On the Maximum Length of Huffman Codes

Michael Buro
Universität–GH Paderborn, FB 17 Mathematik–Informatik
Postfach 1621, W–4790 Paderborn, Germany

Abstract

In this paper the maximum length of binary Huffman codes is investigated dependent on the two lowest probabilities of encoded symbols. Furthermore, the structure of full binary trees with a given number of leaves, a limited depth, and maximum external path length is examined to get an improved upper bound on the external path length of Huffman trees.

Keywords: Huffman Code, Code Length, External Path Length.

1 Introduction

Huffman codes which were introduced in [1] encode symbols by means of their probabilities to minimize the expected code word length. In order to find a binary Huffman code for a given probability distribution, a binary tree is built from a forest initially consisting of unconnected vertices which are associated with the symbols of the alphabet and are labeled with the corresponding probabilities. In each step of the procedure two roots with the lowest probabilities are connected to a new vertex which is labeled with the sum of the probabilities of its children. Finally, the paths from the root of the Huffman tree to the leaves describe the code words of the symbols. In what follows, upper bounds on the length of Huffman code words and on the sum of all code word lengths are investigated.

2 Upper bounds

First, an upper bound on the length of Huffman code words is derived by means of the two lowest probabilities of the encoded symbols. As an immediate consequence we get an upper bound on the external path length of a Huffman tree, that is the sum of all path lengths from leaves to the root. Finally, this upper bound is improved by considering trees with limited depth and maximum external path length.

Theorem 2.1

Given a source alphabet consisting of $b > 1$ symbols, let $(p_1, \ldots, p_b)$ be a discrete probability distribution with probabilities greater than zero and $p_i \leq p_{i+1}$ for all $1 \leq i \leq b-1$. Then no code word of a corresponding binary Huffman code is longer than

$$\min\left\{\lfloor \log_\Phi\left(\frac{\Phi+1}{p_1\Phi+p_2}\right)\rfloor,\ b-1\right\},\quad \text{where } \Phi=\frac{1+\sqrt{5}}{2}.$$

Proof

Clearly, a full binary tree (i.e. a tree with vertices which have two children or none) with b leaves has depth at most $b-1$. To show the other inequality we label each vertex of the Huffman tree with the corresponding probability and prove first that $q \geq p_1F_{t+1} + p_2F_t$ holds for the root label q of a Huffman subtree with depth t. Here the F_i's are the Fibonacci numbers, i.e. $F_0 = 0,\ F_1 = 1$, and $F_{i+2} = F_{i+1} + F_i$ for $i \geq 0$.

