*Journal of Universal Computer Science, vol. 3, no. 11 (1997), 1167-1179*
*submitted: 8/8/97, accepted: 22/11/97, appeared: 28/11/97 © Springer Pub. Co.*

# Optimum Extendible Prefix Codes[1]

Cristian S. Calude
(Computer Science Department, The University of Auckland, Private Bag 92019, Auckland, New Zealand
Email: cristian@cs.auckland.ac.nz.)

Ioan Tomescu
(Bucharest University, Faculty of Mathematics, Str. Academiei 14, R-70109, Bucharest, Romania
Email: ioan@inf.math.unibuc.ro.)

**Abstract:** Suppose that we have $L$ messages coded by a prefix code (over an alphabet $M$ with $m$ letters) having a minimum weighted length. The problem addressed in this paper is the following: How to find $s$ codewords for new messages so that by leaving unchanged the codification of the first $L$ messages (by compatibility reasons), the resulting extended code is still prefix (over $M$) and has a minimum weighted length? To this aim we introduce the notion of *optimum extendible prefix code* and then, by modifying Huffman's algorithm, we give an efficient algorithm to construct the optimum extension of a non-complete prefix code, provided the initial code is optimal.

**Key Words:** Kraft's inequality, Huffman tree, optimum extendible prefix code

**Category:** F.1

## 1 Introduction

A prefix codeword set has to satisfy Kraft's inequality [8], and, conversely, Kraft's inequality is a sufficient condition for the existence of a prefix code with the specified set of codeword lengths.

Huffman's algorithm [7] solves the problem of finding a prefix code with the minimum weighted length: Given the weights $p_1, p_2, \ldots, p_L \geq 0$ find the lengths $l_1, l_2, \ldots, l_L$ satisfying Kraft's inequality whose weighted length $\sum_{i=1}^{L} p_i l_i$ is less than or equal to the weighted length of any prefix code of cardinality $L$.

Chaitin [3] has extended Kraft's inequality for recursively enumerable prefix codes (i.e. for codes enumerated by algorithms): as long as Kraft's *strict* inequality is guaranteed, one can extend indefinitely any prefix code. This result is essential in algorithmic information theory (see [4, 2]). In this context we address the following question: Is it possible to extend an optimal prefix code, under the assumption that the extension is still optimal? Of course, if the code is complete,

[1] Proceedings of the *First Japan-New Zealand Workshop on Logic in Computer Science*, special issue editors D.S. Bridges, C.S. Calude, M.J. Dinneen and B. Khoussainov.

then no extension is possible. We prove that in all remaining cases an optimal extension can be constructed and we give an efficient algorithm to construct the optimum extension.

Mathematically, we will solve an optimization problem: as in Huffman's case we optimize the weighted length function. However, the restrictions are different: classically, one optimizes over all integers satisfying Kraft's inequality, while here we optimize over all integers satisfying Kraft's *strict* inequality which corresponds to extendible prefix codes.

## 2 Notation

Consider an alphabet $M = \{0, 1, \ldots, m-1\}$ containing $m \geq 2$ letters. The set of all words over $M$ (endowed with the natural order $0 < 1 < \cdots < m-1$) can be represented as an infinite complete $m$-ary tree $\mathcal{U}$ in which the root is labeled by the empty word. For instance, in the binary case, the first four levels of this tree are the following:

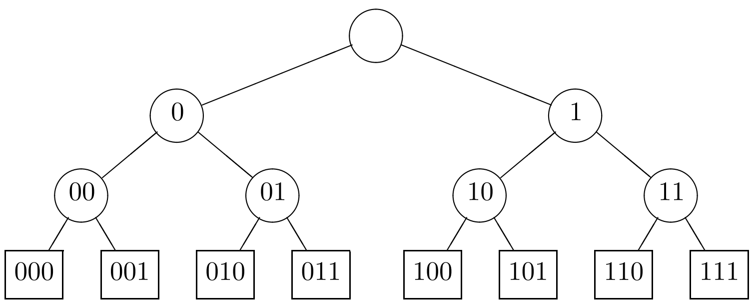


Figure 1

The tree is drawn with the root on the top, all edges are pointed downward, and the sons are lined up horizontally in the lexicographical order. A sub-tree of $\mathcal{U}$ is called "positional $m$-ary directed tree" in the graph-theoretical literature [6].

