Estimating the Size of Huffman Code Preambles

R. J. McEliece
Communications Systems Research Section

T. H. Palmatier
California Institute of Technology

In this article, data compression via block-adaptive Huffman coding is considered. The compressor consecutively processes blocks of N *data symbols, estimates source statistics by computing the relative frequencies of each source symbol in the block, and then synthesizes a Huffman code based on these estimates. In order to let the decompressor know which Huffman code is being used, the compressor must begin the transmission of each compressed block with a short preamble or header file. This file is an encoding of the list* $\mathbf{n} = (\mathrm{n}_1, \mathrm{n}_2, \ldots, \mathrm{n}_\mathrm{m})$, *where* n_i *is the length of the Huffman codeword associated with the* i*th source symbol. A simple method of doing this encoding is to individually encode each* n_i *into a fixed-length binary word of length* $\log_2 \mathrm{l}$, *where* l *is an a priori upper bound on the codeword length. This method produces a maximum preamble length of* $\mathrm{m}\log_2 \mathrm{l}$ *bits. The object of this article is to show that, in most cases, no substantially shorter header of any kind is possible.*

I. Introduction

Huffman data compression is optimal for sources with known statistics ([4], Chapter 10). However, in adaptive implementations, in which the Huffman code is determined empirically by the data, the recipient of the compressed data will not know which code is being used. One way for the transmitter to identify the code is to prefix the encoded data with an ordered list of the codeword lengths being used. The receiver can then synthesize a variable-length code with these lengths, using a prearranged algorithm. (For example, the decoder can use a "greedy" algorithm, in which the shortest codewords are generated first, then the next shortest words, etc.) The object of this article is to demonstrate that, in many cases, this simple scheme is near optimal, by showing that any scheme for specifying the code will use almost as many bits as the simple preamble scheme just described.

Throughout, it will be assumed that both the compressor and decompressor know that there are m codewords, and that each codeword has a maximum length of l. One such bound can be obtained by observing that no codeword in a Huffman code with m words can be longer than $m-1$. However, in many situations, this bound can be improved upon. For example, suppose the compressor works by partitioning the source sequence into blocks of N symbols, and then estimates the source statistics as $p_i = N_i/N$, where N_i denotes the number of times the ith source symbol occurs in the block. A Huffman code for the p_i's is then synthesized and used to compress the block. The maximum

Huffman codeword length corresponds to the least probability in the source probability table, which is at least $1/N$. In a recent paper [2], Abu-Mostafa and McEliece showed that the longest Huffman codeword has a maximum length of $1.44 \log_2 p^{-1}$ for a source whose smallest probability is p, and that no better bound is possible. Thus, for an adaptive Huffman scheme, each Huffman codeword will have a maximum length of $1.44 \log_2 N$, which is often considerably smaller than $m-1$.

In the simple scheme, the code preamble will be an ordered list $(n_1, n_2, \ldots, n_m)$, where n_i is the length of the Huffman codeword for the ith symbol, and m is the number of symbols in the source alphabet. Since $n_i \leq l$, for $i = 1, 2, \ldots, m$, each n_i can be represented by a $\lceil \log_2 l \rceil$-bit binary word, the length of the preamble in bits will be $m \lceil \log_2 l \rceil$. In a slightly more sophisticated scheme, the list $(n_1, n_2, \ldots, n_m)$ can be regarded as an l-ary representation of a large integer and converted to binary, which will require at most $\lceil \log_2 l^m \rceil = \lceil m \log_2 l \rceil$ bits. In either case, one can say that the simple preamble requires around $m \log_2 l$ bits.

On the other hand, if $N(m,l)$ denotes the total number of lists $(n_1, n_2, \ldots, n_m)$ that can possibly occur as length lists for Huffman codes whose codeword lengths are at most l, then at least $\log_2 N(m,l)$ bits are needed to specify one of them. In the next section it will be shown that $\log_2 N(m,l)$ is near $m \log l$ in many cases, which implies that any scheme for specifying the Huffman code must use almost as many bits as the simple preamble scheme.

