will be an optimal even tree. This means that T' will be equilibrated and equivalent to T. Hence T must also be optimal. We have:

$$f(i) = n + i\left(\left\lceil \log \frac{i}{3} \right\rceil + 3\right) - 3.2^{\lceil \log \frac{i}{3} \rceil} + (n-i)\left(\left\lceil \log \frac{n-i}{3} \right\rceil + 3\right) - 3.2^{\lceil \log \frac{n-i}{3} \rceil}.$$

By taking two consecutive values for i,

$$\begin{aligned} f(i) - f(i+1) = n + i & \left(\left\lceil \log \frac{i}{3} \right\rceil + 3\right) - 3.2^{\lceil \log \frac{i}{3} \rceil} \\ & +(n-i)\left(\left\lceil \log \frac{n-i}{3} \right\rceil + 3\right) - 3.2^{\lceil \log \frac{n-i}{3} \rceil} \\ & -\left(n + (i+1)\left(\left\lceil \log \frac{i+1}{3} \right\rceil + 3\right) - 3.2^{\lceil \log \frac{i+1}{3} \rceil}\right. \\ & \left. +(n-i-1)\left(\left\lceil \log \frac{n-i-1}{3} \right\rceil + 3\right) - 3.2^{\lceil \log \frac{n-i-1}{3} \rceil}\right). \end{aligned}$$

Let $\lceil \log \frac{i}{3} \rceil = q_1$ and $\lceil \log \frac{n-i}{3} \rceil = q_2$. There are now four cases, because it is possible to have $\lceil \log \frac{i+1}{3} \rceil = q_1$ or $\lceil \log \frac{i+1}{3} \rceil = q_1 + 1$, and $\lceil \log \frac{n-i-1}{3} \rceil = q_2$ or $\lceil \log \frac{n-i-1}{3} \rceil = q_2 - 1$. We should notice that the number of symbols in the left subtree is always supposed to be not greater than that in the right subtree, that is, $q_2 \geq q_1$.

Case I. If $\lceil \log \frac{i+1}{3} \rceil = q_1$ and $\lceil \log \frac{n-i-1}{3} \rceil = q_2$, then $f(i) - f(i+1) = q_2 - q_1 \geq 0$.

Case II. If $\lceil \log \frac{i+1}{3} \rceil = q_1$ and $\lceil \log \frac{n-i-1}{3} \rceil = q_2 - 1$, then $f(i) - f(i+1) = (n-i-1) - 3.2^{q_2-1} + q_2 - q_1$. The equalities $\lceil \log \frac{n-i}{3} \rceil = q_2$ and $\lceil \log \frac{n-i-1}{3} \rceil = q_2 - 1$ hold only when $n - i = 3.2^{q_2-1} + 1$. Then $f(i) - f(i+1) = q_2 - q_1 \geq 0$.

Case III. If $\lceil \log \frac{i+1}{3} \rceil = q_1 + 1$ and $\lceil \log \frac{n-i-1}{3} \rceil = q_2$, then $f(i) - f(i+1) = -i + 3.2^{q_1} + q_2 - q_1 - 1$. The equalities $\lceil \log \frac{i}{3} \rceil = q_1$ and $\lceil \log \frac{i+1}{3} \rceil = q_1 + 1$ hold only when $i = 3.2^{q_1}$. Then $f(i) - f(i+1) = q_2 - q_1 - 1 \geq 0$, since in this case $q_2 > q_1$.

Case IV. If $\lceil \log \frac{i+1}{3} \rceil = q_1 + 1$ and $\lceil \log \frac{n-i-1}{3} \rceil = q_2 - 1$, then $f(i) - f(i+1) = -2i + n + 3.2^{q_1} - 3.2^{q_2-1} + q_2 - q_1 - 2$. The equalities $\lceil \log \frac{i}{3} \rceil = q_1$ and $\lceil \log \frac{i+1}{3} \rceil = q_1 + 1$ hold only when $i = 3.2^{q_1}$. Also, the equalities $\lceil \log \frac{n-i}{3} \rceil = q_2$ and $\lceil \log \frac{n-i-1}{3} \rceil = q_2 - 1$ hold only when $n - i = 3.2^{q_2-1} + 1$. Then $f(i) - f(i+1) = q_2 - q_1 - 1 \geq 0$, since in this case $q_2 > q_1 + 1$.