A *prefix code* is a set of words $C$ such that no word in $C$ is a proper prefix of another word in $C$. Prefix codes are uniquely decodable, as the end of a codeword is immediately recognizable. For example, the set $\{0^i 1 \mid i \geq 1\}$ is a prefix code over the binary alphabet. The codewords of a prefix code $C = \{w_1, w_2, \ldots, w_L\}$ over $M$ satisfy Kraft's inequality

$$\sum_{i=1}^{L} m^{-l_i} \leq 1, \tag{1}$$

where $l_i = |w_i|$ denotes the length of $w_i$. See [1, 5] for more facts on prefix codes. The sum in the left-hand side of (1) is called the *characteristic sum of* $C$.

The set of words which correspond to terminal vertices of a finite sub-tree of $\mathcal{U}$ form a prefix code; this correspondence is bijective. If $C$ is a prefix code over $M$, then the associated $m$-ary tree $\mathcal{T}(C)$ has height $h(\mathcal{T}(C)) = \max\{|w| \mid w \in C\}$. The inequality (1) holds true with equality iff $\mathcal{T}(C)$ is complete, i.e. in case every non-terminal vertex of $\mathcal{T}(C)$ has exactly $m$ sons. A prefix code $C$ is complete if $\mathcal{T}(C)$ is complete. In what follows we shall adopt the graph-theoretical terminology from [6].

## 3 Kraft's Inequality Revisited

In this section we prove a slightly stronger version of the Kraft's inequality. Recall that the degree of a vertex is equal to the number of its sons.

**Lemma 3.1.** *Let $C = \{w_1, w_2, \ldots, w_L\}$ be a prefix code over $M$ such that the root of its associated $m$-ary tree $\mathcal{T}(C)$ has the degree $f$, $1 \leq f \leq m$. Then the characteristic sum of $C$ satisfies the inequality*

$$\sum_{i=1}^{L} m^{-l_i} \leq \frac{f}{m}. \tag{2}$$

*The equality holds true iff every non-terminal vertex of $\mathcal{T}(C)$, different from the root, has exactly $m$ sons.*

*Proof.* By hypothesis, the root of $\mathcal{T}(C)$ has $f$ sons, say $v_1, v_2, \ldots, v_f$. Every $v_i$ is the root of some sub-tree of $\mathcal{T}(C)$. Let $C_i$ be the corresponding code, i.e. a word is in $C_i$ if it is obtained from a unique word in $C$, by removing its first letter; so a word in $C_i$ has the length shorter by one than its corresponding word in $C$. Accordingly, if $S_i$ is the characteristic sum of $C_i$, then

$$\sum_{i=1}^{L} m^{-l_i} = \frac{1}{m}\sum_{i=1}^{f} S_i \leq \frac{f}{m}.$$

The last inequality is a consequence of Kraft's inequality for $C_i$. □

For example, if $m = 3$, $C = \{00, 01, 022, 20\}$, then $f = 2$, $C_1 = \{0, 1, 22\}$, $C_2 = \{0\}$, and $S_1 = 7/9$, $S_2 = 1/3$, $S = \frac{1}{3}(S_1 + S_2) \leq 2/3$.

**Lemma 3.2.** *If a vector of word lengths $(l_1, l_2, \ldots, l_L)$ satisfies the inequality (2), then one can effectively construct a prefix code $C$ over $M$ corresponding to the given vector of word lengths such that its associated $m$-ary tree $\mathcal{T}(C)$ has its root of degree $f$ $(1 \leq f \leq m)$.*

The proof is similar to the case in which (1) is used instead of (2); see, for instance, [6], pp. 130-131.

Assume now that we have $L$ positive weights, $p_1 \geq p_2 \geq \cdots \geq p_L$.

**Huffman's Optimization Problem (HOP)**: *Construct a prefix code* $C = \{w_1, w_2, \ldots, w_L\}$ *over* $M$ *such that if* $l_i = |w_i|$, *for all* $1 \leq i \leq L$, *then the cost*

$$cost(C) = \sum_{i=1}^{L} p_i l_i$$

*is minimum among all prefix codes of cardinality* $L$.

In view of Lemma 3.2, to solve the above problem we have to construct a vector of word lengths $l_1, l_2, \ldots, l_L$ which has a minimum cost among all vectors satisfying the inequality (1). Huffman's algorithm is actually producing such a solution: start with $L' = L$ and the weights $p_1 \geq p_2 \geq \cdots \geq p_L$. At every step we add the least, i.e. the last, $d$ numbers in the ordered sequence, put the result in the proper place, and decrease the length $L' = L' - d + 1$. The current number $d$ is computed as follows: if $m = 2$, then $d = 2$; otherwise,

$$d = \begin{cases} m, & \text{if } L' \equiv 1 \pmod{m-1}, \\ m-1, & \text{if } L' \equiv 0 \pmod{m-1}, \\ \rho, & \text{if } L' \equiv \rho \pmod{m-1}, \text{ and } 2 \leq \rho \leq m-2. \end{cases} \tag{3}$$

The values for $d$ stabilize after one step, as $L' \equiv 1 \pmod{m-1}$, and $d$ becomes equal to $m$ from there on.