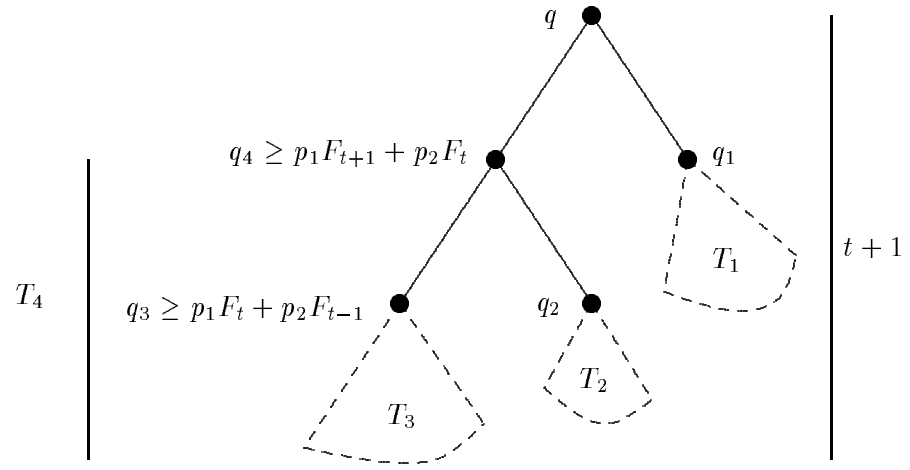

Figure 1: Huffman subtree with depth $t+1$

The proof is by induction on t. Obviously, for $t = 0$ and $t = 1$ the inequality holds since each probability is not less than p_1 and each sum of two probabilities is not less than $p_1 + p_2$. In order to show the induction step we examine a Huffman subtree T with depth $t+1$ (Figure 1). The induction hypothesis states $q_3 \geq p_1F_t + p_2F_{t-1}$ and $q_4 \geq p_1F_{t+1} + p_2F_t$ since T_3 and T_4 are Huffman subtrees with depth $t-1$ and t, respectively. Furthermore, $q_1 \geq p_1F_t + p_2F_{t-1}$ holds since otherwise T is not a Huffman subtree because in this case the expected code word length can be lowered by exchanging T_1 and T_3. This proves $q = q_1 + q_4 \geq p_1F_{t+2} + p_2F_{t+1}$.

Now we are able to show $t \leq \lfloor \log_\Phi(\frac{\Phi+1}{p_1\Phi+p_2}) \rfloor$ by means of the "closed form" expression for the Fibonacci numbers which is proved for instance in [2]: $F_i = (\Phi^i - (-\Phi)^{-i})/\sqrt{5}$. The root of a Huffman tree has label 1. Hence, $1 \geq p_1F_{t+1} + p_2F_t$. Assuming $t \geq 1$ a few transformations yield

$$t \;\leq\; \log_\Phi(\sqrt{5} + p_2/\Phi^2 - p_1/\Phi^3) - \log_\Phi(p_1\Phi + p_2) \;\leq\; 2 - \log_\Phi(p_1\Phi + p_2) \;=\; \log_\Phi\left(\frac{\Phi+1}{p_1\Phi+p_2}\right)$$

Since the inequality also holds for $t < 1$ and t is an integer, the result is shown.

□

In the case that we only know a lower bound $p > 0$ on the minimum probability p_1 we immediately obtain $\min\{\lfloor -\log_\Phi p \rfloor,\ b-1\}$ as an upper bound on the maximum length of a code word. For example, given $p_1 \geq 0.2 \cdot 10^{-6}$, no Huffman code word is longer than 32.

Corollary 2.2

In terms of the notation above, the external path length of a Huffman tree is at most

$$\min\left\{\lfloor \log_\Phi\left(\frac{\Phi+1}{p_1\Phi+p_2}\right)\rfloor \cdot b,\ (b+2)(b-1)/2\right\}.$$

Proof

The Huffman tree has b leaves and each length of a path from a leaf to the root is bounded by $\lfloor \log_\Phi(\frac{\Phi+1}{p_1\Phi+p_2}) \rfloor$. On the other hand, a tree with b leaves has maximum external path length if and only if it is degenerated, that is a list like tree T_1 in Figure 2 (see Theorem 2.4 for a proof). A degenerated tree with b leaves has external path length $(b+2)(b-1)/2$.