In all the above cases, $f(i) \geq f(i+1)$ for $i \in [3, \lfloor \frac{n}{2} \rfloor - 1]$. Hence, the minimum is attained at $i = \lfloor \frac{n}{2} \rfloor$, and the even treewith $\lfloor \frac{n}{2} \rfloor$ symbols in the left subtree is optimal and equilibrated. Consequently, the hypothesis is also valid for n symbols and the induction is complete. $\square$

4. Characterization of optimal even trees

In Section 3, we defined equilibrated even trees and showed that they are optimal. However, there are optimal even trees that are not equilibrated. In this section, we characterize all the optimal even trees.

Let T be a binary tree. Denote by L_T the number of leaves in T.

Theorem 5. *An even tree with $n > 5$ symbols is optimal if and only if*

(i) T_L is an optimal even tree and T_R is an optimal odd tree, and

(ii) L_{T_L} or L_{T_R} is an integer in the interval $[i_{\min}, \lfloor n/2 \rfloor]$, where

$$i_{\min} = \begin{cases} 3.2^{\lfloor \log \frac{n}{3} \rfloor - 1} & \textit{if } n \leq 9.2^{\lfloor \log \frac{n}{3} \rfloor - 1} \\ n - 3.2^{\lfloor \log \frac{n}{3} \rfloor} & \textit{otherwise.} \end{cases}$$

Proof. Let T be an optimal even tree with n symbols. We will show that T satisfies (i) and (ii). If (i) is not satisfied then replacing the left or right subtree of T, respectively, by an optimal even or odd tree with the same number of symbols would reduce the cost of the tree, a contradiction.

Now we will prove that (ii) must hold, using Theorem 4. Without loss of generality, suppose $L_{T_L} \leq L_{T_R}$. We can write $n = 3.2^{q-1} + p$, p and q integers, $1 \leq p \leq 3.2^{q-1}$. Let us call $\lceil \log \frac{i}{3} \rceil = q_1$, $\lceil \log \frac{n-i}{3} \rceil = q_2$ and $\lceil \log \frac{n}{3} \rceil = q$. The function $f(i) = P_T(n)$ is non-increasing in the interval $[3, \lfloor \frac{n}{2} \rfloor]$ and has a minimum at $i = \lfloor \frac{n}{2} \rfloor$. Consequently, there exists exactly one minimum value $i_{\min}$ for which $f(i)$ remains constant in the interval $[i_{\min}, \lfloor \frac{n}{2} \rfloor]$. For each value i in this interval we have a distinct optimal even tree T such that T_L has i leaves. Let us determine $i_{\min}$. There are two possible starting points from where $f(i)$ remains constant, depending on certain conditions involving q_1 and q_2.

Case 1. $q_1 = q_2$. Then $\lceil \log \frac{i_{\min}}{3} \rceil = q_1 = \lceil \log \frac{\lfloor \frac{n}{2} \rfloor}{3} \rceil = \lceil \log \frac{n-i}{3} \rceil = q_2$. Now we have two subcases:

Case 1.1. n is of the form $n = 3.2^{q-1} + 1$. Hence $q_1 = q_2 = q - 2$. In this case there is only one possible value for $i_{\min}$:

$$i_{\min} = \lfloor n/2 \rfloor = 3.2^{q-2}. \tag{1}$$

Case 1.2. n is not of the form $n = 3.2^{q-1} + 1$. Hence $q_1 = q_2 = q - 1$, and $i_{\min}$ should satisfy two inequalities:

$$3.2^{q-2} + 1 \leq i_{\min} \leq \lfloor n/2 \rfloor \tag{2}$$

$$n - i_{\min} \leq 3.2^{q-1}. \tag{3}$$

From (2) and (3), we have:

$$\max\{3.2^{q-2} + 1, n - 3.2^{q-1}\} \leq i_{\min} \leq \lfloor n/2 \rfloor. \tag{4}$$