The operation is repeated until we end up with $L' = m$ weights, each to be assigned length one. Then we start working our way back up: we assign the same length to weights in the previous step, and we increase by one the length of each of the last $d$ weights. Starting with the second step one has $L' \equiv 1 \pmod{m-1}$; the correction, if any, for the initial $L$ is operated at the first step, in case $m \geq 3$. The resulting prefix code $C$ is optimal.

Notice that every prefix code constructed by Huffman's algorithm is complete, satisfying (1) with equality.

## 4 Binary Optimum Extendible Prefix Codes

In this section we solve the following problem:

**Extendible Huffman's Optimization Problem (EHOP). The Binary Case**: *Given* $L$ *positive weights* $p_1 \geq p_2 \geq \cdots \geq p_L$ *find a binary prefix code* $C = \{w_1, w_2, \ldots, w_L\}$ *such that if* $l_i = |w_i|, 1 \leq i \leq L$, *then*

$$\sum_{i=1}^{L} p_i l_i$$

*is minimum over all positive integers* $l_1, l_2, \ldots, l_L$ *such that*

$$\sum_{i=1}^{L} 2^{-l_i} < 1. \tag{4}$$

Using Huffman's algorithm we build an optimal tree $HT(p_1, p_2, \ldots, p_L)$ for the given weights $p_1 \geq p_2 \geq \cdots \geq p_L$. This optimal tree is *complete*, i.e. (1) is satisfied with equality.

Now consider a terminal vertex $u$ of $HT(p_1, p_2, \ldots, p_L)$ having the maximum depth associated to a minimum weight $p = \min\{p_1, p_2, \ldots, p_L\}$. We transform $u$ into an internal vertex having a new (left or right) son $v$ on the level $depth(u)+1$, and we associate to $v$ the weight $p$. We have obtained a non-complete binary tree $EHT(p_1, p_2, \ldots, p_L)$, call it *extendible Huffman tree* (built for the weights $p_1, p_2, \ldots, p_L$).

**Theorem 4.1.** *The extendible Huffman tree $EHT(p_1, p_2, \ldots, p_L)$ is a solution for the binary* **EHOP**.

*Proof.* Let $C$ be an optimum extendible prefix code for **EHOP**. Since Kraft's strict inequality (4) is satisfied, its associated tree $\mathcal{T}_0 = \mathcal{T}(C)$ is not complete. Suppose first that $\mathcal{T}_0$ contains an internal vertex $x$ having $depth(x) \leq h(\mathcal{T}_0) - 2$ and a single son; assume that $x$ has maximum depth.