□

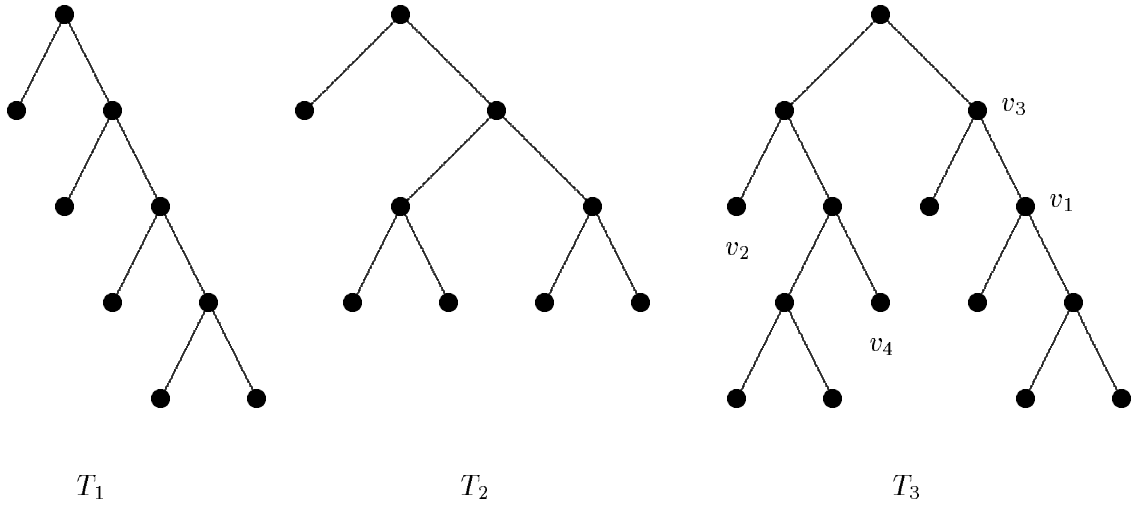

Figure 2: Example trees

The usual technique to encode a sequence of symbols by the use of a Huffman code is to count the symbols in order to estimate their probabilities. For this method the statements above can be adapted: Given a sequence of length n over an alphabet with $b > 1$ symbols, let $n_1 \geq 1$ and $n_2 \geq n_1$ be the two lowest frequencies. Then the statements of Theorem 2.1 and Corollary 2.2 hold for $p_1 = n_1/n$ and $p_2 = n_2/n$, and the maximum length of a code word can be bounded for instance by $\min\{\lfloor \log_\Phi(\frac{n(\Phi+1)}{n_1\Phi+n_2})\rfloor,\ b-1\}$ or simply by $\min\{\lfloor \log_\Phi n\rfloor,\ b-1\}$.

In what follows, a better upper bound on the maximum external path length of Huffman trees is derived. The idea is to examine the structure of full binary trees with a given number of leaves and a limited depth which have maximum external path length. Figure 2 shows some examples: T_1 and T_2 are trees with five leaves. They have maximum external path length in the case that the depth is limited by $t \geq 4$ and $t = 3$, respectively. On the other hand, T_3 is not maximum for a depth limit $t \geq 4$ since the external path length can be increased by shifting the subtree under v_1 to v_2 and then shifting the remaining subtree under v_3 to v_4. *Let E_t denote the property that the external path length of a tree with depth at most t can not be increased by first shifting subtrees on the same depth and then shifting down a subtree.* Clearly, a maximum tree with depth not greater than t has property E_t. In order to prove the converse we first give an equivalent condition for E_t which can be tested easily.

Lemma 2.3

Given a full binary tree T with depth at most t, let $(a_i)_{i=0}^t$ be the leaf distribution of T, i.e. a_i is the number of leaves on depth i. Then T has property E_t if and only if the following conditions hold:

i) $0 \leq a_i \leq 2$ for all $0 \leq i \leq t-1$

ii) If there is an $i < t$ with $a_i = 2$ then $a_j = 0$ for all $i < j < t$.

Proof

If i) is violated, i.e. there is an i with $a_i \geq 3$, a small subtree with leaves on depth i and root on depth $i-1$ can be shifted down to the third leaf on depth i — possibly after moving another subtree rooted on depth i — in order to increase the external path length. If ii) is

not met, we can analogously shift a subtree to get a small tree with two leaves on depth i which then can be shifted down to depth j. Conversely, if a subtree can be shifted down and i) is true then ii) can not hold. To show this let the subtree be rooted on depth i and be moved to depth $j > i$. If the maximum depth of leaves in the subtree is at most j then i) or ii) is violated. Otherwise, there is a subtree with root v on depth $j-1$ which can be moved to depth j. Clearly, among the two children of v there are one or two leaves or none. In each case we get a contradiction to i) or ii).