Case 2. $q_2 = q_1 + 1$. In this case, $i_{\min}$ is of the form $i_{\min} = 3.2^{q_1}$. We have two subcases:

Case 2.1. n is of the form $n = 3.2^q$. Let us check the possible values for q_1. We know that $q_1 < q$. We cannot have $q_1 = q - 1$ because this would imply $n - i_{\min} = 3.2^{q-1}$ and $\lceil \log \frac{n-i_{\min}}{3} \rceil = q_2 = q - 1 = q_1$. Also, we cannot have $q_1 < q - 1$ because this would imply $n - i_{\min} > 3.2^{q-1}$ and $\lceil \log \frac{n-i_{\min}}{3} \rceil = q_2 = q$. Hence $q_2 > q_1 + 1$. We do not have possible values for q_1. Consequently, n cannot be of the form $n = 3.2^q$.

Case 2.2. n is of the form $n = 3.2^{q-1} + p$, $1 \leq p < 3.2^{q-1}$. Let us check the possible values for q_1. We cannot have $q_1 = q$ because this would imply $n - 3.2^{q-1} = p$ and $\lceil \log \frac{p}{3} \rceil = q_2 < q - 1 = q_1$. We cannot have $q_1 < q - 2$ because this would imply that $q_2 \geq q - 1$ and hence $q_2 > q_1 + 1$. But we can have $q_1 = q - 2$, the only possible value for q_1. In this case, $n - i_{\min} = 3.2^{q-2} + p$ and $q_2 = \lceil \log \frac{3.2^{q-2}+p}{3} \rceil = q - 1 = q_1 + 1$. Consequently, we can state the two relations below:

$$3.2^{q-2} = i_{\min} < \lfloor n/2 \rfloor \tag{5}$$

$$n - i_{\min} \leq 3.2^{q-1}. \tag{6}$$

From (5) and (6), we have:

$$n \leq 9.2^{q-2}. \tag{7}$$

We have determined the conditions that $i_{\min}$ should satisfy for the existence of an optimal even tree having $i_{\min}$ leaves in its left subtree. These conditions lead to the determination of the exact values of $i_{\min}$ as follows. Now, consider the possible values for n:

Case I. $n = 3.2^q$. Then $n = 6.2^{q-1} < 9.2^{q-1} = 9.2^{\lfloor \log \frac{n}{3} \rfloor - 1}$. This situation corresponds to case 1.2. By applying 4, $i_{\min} = 3.2^{q-1} = \lfloor n/2 \rfloor$. Thus there is only one element in the interval. We can write $i_{\min} = 3.2^{\lfloor \log \frac{n}{3} \rfloor - 1}$, which corresponds to the first situation of the theorem.

Case II. $n = 3.2^{q-1} + 1$. Then $n = 6.2^{q-1} + 1 < 9.2^{q-1} = 9.2^{\lfloor \log \frac{n}{3} \rfloor - 1}$. This situation corresponds to case 1.1. By applying 1, $i_{\min} = 3.2^{q-2}$. Thus there is also only one element in the interval. We can write $i_{\min} = 3.2^{\lfloor \log \frac{n}{3} \rfloor - 1}$, which also corresponds to the first situation of the theorem.