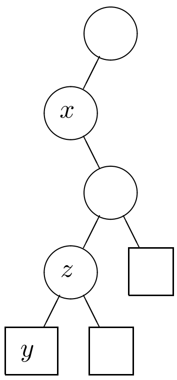


Figure 2$a$

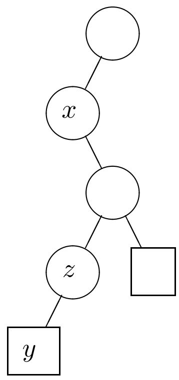


Figure 2$b$

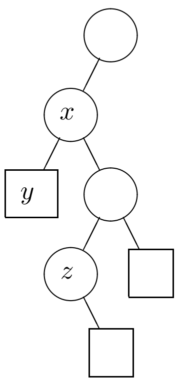


Figure 3$a$

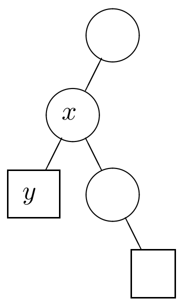


Figure 3$b$

We can move a terminal vertex $y$ lying on the last level (see Figure 2$a$, respectively, Figure 2$b$) as son of $x$ (see Figure 3$a$, respectively, Figure 3$b$; in the last case its father $z$ is deleted). A new non-complete binary tree $\mathcal{T}_1$ is obtained from $\mathcal{T}_0$, and $cost(\mathcal{T}_1) < cost(\mathcal{T}_0)$, which contradicts the hypothesis on $C$. It follows that all internal vertices having a single son have the depth equal to $h(\mathcal{T}_0) - 1$. If $\mathcal{T}_0$ has two vertices $x$ and $z$ having each a single son, $y$, and $t$, respectively, on level $h(\mathcal{T}_0)$, we can delete the vertex $t$. In this way $z$ becomes a terminal vertex and we associate to $z$ the weight of $t$. A new non-complete binary tree $\mathcal{T}_2$ is obtained such that $cost(\mathcal{T}_2) < cost(\mathcal{T}_0)$. Hence $\mathcal{T}_0$ has a unique vertex $u$ on the level $h(\mathcal{T}_0) - 1$ having a single son $v$ on the level $h(\mathcal{T}_0)$. The optimal extendible tree is among the trees whose internal nodes have two sons except one, and the only one internal node that has a single son is of dept $h(\mathcal{T}_0) - 1$. Let $p(v)$ be the weight of $v$ and $\mathcal{T}'$ be the binary tree deduced from $\mathcal{T}_0$ be deleting $v$; associate to $u$ the weight $p(v)$. Then,

$$\begin{aligned} cost(\mathcal{T}_0) &= cost(\mathcal{T}') + p(v) \\ &\geq cost(HT(p_1, p_2, \ldots, p_L)) + \min\{p_1, p_2, \ldots, p_L\} \\ &= cost(EHT(p_1, p_2, \ldots, p_L)), \end{aligned}$$

since $cost(\mathcal{T}') \geq cost(HT(p_1, p_2, \ldots, p_L))$ and $p(v) \geq \min\{p_1, p_2, \ldots, p_L\}$. □

Now we propose an algorithm for constructing an optimal extension of an optimal, non-complete, binary prefix code. Given the weights $p_1, p_2, \ldots, p_L$ we build the extendible Huffman tree $\mathcal{T}_1 = EHT(p_1, p_2, \ldots, p_L)$ which is optimal by Theorem 4.1. This tree generates the code $C = \{w_1, w_2, \ldots, w_L\}$ satisfying **EHOP**. Let $l_i = |w_i|$, for $1 \leq i \leq L$. Suppose that we want to extend $C$ with $s$ new words $w_{L+1}, w_{L+2}, \ldots, w_{L+s}$ having the weights $p_{L+1}, p_{L+2}, \ldots, p_{L+s}$, such that if $l_i = |w_i|$, for $L + 1 \leq i \leq L + s$, then

$$\sum_{i=L+1}^{L+s} p_i l_i$$

is minimum over all positive integers $l_{L+1}, l_{L+2}, \ldots, l_{L+s}$ such that

$$\sum_{i=L+1}^{L+s} 2^{-l_i} < 1 - \sum_{i=1}^{L} 2^{-l_i}.$$

We notice that in this case the extendibility condition

$$\sum_{i=1}^{L+s} 2^{-|w_i|} < 1,$$

is fulfilled. To this aim let $\mathcal{R} = EHT(p_{L+1}, p_{L+2}, \ldots, p_{L+s})$ be an extendible Huffman tree built for the weights $p_{L+1}, p_{L+2}, \ldots, p_{L+s}$. Let $v$ be the unique terminal vertex of $\mathcal{T}_1$ having $depth(v) = h(\mathcal{T}_1)$ and $u$ its father. We shall join $u$ and the root $r$ of $\mathcal{R}$ by an edge such that $u$ has now two sons: $v$ and $r$ (see Figure 4).

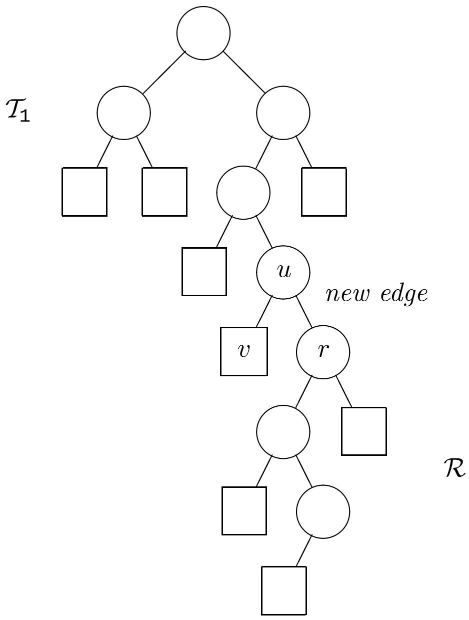


Figure 4

We have obtained an extendible binary tree $\mathcal{T}$ which is a solution of our problem, i.e. its associated prefix code extends $C$, and satisfies the extendibility and optimality conditions.

Notice that if the strict inequality above is relaxed to a non-strict one, $\mathcal{R}$ must be the complete tree $HT(p_{L+1}, p_{L+2}, \ldots, p_{L+s})$.