□

Theorem 2.4

In the set of full binary trees with b leaves and depth at most t a tree has maximum external path length if and only if conditions i) and ii) of Lemma 2.3 are met. Furthermore, the leaf distribution of a tree with maximum path length is unique.

Proof

Of course, a tree with depth limit t and maximum external path length has property E_t. Therefore, conditions i) and ii) of Lemma 2.3 are valid. Next we show that the leaf distribution is unique if i) and ii) hold. This proves the other implication and the second statement since a maximum tree has property E_t.

Let T be a full binary tree with depth at most t, b leaves, and property E_t. We show by induction on t that its leaf distribution is unique. The case $t = 0$ is trivial. Suppose now that the leaf distribution is unique for depths less than t. If $b \leq t$ then T has depth less than t and the leaf distribution is unique according to the induction hypothesis (T is degenerated). In the other case ($b > t$) T has a shape like the tree in Figure 3 up to shifting subtrees on the same depth with $k = \max\{r \in \mathbb{N} \mid 2^r + t - r \leq b\}$, where $\mathbb{N}$ denotes the set of integers greater than zero: Clearly, if $k = t$ then T is complete and its leaf distribution is unique. Otherwise, we use conditions i) and ii) to show that T has depth t, $a_i = 1$ for all $0 < i < t - k$, and $a_t \geq 2^k$. Since T_1 has depth at most $k < t$ and has property E_k, the leaf distribution of T_1 is unique and consequently the leaf distribution of T, too.

- T has depth t. Otherwise i) or ii) is violated because then there are at least two leaves with maximum depth $i < t$ and at least two more leaves with equal depths not greater than i.

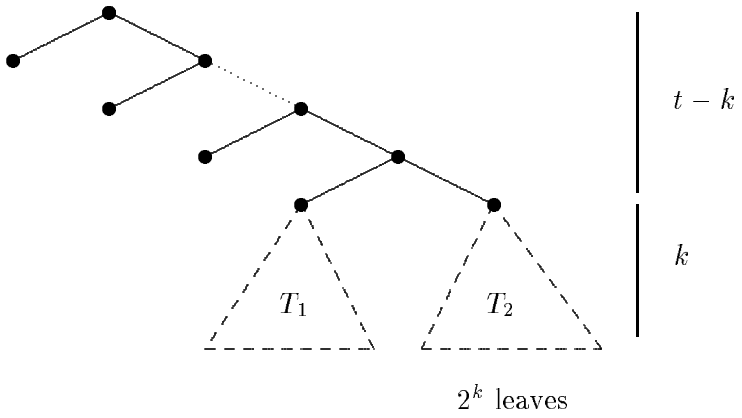

Figure 3: Shape of a maximum tree

- There are no other subtrees with more than one leaf rooted on depth less than $t-k$. If T' is such a tree then we shift subtrees from T' to T_1 or T_2 on the same depth until each leaf of T' has depth less than t and not greater than the minimum depth of leaves in T_1 and T_2. Again i) or ii) is violated.
- There are at least 2^k leaves with depth t. If this is not true then in T_1 and T_2 there are respectively less than 2^k and at least two leaves since the number of leaves in T_1 and T_2 is at least 2^k+1. This follows from $b \geq 2^k + t - k$. Shifting subtrees from T_1 to T_2 like above yields the desired contradiction.

□

Based on Theorem 2.4 the following recursive function quickly computes the maximum external path length of full binary trees with a given number of leaves and a depth limit:

Function MAX–LENGTH

Input: $t, b \in \mathbb{N} \quad (b \leq 2^t)$
Output: Maximum external path length of a full binary tree with b leaves and depth $\leq t$

if $b \leq t$ **then return** $(b+2)(b-1)/2$
else begin
  $k := \lfloor \log_2(b-t) \rfloor$
  if $2^{k+1} - (k+1) \leq b - t$ **then** $k := k+1$
  return $2^k k + (t-k)(2b-t+k+1)/2 +$ MAX–LENGTH$(k, b - 2^k - t + k + 1)$
end

Corollary 2.5
Function MAX–LENGTH is correct and has unit–cost time complexity $O(\log(b-t))$.