Case III. $n \leq 9.2^{\lfloor \log \frac{n}{3} \rfloor - 1}$ and cases I and II do not occur. This is equivalent to say that n is of the form $n = 3.2^{q-1} + p$, $2 \leq p \leq 3.2^{q-2}$. The smallest value for $i_{\min}$ occurs by applying 5, because if we apply 4 we would have a worst value $i_{\min} \geq 3.2^{q-2} + 1$. We can rewrite this condition as $i_{\min} = 3.2^{\lfloor \log \frac{n}{3} \rfloor - 1}$, because here we have $\lfloor \log \frac{n}{3} \rfloor = q - 1 = \lceil \log \frac{n}{3} \rceil - 1$. This case concludes the first situation of the theorem.

Case IV. $n > 9.2^{\lfloor \log \frac{n}{3} \rfloor - 1}$ and cases I–III do not occur. In this situation we must have $n = 3.2^{q-1} + p$, $3.2^{q-2} < p < 3.2^{q-1}$, since $9.2^{q-2} = 3.2^{q-1} + 3.2^{q-2}$ implies $\lfloor \log \frac{n}{3} \rfloor = q - 1 = \lceil \log \frac{n}{3} \rceil - 1$. Hence, this is equivalent to $n > 9.2^{\lceil \log \frac{n}{3} \rceil - 1}$. There is only one possible way to satisfy 4 in case 1: $\lfloor \frac{n}{2} \rfloor \geq i_{\min} \geq \max\{3.2^{q-2} + 1, n - 3.2^{q-1}\}$. But we have $n - 3.2^{q-1} > 9.2^{q-2} - 3.2^{q-1} = 3.2^{q-2}$. Then $i_{\min} = n - 3.2^{q-1}$, that is, $i_{\min} = n - 3.2^{\lfloor \log \frac{n}{3} \rfloor}$, corresponding to the second situation of the theorem. This completes the proof of the necessary condition.

For the sufficiency, suppose that T is an even tree satisfying (i) and (ii). The proof of the necessity and Theorem 4 imply that an optimal even tree satisfies (i) and (ii). Moreover, there exists an optimal tree T' satisfying (i), (ii) and also $L_{T'} = L_T$. Consequently, T' and T have the same cost, meaning that T is indeed optimal. $\square$

The above theorem provides a direct recognition of optimal even trees. It also describes a method for generating all optimal trees for a given set of symbols.

5. Optimal even trees and Huffman trees

In this section, we compare optimal even trees with Huffman trees. Here, Huffman trees correspond to complete strictly binary trees. First, we compare the sizes of the maximum encodings of these trees. Finally, we compare the average sizes of an encoding in the trees.

Let us determine the minimum and maximum depths of optimal even trees with n symbols. These parameters can be obtained from the following observations:

- Optimal even trees with minimum depth can be recursively constructed by using left subtrees with maximum number of symbols, provided that this number is not greater than the number of symbols in the right subtree. Consequently, the left subtree contains $\lfloor \frac{n}{2} \rfloor$ symbols. As discussed before, these are the equilibrated even trees.
- An optimal even tree T with maximum depth can also be recursively constructed by using left subtrees with minimum number of symbols. That is, the left subtree of T has $i_{\min}$ leaves, where $i_{\min}$ is given by Theorem 5.

Let $D_{\min}(n)$ and $D_{\max}(n)$ be respectively the minimum and maximum depth of an optimal even tree for n symbols. Then:

$$D_{\min} = \begin{cases} 1, & \text{if } n = 1 \\ 1 + D_{\min}(\lceil n/2 \rceil), & \text{if } n > 1 \end{cases}$$

$$D_{\max} = \begin{cases} 1, & \text{if } n = 1 \\ 1 + D_{\max}(n - i_{\min}), & \text{if } n > 1. \end{cases}$$

Solving the above recurrences, we obtain:

$$D_{\min} = \begin{cases} 1, & \text{if } \ n = 1, \\ \lceil \log n \rceil + 1, & \text{if } \ n > 1 \end{cases}$$

$$D_{\max} = \begin{cases} 1, & \text{if } \ n = 1 \\ \lceil \log \frac{n}{3} \rceil + 3, & \text{if } \ n > 1. \end{cases}$$

In Huffman trees, the minimum and the maximum depths coincide. Let H be a Huffman tree for n symbols and D_H its depth. Then $D_H = \lceil \log n \rceil$. By comparing $D_{\min}$ and D_H, we conclude that we can always construct an optimal even tree whose largest encoding is one bit larger than the corresponding one for a Huffman tree. From the values of $D_{\max}$ and D_H, we can additionally conclude that the maximum encoding in an arbitrary optimal even tree is at most two bits larger than that in a Huffman tree.