**Theorem 4.2.** *The tree $\mathcal{T}$ is an optimal extension of $\mathcal{T}_1$.*

*Proof.* Let $h(\mathcal{T}_1) = t$. We get

$$cost(\mathcal{T}) = cost(\mathcal{T}_1) + cost(\mathcal{R}) + t \sum_{i=L+1}^{L+s} p_i,$$

and any binary tree built by extension of $\mathcal{T}_1$ has its cost function of this form as the unique vertex of $\mathcal{T}_1$ having a single son is $u$.
It is clear that $\mathcal{T}$ is extendible iff $\mathcal{R}$ is extendible; since $\mathcal{R} = EHT(p_{L+1}, p_{L+2}, \ldots, p_{L+s})$ it follows that $\mathcal{T}$ is extendible and optimal. □

The above extension can continue indefinitely; it may be stopped by extending the tree with a complete Huffman tree, not with an extendible Huffman tree.

## 5 Optimum Extendible Prefix Codes: The Non-Binary Case

We are now ready to deal with the general problem:

**Extendible Huffman's Optimization Problem (EHOP)**: *Given $L$ positive weights $p_1 \geq p_2 \geq \cdots \geq p_L$ and a positive integer $m \geq 2$, find a prefix code $C = \{w_1, w_2, \ldots, w_L\}$ over $M = \{0, 1, \ldots, m-1\}$ such that if $l_i = |w_i|, 1 \leq i \leq L$, then*

$$\sum_{i=1}^{L} p_i l_i$$

*is minimum over all positive integers $l_1, l_2, \ldots, l_L$ such that*

$$\sum_{i=1}^{L} m^{-l_i} < 1. \tag{5}$$

### 5.1 Restricted Huffman Trees

We start solving the following problem:

**Restricted Huffman's Optimization Problem (RHOP)**: *For given integers $1 \leq f \leq m$, $m \geq 3$, and a vector of $L \geq f$ positive weights $(p_1, p_2, \ldots, p_L)$ such that $p_1 \geq p_2 \geq \cdots \geq p_L$, construct a prefix code $C$ over the alphabet $M = \{0, 1, \ldots, m-1\}$ such that the word lengths $(l_1, l_2, \ldots, l_L)$ satisfy the following two conditions:*

(a) *The root* of $\mathcal{T}(C)$ *has degree* $f$.
(b) *The weighted length* $cost(C) = \sum_{i=1}^{L} p_i l_i$ *is minimum among all prefix codes of cardinality* $L$.

An optimum Huffman tree satisfying (b) will be denoted, as in the binary case, by $HT(p_1, p_2, \ldots, p_L)$, and an optimum tree restricted by the first condition will be denoted by $RHT(f; p_1, p_2, \ldots, p_L)$. It is clear that an $RHT(m; p_1, p_2, \ldots, p_L)$ is also an $HT(p_1, p_2, \ldots, p_L)$.

In building $RHT(f; p_1, p_2, \ldots, p_L)$ we again rely on Huffman's algorithm and change the construction of $d$ in (3) as follows:

$$d = \begin{cases} \rho + 1, & \text{if } \rho \neq 0, \\ m, & \text{if } \rho = 0 \text{ and } L' > f, \\ f, & \text{if } \rho = 0 \text{ and } L' = f, \end{cases} \tag{6}$$

where $L' - f \equiv \rho \pmod{m-1}$ and $0 \leq \rho \leq m-2$. After the first step the length of the vector of word lengths, $L'$, satisfies the relation $L' \equiv f \pmod{m-1}$ and will be equal to $f$ modulo $m-1$ from there on. The rule (6) ensures that eventually we end up with exactly $f$ weights, each to be assigned length one.

Let $\Lambda = (\lambda_1, \lambda_2, \ldots, \lambda_L)$ be an optimal word length vector for the weight vector $\Pi = (p_1, p_2, \ldots, p_L)$, $p_1 \geq p_2 \geq \cdots \geq p_L$, such that conditions (a) and (b) are fulfilled.

Exactly as for the case when only the second condition is satisfied one can prove that if $p_i > p_j$, then $\lambda_i \leq \lambda_j$, so we may assume that $\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_L$.