Proof
The correctness is an immediate consequence of the proof of Theorem 2.4:
If $b \leq t$, a maximum tree is degenerated and has external path length $b(b-1)/2 + b - 1 = (b+2)(b-1)/2$. In the other case, $k = \max\{r \in \mathbb{N} \mid 2^r - r \leq b - t\}$ is computed using $\lfloor \log_2(b-t) \rfloor \leq k \leq \lfloor \log_2(b-t) \rfloor + 1$ which follows immediately. In order to find k exactly the algorithm tests whether $2^{k+1} - (k+1) \leq b - t$ holds. The maximum external path length is the sum of the external path length of the complete subtree $[2^k t]$, the degenerated initial part $[(t-k)(t-k-1)/2]$, and the remaining subtree $[(b - 2^k - (t-k-1))(t-k) +$ MAX–LENGTH$(k, b - 2^k - (t-k-1))]$. It remains to prove that the function terminates. But this is clear since $k \geq 1$ holds and, therefore, the difference $b-t$ is lowered by each function call. Moreover, a short calculation shows that after a function call $b-t$ is not greater than $\lfloor (b'-t')/2 \rfloor + 1$, where b' and t' denote the variables b and t of the caller. Thus, the time complexity of function MAX–LENGTH is $O(\log(b-t))$.

□

Now Theorem 2.1 and function MAX–LENGTH can be combined to get an improved upper bound on the external path length of Huffman trees:

- First, the maximum depth of a Huffman tree is determined using Theorem 2.1 and
- then the maximum external path length of full binary trees with this depth limit is computed by function MAX–LENGTH.

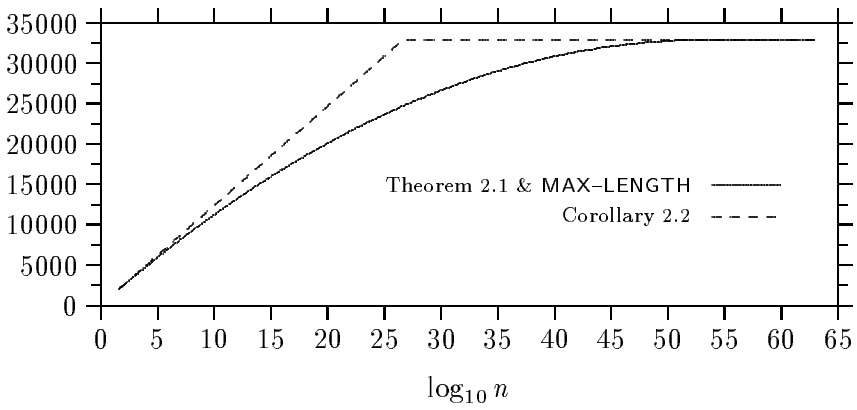

Figure 4: Bounds on the external path length ($b = 256$)

It is clear that the result is not greater and often less than the upper bound stated in Corollary 2.2. Figure 4 illustrates the behavior of the two bounds, where n denotes the length of the encoded sequence over an alphabet consisting of 256 symbols.

3 Conclusion

In the explanations above, upper bounds on the maximum length of Huffman code words and the maximum external path length of Huffman trees have been derived taking into account the lowest probabilities and the lowest frequencies of encoded symbols, respectively. As a next step one could try to find an approximation to the maximum external path length which is computed by function MAX–LENGTH, or to use more properties of Huffman trees to improve the given bounds.

References

[1] Huffman, D. A.: *A Method for the Construction of Minimum–Redundancy Codes*, Proceedings of the IRE 40 (1952), pp. 1098-1101

[2] Knuth, D. E.: *The Art of Computer Programming. Vol. 1: Fundamental Algorithms*, Addison-Wesley (1973)