II. Main Result

It is well-known ([4], Chapter 10) that the codeword lengths in a Huffman code must satisfy the Kraft–McMillan equation

$$\sum_{i=1}^{m} 2^{-n_i} = 1 \qquad (1)$$

In this section, an estimate will be obtained for the number of ordered solutions of Eq. (1), where each n_i is further restricted to lie in the range $1 \leq n_i \leq l$. For future reference, denote this number by $N(m,l)$. A first observation is that since there are l possible values for each of the n_i's, then $N(m,l) \leq l^m$. Equivalently, if one defines $B(m,l) = \log_2 N(m,l)$ (the number of bits required to specify an arbitrary ordered solution to Eq. (1), if $1 \leq n_i \leq l$), one has

$$B(m,l) \leq m \log_2 l \qquad (2)$$

In a sense, the object of this article is to show that the upper bound in Eq. (2) is quite good; thus, in the rest of the section, lower bounds on $B(m,l)$ will be considered. The key to these bounds is the observation that if $\mathbf{n} = (n_1, \ldots, n_m)$ is a particular solution to Eq. (1), any permutation of the components of $\mathbf{n}$ will also be a solution. Indeed, if g_j denotes the multiplicity of the integer j as a component of $\mathbf{n}$, then there are exactly

$$\binom{m}{g_1, g_2, \ldots, g_l} = \frac{m!}{g_1! g_2! \cdots g_l!} \qquad (3)$$

distinct solutions to Eq. (1) that can be generated by permuting the components of $\mathbf{n}$. For example, with $m = 4$, $l = 3$, the unordered solution $(1,3,3,3,3)$ to Eq. (1) yields 5 ordered solutions, and the unordered solution $(2,2,2,3,3)$ yields 10 unordered solutions, so that $B(5,3) = 15$.

As a first step towards the general results, consider solutions to Eq. (1) with no restrictions on l. Since the longest word in a Huffman code with m words is $m-1$, this is equivalent to taking $l = m-1$. In this case, the upper bound in Eq. (2) is $m \log_2(m-1)$. On the other hand, the particular unordered solution $(1, 2, 3, \ldots, m-2, m-1, m-1)$ to Eq. (1) yields, upon permutation of its components, $m!/2$ ordered solutions, so that $B(m, m-1) \geq \log_2 m!/2$. Thus

$$\log_2(m!/2) \leq B(m, m-1) \leq m \log_2(m-1) \qquad (4)$$

It follows from Stirling's approximation to the factorial ([3], Section 1.2.11) that

$$\lim_{m \to \infty} \frac{\log(m!/2)}{m \log(m-1)} = 1$$

which means that for large m, $B(m, m-1) \sim m \log_2 m$ (see Theorem 1, below).

For restricted solutions to Eq. (1), i.e., cases when $l < m-1$, one can do something very similar. The idea is again to find a particular solution to Eq. (1) with as many distinct permutations as possible. To facilitate the discussion, now rewrite Eq. (1) as

$$\sum_{j=0}^{l} g_j 2^{-j} = 1 \qquad (5)$$

where g_j denotes the multiplicity of the integer j in the list $\mathbf{n} = (n_1, \ldots, n_m)$. The goal is to maximize the multinomial coefficient of Eq. (3), subject to Eq. (5).

The following construction yields a family of particular solutions to Eq. (5), for which the multinomial coefficient of Eq. (3) is relatively large, and which therefore provides reasonably good lower bounds for $B(m,l)$. For a given value of l, choose integers u, r, and s such that

$$3 \leq u \leq l-2, \quad 1 \leq r \leq u-2, \quad 1 \leq s \leq 2^r - 2 \quad (6)$$

Now define $g_0, g_1, \ldots, g_l$ as follows:

$$\begin{aligned} g_0 = \cdots = g_{r-1} &= 0 \\ g_r &= s \\ g_{r+1} = \cdots = g_{u-1} &= 2^r - s \\ g_u &= 2^r - s + 1 \\ g_{u+1} = \cdots = g_{l-1} &= 2^r - s - 1 \\ g_l &= 2(2^r - s - 1) \quad (7) \end{aligned}$$

It is then routine but tedious to verify algebraically that Eq. (5) holds. However, it is much easier to see that this is true by visualizing a binary tree with g_j external nodes at level j, for $j = 0, 1, \ldots, l$. Figure 1 is such a tree, for $l = 7$, $u = 5$, $r = 2$, and $s = 1$. In general, such a tree has

$$m = \sum_{j=0}^{l} g_j = s + (2^r - s)(l + 1 - r) - (l - u) \quad (8)$$

external nodes. For example, in Fig. 1 there are $m = 17$ external nodes. It thus follows that for any choice of l, u, r, and s satisfying (6), the numbers $(g_0, \ldots, g_l)$ defined in Eq. (7) give a particular solution to Eq. (5), and so

$$B(m,l) \geq \log_2 \binom{m}{g_1, g_2, \ldots, g_l}$$

where m is given by Eq. (8).