In the following, we consider average encoding sizes.

Theorem 6. *The difference between the average encoding sizes of an optimal even tree and a Huffman tree, with $n > 1$ symbols, lies in the interval $[1/3, 1/2]$. It is minimum when $n = 3.2^k$, and maximum when $n = 2^k$, for some k.*

Proof. The average encoding size of an optimal even tree (Th. 4) is:

$$\frac{n(\lceil \log \frac{n}{3} \rceil + 3) - 3.2^{\lceil \log \frac{n}{3} \rceil}}{n}$$

while that of an optimal Huffman tree [5] is:

$$\frac{n(\lceil \log n \rceil + 1) - 2^{\lceil \log n \rceil}}{n}.$$

Then the difference is given by:

$$d = \frac{n(\lceil \log \frac{n}{3} \rceil + 3) - 3.2^{\lceil \log \frac{n}{3} \rceil} - (n(\lceil \log n \rceil + 1) - 2^{\lceil \log n \rceil})}{n}.$$

First, we will prove that $d \leq 1/2$. Consider the following two cases:

Case 1. $n = 2^k$, $k \geq 1$. Then $\lceil \log n \rceil = k$ and $\lceil \log \frac{n}{3} \rceil = \lceil \log ((2^{k-2})(\frac{4}{3})) \rceil = \lceil (k-2) + \log \frac{4}{3} \rceil = k - 2 + 1 = k - 1$. We conclude that:

$$d = \frac{2^k(k - 1 + 3) - 3.2^{k-1} - (2^k(k+1) - 2^k)}{2^k} = 1/2.$$

Case 2. $n = 2^k + p$, $1 < p < 2^k$. Then $\lceil \log n \rceil = k + 1$ and $\lceil \log \frac{n}{3} \rceil$ may be equal to $k - 1$ or k. We have two subcases:

Case 2.1. If $\lceil \log \frac{n}{3} \rceil = k - 1$ then $d = \frac{2^{k-1}}{2^k+p}$. By using the fact that $p > 1$, we conclude that $d < 1/2$.

Case 2.2. If $\lceil \log \frac{n}{3} \rceil = k$ then $d = \frac{p}{2^k+p}$. By using the fact that $p < 2^k$, we conclude that $d < 1/2$.
Thus, the maximum difference between the encodings is $1/2$. Now, we show that d is at least $1/3$. Consider two additional cases:

Case 3. $n = 3.2^k$, $k \geq 0$. Then $\lceil \log n \rceil = \lceil k + \log 3 \rceil = (k + 2)$ and $\lceil \log \frac{n}{3} \rceil = \lceil \log 2^k \rceil = k$. Hence:

$$d = \frac{-3.2^k + 2^{k+2}}{3.2^k} = 1/3.$$

Case 4. $n = 3.2^k + p$, $1 < p < 3.2^k$. Then $\lceil \log n \rceil$ may be equal to $(k + 2)$ or $(k + 3)$, and $\lceil \log \frac{n}{3} \rceil = k + 1$. Consider two additional subcases:

Case 4.1. If $\lceil \log n \rceil = k + 2$ then $d = \frac{2^k+p}{3.2^k+p}$. Since $p > 1$, we conclude that $d > 1/3$.

Case 4.2. If $\lceil \log n \rceil = k + 3$ then $d = \frac{2.2^k}{3.2^k+p}$. By using the fact that $p < 3.2^k$, we conclude that $d > 1/3$.