**Lemma 5.1.** *Let $\Lambda, \Pi$ be as above. Then,*

$$\lambda_{L-d+1} = \lambda_{L-d+2} = \cdots = \lambda_{L-1} = \lambda_L,$$

*where $d$ comes from the above formula (6) replacing $L'$ by $L$.*

*Proof.* First we show that

$$m - 2 \geq f\, m^{\lambda_L - 1} - \sum_{i=1}^{L} m^{\lambda_L - \lambda_i} \geq 0. \tag{7}$$

The right-hand inequality follows directly from Kraft's inequality (Lemma 3.1). The left-hand inequality can be proved as follows. Assume that

$$m - 1 \leq f\, m^{\lambda_L - 1} - \sum_{i=1}^{L} m^{\lambda_L - \lambda_i}.$$

Thus,

$$\sum_{i=1}^{L-1} m^{\lambda_L - \lambda_i} + m \leq f\, m^{\lambda_L - 1}.$$

Dividing by $m^{\lambda_L}$ we get

$$\sum_{i=1}^{L-1} m^{-\lambda_i} + m^{-(\lambda_L - 1)} \leq \frac{f}{m},$$

which contradicts the optimality of $\Lambda$ by Lemma 3.2. Now let $L - f \equiv \rho \pmod{m-1}$, where $0 \leq \rho \leq m-2$. Since for every non-negative integer $k$, $m^k \equiv 1 \pmod{m-1}$, we have

$$f\, m^{\lambda_L - 1} - \sum_{i=1}^{L} m^{\lambda_L - \lambda_i} \equiv f - L \equiv -\rho \pmod{m-1}.$$

By (7) we conclude that

$$f\, m^{\lambda_L - 1} - \sum_{i=1}^{L} m^{\lambda_L - \lambda_i} = \begin{cases} 0, & \text{if } \rho = 0, \\ m - 1 - \rho, & \text{if } 1 \leq \rho \leq m-2. \end{cases} \tag{8}$$

Let $j$ be the last index which satisfies $\lambda_j < \lambda_L$. It follows that

$$\lambda_{j+1} = \lambda_{j+2} = \cdots = \lambda_L.$$

We can re-write (8) as

$$f\, m^{\lambda_L - 1} - \sum_{i=1}^{j} m^{\lambda_L - \lambda_i} = \begin{cases} L - j, & \text{if } \rho = 0, \\ L - j + m - 1 - \rho, & \text{if } 1 \leq \rho \leq m - 2. \end{cases} \tag{9}$$

If $\lambda_L = 1$, then $\lambda_1 = \cdots = \lambda_L = 1$, and the proof is complete. Otherwise, in (9) the left-hand side is divisible by $m$ since all powers of $m$ are positive. Then, the right-hand side, in addition to being positive must be divisible by $m$ too. Thus,

- if $\rho = 0$, then $L - j = km$, for some positive integer $k$, and
- if $1 \leq \rho \leq m-2$, then $L - j + m - 1 - \rho = km$, for some positive integer $k$.

By (6) this implies that

- if $\rho = 0$, then $L - j \geq m \geq d$, since $f \leq m - 1$, and
- if $1 \leq \rho \leq m - 2$, then $L - j \geq \rho + 1 = d$.

□

We continue by proving that the modified rule (6) leads to a solution for **RHOP**. Let us denote by $\Pi_0$ the original vector of heights (in a non-increasing order), and by $\Pi_i$ the weight vector after $i$ iterations of the above process. The vector of word lengths assigned to $\Pi_i$ is denoted by $\Lambda_i$.

**Theorem 5.2.** *Let $\Pi_0, \Pi_1, \ldots, \Pi_s$ and $\Lambda_0, \Lambda_1, \ldots, \Lambda_s$ be the weight vectors and the word length vectors, as constructed above. Then, for every $0 \leq i \leq s$, $\Lambda_i$ is optimal for $\Pi_i$.*

*Proof.* The construction assures that $L_i$, the number of weights in $\Pi_i$, satisfies the equation $L_i \equiv f \pmod{m-1}$, for every $1 \leq i \leq s$. Also, $L_s = f$, and obviously $\Lambda_s = \underbrace{(1, 1, \ldots, 1)}_{f \text{ times}}$ is optimal for $\Pi_s$. For the proof of the fact that if $\Lambda_{i+1}$ is optimal for $\Pi_{i+1}$, then $\Lambda_i$ is optimal for $\Pi_i$ we use Lemma 5.1 and follow closely the proof of the validity of Huffman's algorithm for **HOP** (see [6], for instance). □

In conclusion, the $m$-ary tree built by means of the rule (6) is indeed a $RHT(f; p_1, p_2, \ldots, p_L)$.

### 5.2 Extendible Restricted Huffman Trees

An $m$-ary tree is called *extendible* if the vector of word lengths associated with its terminal vertices satisfies Kraft's strict inequality (5), and a restricted $m$-ary tree is *extendible* if the inequality

$$\sum_{i=1}^{L} m^{-l_i} < \frac{f}{m} \tag{10}$$

is verified.