For example, since there are $m = 17$ external nodes on the Fig. 1 tree, it follows that

$$B(17,7) \geq \log_2 \binom{17}{1,3,3,4,2,4} = 33.00$$

On the other hand, from Eq. (2), $B(17,7) \leq 16 \log_2 7 = 44.92$ so that, at least on a logarithmic scale, the particular solution to Eq. (5) represented by the binary tree depicted in Fig. 1, together with its permutations, accounts for a substantial fraction of the total number of ordered solutions to the Kraft–McMillan [Eq. (1)].

The same kind of thing happens in general. That is, the largest multinomial coefficient of the form of Eq. (3), where the g_j's are given by Eqs. (6) and (7), is nearly always close to $m \log_2 l$. To see why this is so, one further specializes the solution to Eq. (5) given by Eq. (7). First, notice that Eq. (8) implies that

$$m \leq 2^r (l + 1 - r) \quad (9)$$

Choose r to be the least integer such that Eq. (9) holds. Second, having chosen r, notice that Eq. (8) implies that

$$m \leq s + (2^r - s)(l + 1 - r) \quad (10)$$

Thus, choose s to be the largest integer that Eq. (6) holds. Explicitly,

$$s = \left\lfloor \frac{2^r(l + 1 - r) - m}{l - r} \right\rfloor \quad (11)$$

Finally, having chosen both r and s, u is determined by Eq. (8), i.e.,

$$u = m + l - s - (2^r - s)(l + 1 - r) \quad (12)$$

In this way, the numbers u, r, and s are uniquely determined by m and l, as are the g_j's in Eq. (7), which in turn define the multinomial coefficient of Eq. (3). Define the logarithm of this multinomial coefficient as $B'(m,l)$. Thus, from the foregoing discussion

$$B(m,l) \geq B'(m,l)$$

For example, if $m = 20$ and $l = 8$, the least value of r satisfying Eq. (9) is $r = 2$. Then from Eq. (11), one finds that $s = 1$, and from Eq. (12), that $u = 6$. Thus from Eq. (7), $g_0 = g_1 = 0$, $g_2 = 1$, $g_3 = g_4 = g_5 = 3$, $g_6 = 4$, $g_7 = 2$, $g_8 = 4$. Finally

$$B'(20,8) = \log_2 \binom{20}{1,3,3,3,4,2,4} = 43.15$$

as compared to the upper bound of Eq. (2) $B(20,8) \le 20\log_2 8 = 60$.

A theorem will next be presented to illustrate what happens if l is a fixed fraction of m (e.g., $l = \beta m$ and $m \rightarrow \infty$), to further substantiate the claim that the upper bound in Eq. (2) is usually fairly tight.

Theorem 1. *For any fixed β, with $0 < \beta \le 1$, there are positive constants K_1 and K_2 (dependent on β but independent of m) such that for sufficiently large m,*

$$m\log_2 m - K_2 m \le B(m,\beta m) \le m\log_2 m - K_1 m \quad (13)$$

Proof: The upper bound from Eq. (2), $m\log\beta m$, is asymptotically given by

$$m\log\beta m = m\log m - \log\beta^{-1}m$$

which proves the upper bound in Eq. (13), with $K_1 = \log_2\beta^{-1}$.