Thus, in the two subcases above, the minimum difference between the encodings is $1/3$. The proof is complete. □

The limits $1/3$ and $1/2$ can be explained in terms of the equilibrated even trees, as discussed in the following.

When n is of the form $n = 2^k$, the Huffman tree is a full tree, where all the leaves have the same depth $\lceil \log n \rceil$. On the other hand, the equilibrated even tree is the composition of 2^{k-1} subtrees with two encodings. In this tree, half of the leaves have depth $\lceil \log n \rceil$ and half have depth $\lceil \log n \rceil + 1$. Then, the difference of the average encoding lengths is $1/2$.

When n is of the form $n = 3.2^k$, the Huffman tree is a complete tree, where there are 2^{k+1} leaves at depth $\lceil \log n \rceil$, and 2^k leaves at depth $\lceil \log n \rceil - 1$. On the other hand, the equilibrated even tree is the composition of 2^k subtrees with three encodings. In this case, we have 2^k leaves at depth $\lceil \log n \rceil + 1$, 2^k leaves at depth $\lceil \log n \rceil$, and 2^k leaves at depth $\lceil \log n \rceil - 1$. This situation leads to the difference of $1/3$ in the average encoding lengths.

In the remaining cases for n, the corresponding trees have intermediate configurations between these limits.

6. Even codes for arbitrary frequencies

Here we describe some results for the situation of arbitrary frequencies [13].

We describe an exact algorithm for finding an optimal even code and afterwards an approximation.

Let $\mathscr{S} = \{s_1, \ldots, s_n\}$ be a set of symbols, each s_i having a frequency f_i satisfying $f_i \leq f_{i+1}$. For $m \leq n$, denote $\mathscr{S}_m = \{s_1, \ldots, s_m\}$. The cost of a code $\mathscr{C}$ for $\mathscr{S}$ is $sum_{i=1}^{n} f_i |e_i|$, where e_i is the encoding associated to s_i. Our aim is to find an even code $\mathscr{C}$ for $\mathscr{S}$ with minimum cost. In fact, we propose a solution for a slightly more general problem, employing dynamic programming.

A *parity forest* F for $\mathscr{S}_m$ is a set of p even trees and q odd trees such that their leaves correspond to the symbols of $\mathscr{S}_m$, for $0 \leq p, q \leq m$ $(pq \neq 0)$. Define the *cost* of F as the sum of the costs of its trees. Say that F is (m,p,q)-optimal when its cost is the least among all forests for $\mathscr{S}_m$ having p even trees and q odd trees. Denote by $c(m,p,q)$ the cost of an (m,p,q)-optimal forest. In terms of this notation, the solution of our problem is $c(n,1,0)$. First, define the funtion

$$A_i = \begin{cases} \sum_{j=1}^{i} f_j, & \text{if } i > 0 \\ 0, & \text{otherwise.} \end{cases}$$

The following recurrence defines the computation of $c(m,p,q)$.

Let m, p, q be integers such that $0 \leq m \leq n,\ m \geq p \geq 0$ and $m \geq q \geq 1$. Let $d_{\max} = \lfloor \log \frac{m}{p+q} \rfloor + 1$. Then:

(1) if $m \leq p+q$ then $c(m,p,q) = A_{m-p}$;

(2) if $m > p+q$ and $p = 0$ then $c(m,p,q) = c(m,q,q) + A_m$;

(3) if $m > p+q$ and $p \neq 0$, then $c(m,p,q)$ is equal to the minimum between $c(m-1,p-1,q)$ and $\min_{1 \leq d \leq d_{\max}} \{c(m-1,(p+q)2^d-1,(p+q).2^d)+d.A_m\}$.