From the above conditions we deduce that we can construct arbitrarily many new descendants of non-terminal vertices such that for the extended tree the vector of word lengths satisfies again the inequalities (1) and (2), respectively. As in the binary case we shall denote by $EHT(p_1, p_2, \ldots, p_L)$ and $ERHT(f; p_1, p_2, \ldots, p_L)$, respectively, an $m$-ary tree associated to a prefix code $C$ over $M$ satisfying (5) plus (b), or (10) plus (a), (b), respectively. These trees will be called *optimum extendible Huffman tree* and *optimum extendible restricted Huffman tree*, respectively.

From (3) and (6) we deduce that $EHT(p_1, p_2, \ldots, p_L) = HT(p_1, p_2, \ldots, p_L)$ unless $L \geq m$ and $L \equiv 1 \pmod{m-1}$, and $ERHT(f; p_1, p_2, \ldots, p_L) = RHT(f; p_1, p_2, \ldots, p_L)$ unless $L > f$ and $L \equiv f \pmod{m-1}$ (if $f \leq m-1$).

Suppose now that $f \leq m-1$ (in case $f = m$ a restricted Huffman tree coincides with a Huffman tree). The construction in the next lemma closely follows the construction described in the binary case.

**Lemma 5.3.** *The trees $EHT(p_1, p_2, \ldots, p_L)$ and $ERHT(f; p_1, p_2, \ldots, p_L)$ can be constructed as follows:*

*A) If $L \geq m$ and $L \equiv 1 \pmod{m-1}$ we consider the terminal vertex $x$ in $\mathcal{T} = HT(p_1, p_2, \ldots, p_L)$ belonging to the maximum level $h(\mathcal{T})$, and having assigned a minimum weight $p = \min\{p_1, p_2, \ldots, p_L\}$. The vertex $x$ is transformed into an internal vertex, joined to a new son $y$ on the level $h(\mathcal{T})+1$; we associate to $y$ the weight $p$.*

*B) If $L > f$ and $L \equiv f \pmod{m-1}$, then the same construction is performed on the tree $RHT(f; p_1, p_2, \ldots, p_L)$.*

*Proof.* We prove only case B), as the first one is similar. Let $C$ be a prefix code over $M$ satisfying (10), and (a), (b) in **RHOP**; let $\mathcal{T}(C)$ be the $m$-ary tree associated with $C$. By Lemma 3.1 $\mathcal{T}(C)$ contains non-terminal vertices $x$ different from the root that have less than $m$ sons. As in the binary case, such a vertex $x$ belongs to the level $h(\mathcal{T}(C))-1$ (otherwise, $C$ would not be optimum). Now we assume that there exist two vertices $x$ and $y$ on the level $h(\mathcal{T}(C))-1$ such that $x$ has $n_1 \leq m-1$ sons, and $y$ has $n_2 \leq m-1$ sons on the level $h(\mathcal{T}(C))$. If $n_1 + n_2 \leq m$, then we take $n_1 - 1$ sons of $x$ and make them sons of $y$; the unique remaining son of $x$ is deleted and its weight is assigned to $x$ (which becomes a terminal vertex). A new extendible tree $\mathcal{T}_1$ is obtained and $cost(\mathcal{T}_1) < cost(\mathcal{T}(C))$, a contradiction. If $n_1 + n_2 \geq m+1$, then we move some sons of $x$ and make them sons of $y$ such that $y$ has now exactly $m$ sons on the level $h(\mathcal{T}(C))$. The tree $\mathcal{T}_2$ thus obtained has the same cost as $\mathcal{T}(C)$: $cost(\mathcal{T}_2) = cost(\mathcal{T}(C))$. By repeating this procedure we find an optimum tree $\mathcal{T}$ having a unique vertex $x$ on the level $h(\mathcal{T})-1$ such that $x$ has less than $m$ sons. If $x$ has $a \leq m-1$ sons, then

$$L - (a-1) \equiv f \pmod{m-1}.$$

Since $L \equiv f \pmod{m-1}$ we deduce that $a = 1$. From now on the proof goes on as in binary. $\square$

**Lemma 5.4.** *If $L > f$ and $f \le m-2$, then*

$$cost(ERHT(f; p_1, p_2, \ldots, p_L)) > cost(ERHT(f+1; p_1, p_2, \ldots, p_L)).$$

*Proof.* In $ERHT(f; p_1, p_2, \ldots, p_L)$ we take a terminal vertex lying on a level greater than one and make it son of the root. A new tree $\mathcal{T}$ with the root of degree $f+1$ is produced. This tree is extendible and

$$cost(ERHT(f; p_1, p_2, \ldots, p_L)) > cost(\mathcal{T}) \ge cost(ERHT(f+1; p_1, p_2, \ldots, p_L)).$$