To obtain an asymptotic lower bound on $B(m,\beta m)$, the reasoning is as follows. For sufficiently large m, the smallest integer solution r to Eq. (9) (using $l = \beta m$) is

$$r = r(\beta) = \lfloor \log_2\beta^{-1} \rfloor + 1 \quad (14)$$

which is a constant, independent of m. Note then from Eq. (7) that

$$g_j \le 2^r \qquad \text{for } j = r, r+1, \ldots, l-1$$

$$g_l \le 2^{r+1}$$

Thus

$$\binom{m}{g_1,\ldots,g_l} \ge \frac{m!}{(2^r!)^{l-r}(2^{r+1}!)}$$

This means that one can estimate $B(m,\beta m)$ using Stirling's formula, as follows:

$$B(m,\beta m) \ge \log_2\binom{m}{g_1,\ldots,g_l} \ge \log m!$$

$$- (l-r)\log 2^r! - \log 2^{r+1}!$$

$$\sim m\log m - (1+\beta\log 2^r!)m + O(\log m)$$

which gives a lower bound of the form promised in Eq. (13).

In conclusion, some brief remarks about the constants in Eq. (13) will be provided. When $\beta = 1$, i.e., when there are no restrictions on the length of the codewords, then the lower bound derived in Eq. (4) implies

$$B(m,m-1) \ge m\log_2 m - \log_2 em = m\log_2 m - 1.4427m$$

for sufficiently large m. On the other hand, [1] shows that T_m, the total number of *unordered* solutions to Eq. (1), satisfies $\log_2 T_m/m \rightarrow \lambda = 1.794...$. But the total number of ordered solutions to Eq. (1) cannot exceed $m!T_m$, which means (by Stirling's formula)

$$B(m,m-1) \le m\log_2 m - \log_2\frac{e}{\lambda}m = m\log_2 m - 0.5994m$$

For general β, from the proof of Theorem 1, it follows that K_1 can be taken to be

$$K_1(\beta) = \log\beta^{-1} \quad (15)$$

and that K_2 can be taken as

$$K_2(\beta) = 1 + \beta\log_2(2^r!) \quad (16)$$

where r is given by Eq. (14). Using the same method, but with a little more work, K_2 can be improved. Indeed, it can be shown that

$$K_2(\beta) = 1 + \alpha\log(2^r - s)! + (\beta-\alpha)\log(2^r - s - 1)!$$

where

$$r = \lfloor \log b^{-1} \rfloor + 1$$

$$s = 2^r - \lfloor b^{-1} \rfloor$$

$$\alpha = \beta - \left(1 - \frac{\lfloor b^{-1} \rfloor}{2^r}\right)$$

For example, with $\beta = 1/3$, Eqs. (15) and (16) give

$$K_1(1/3) = 1.58, \quad K_2(1/3) = 1.908$$

The determination of the best possible constants in Eq. (13) is an interesting and important unsolved problem.

References

[1] D. W. Boyd, "The Asymptotic Number of Solutions of a Diophantine Equation from Coding Theory," *J. Comb. Theory*, vol. (A) 18, pp. 210–215, 1975.

[2] Y. S. Abu-Mostafa and R. J. McEliece, "Maximal Codeword Lengths in Huffman Codes," *The Telecommunications and Data Acquisition Progress Report, 42-110, vol. April–June 1992*, Jet Propulsion Laboratory, Pasadena, California pp. 188–193, August 15, 1992.

[3] D. E. Knuth, *The Art of Computer Programming, vol. 1: Fundamental Algorithms*, Reading, Massachusetts: Addison-Wesley, 1968.

[4] R. J. McEliece, *The Theory of Information and Coding*, Reading, Massachusetts: Addison-Wesley, 1977.

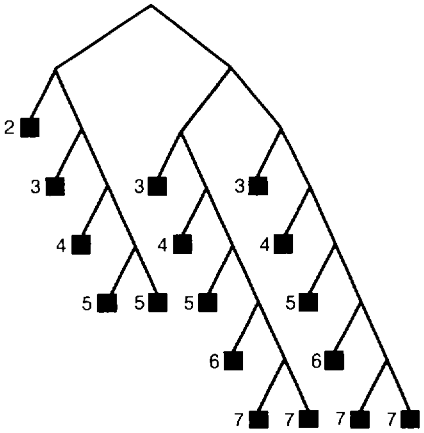

Fig. 1. A high entropy binary tree, for $m = 17$, $l = 7$. (Here $u = 5$, $r = 2$, and $s = 1$. Refer to Eq. (7).) The level of each external vertex is indicated.