The resulting dynamic programming algorithm has time complexity $O(n^3 \log n)$ and space complexity $O(n^3)$. Further, we describe the algorithm for finding nearly optimal even codes. It consists of the following steps:

a) First, obtain a Huffman tree having a small number of odd leaves, if possible. It can be done by slightly modifying the process of construction of Huffman trees, avoiding as much as possible to join two trivial subtrees (with only one node). After the tree is constructed, exchange odd leaves with even internal nodes at the same level, when possible.

b) Next, transform the resulting Huffman tree into an even tree, by forcing odd leaves to become even ones. This is accomplished by moving them one level deeper in the tree and creating a corresponding error sibling leaf.

c) Finally, exchange leaves in such a way that the higher frequencies become closer to the root.

This heuristics is easy to implement and has the same complexity as the Huffman method. It can be proved that the cost of the resulting tree obtained by this heuristics is at most 50% higher than the corresponding Huffman tree, but experimental results suggest a much smaller bound.

7. Conclusion

Even codes have been defined as those whose encodings are all even. A characterization of optimal even codes has been described. It has been shown that an optimal even code for n symbols can always be found, such that the largest encoding size is just one bit larger than that for corresponding Huffman code (when considering equal frequencies). Furthermore, the difference is at most 2 bits, for any optimal even tree. In addition, the average size of an encoding in the optimal even code is at least 1/3 bit larger and at most 1/2 larger than that of an Huffman code. Optimal even codes for n symbols and equal frequencies can be constructed in $O(n)$ time, applying the definition of equilibrated trees in Section 3. On the other hand, optimal codes for arbitrary frequencies can be consctructed in $O(n^3 \log n)$ time by a dynamic programming algorithm, and nearly optimum even codes can be constructed in $O(n \log n)$ time. It is worth mentioning that we are currently working on a more efficient exact algorithm, as well as a tighter bound for the approximation.

References

[1] N. Faller, An adaptive Method for Data Compression, in *Record of the 7th Asilomar Conference on Circuits, Systems and Computers*, Naval Postgraduate School, Monterrey, Ca. (1973) 593–597.

[2] R.G. Gallager, Variations on a Theme by Huffman. *IEEE Trans. Inform. Theory* **24** (1978) 668–674.

[3] R.W. Hamming, *Coding And Information Theory.* Prentice Hall (1980).

[4] D.A. Huffman, *A Method for the Construction of Minimum Redundancy Codes,* in *Proc. of the IRE* **40** (1951) 1098–1101.

[5] D.E. Knuth, *The Art of Computer Programming.* Addison Wesley (1973).

[6] D.E. Knuth, Dynamic Huffman Coding. *J. Algorithms* **6** (1985) 163–180.

[7] E.S. Laber, *Um algoritmo eficiente para construção de códigos de prefixo com restrição de comprimento.* Master Thesis, PUC-RJ, Rio de Janeiro (1997).

[8] L.L. Larmore and D.S. Hirshberg, A fast algorithm for optimal length-limited Huffman codes. *JACM* **37** (1990) 464–473.

[9] R.L. Milidiu, E.S. Laber and A.A. Pessoa, Improved Analysis of the FGK Algorithm. *J. Algorithms* **28** (1999) 195–211.

[10] R.L. Milidiu and E.S. Laber, The Warm-up Algorithm: A Lagrangean Construction of Length Restricted Huffman Codes. *SIAM J. Comput.* **30** (2000) 1405–1426.

[11] R.L. Milidiu and E.S. Laber, Improved Bounds on the Inefficiency of Length Restricted Codes. *Algorithmica* **31** (2001) 513–529.

[12] A. Turpin and A. Moffat, Practical length-limited coding for large alphabets. *Comput. J.* **38** (1995) 339–347.

[13] P.E.D Pinto, F. Protti and J.L. Szwarcfiter, A Huffman-Based Error Detection Code, in *Proc. of the Third International Workshop on Experimental and Efficient Algorithms (WEA 2004)*, Angra dos Reis, Brazil, 2004. *Lect. Notes Comput. Sci.* **3059** (2004) 446–457.

[14] E.S. Schwartz, An Optimum Encoding with Minimal Longest Code and Total Number of Digits. *Inform. Control* **7** (1964) 37–44.