□

Now we are able to propose an algorithm to generate an optimal extension of an optimal, non-complete, $m$-ary prefix code. Starting with weights $p_1, p_2, \ldots, p_L$ and proceeding as in binary, we build $\mathcal{T}_1 = EHT(p_1, p_2, \ldots, p_L)$ which generates an $m$-ary code $C = \{w_1, w_2, \ldots, w_L\}$ such that $\sum_{i=1}^{L} m^{-l_i} < 1$ and $\sum_{i=1}^{L} p_i l_i$ is minimum among all prefix codes of cardinality $L$. Suppose we want to extend $C$ with $s$ new words $w_{L+1}, \ldots, w_{L+s}$ having lengths $|w_i| = l_i$, for $L+1 \le i \le L+s$, such that $\sum_{i=1}^{L+s} m^{-l_i} < 1$, i.e. the new code is extendible again. If the new words have weights $p_{L+1}, \ldots, p_{L+s}$ we have to choose the word lengths $l_{L+1}, \ldots, l_{L+s}$ such that $\sum_{i=L+1}^{L+s} p_i l_i$ is minimum. We proceed as follows: let $x$ be the unique non-terminal vertex of $\mathcal{T}_1$ having $b \le m-1$ sons on the last level of $\mathcal{T}_1$. If $b = m-1$, then we build the tree $\mathcal{T}_2 = EHT(p_{L+1}, \ldots, p_{L+s})$ having the root $r$ and define $\mathcal{T}$ to be the union of $\mathcal{T}_1$ and $\mathcal{T}_2$, where $r$ becomes the $m$th son of $x$. In case $1 \le b \le m-2$ we construct $\mathcal{T}_2 = ERHT(m-b; p_{L+1}, \ldots, p_{L+s})$ with the root $r$ and define $\mathcal{T}$ by identifying $r$ with $x$ into a single vertex having $m$ sons.

**Theorem 5.5.** *The tree $\mathcal{T}$ is an optimal extension of $\mathcal{T}_1$ for every $m \ge 3$.*

*Proof.* In view of Lemma 5.4 and proceeding as in binary (see the proof of Theorem 4.2), we show that $\mathcal{T}$ is an extendible $m$-ary tree which extends $\mathcal{T}_1$ and achieves the minimum weighted length $\sum_{i=L+1}^{L+s} p_i l_i$. □

Notice that if we want $\mathcal{T}$ not to be extendible then, as in binary, we must choose $\mathcal{T}_2 = HT(p_{L+1}, \ldots, p_{L+s})$, respectively, $RHT(m-b; p_{L+1}, \ldots, p_{L+s})$, instead of $EHT(p_{L+1}, \ldots, p_{L+s})$, respectively, $ERHT(m-b; p_{L+1}, \ldots, p_{L+s})$.

## 6 Concluding Remarks

The extension discussed in this paper has been performed under the assumption that the initial prefix code is itself optimal and satisfies Kraft's strict inequality. It would be interesting to study the corresponding problem for an initial arbitrary extendible prefix code, satisfying only Kraft's strict inequality.

## Acknowledgement

This work has been done while the second author has visited the University of Auckland. The first author has been partially supported by AURC A18/XXXXX/62090/F3414030, 1994.

## References

1. J. Berstel, D. Perrin. *Theory of Codes*, Academic Press, New York, 1985.
2. C. Calude. *Information and Randomness: An Algorithmic Perspective*, Springer-Verlag, Berlin, 1994.
3. G. J. Chaitin. A theory of program size formally identical to information theory, *J. Assoc. Comput. Mach.* 22(1975), 329–340. (Reprinted in: G. J. Chaitin. *Information, Randomness and Incompleteness, Papers on Algorithmic Information Theory*, World Scientific, Singapore, 1987. (2nd ed., 1990), 197–223.)
4. G. J. Chaitin. *Algorithmic Information Theory*, Cambridge University Press, Cambridge, 1987. (third printing 1990)
5. T. M. Cover, J. A. Thomas. *Elements of Information Theory*, John Wiley, New York, 1991.
6. S. Even. *Algorithmic Combinatorics*, Macmillan, New York, 1973.
7. D. A. Huffman. A method for the construction of minimum-redundancy codes, *Proc. IRE* 40 (1952), 1098–1101.
8. L. G. Kraft. *A Device for Quantizing Grouping and Coding Amplitude Modulated Pulses*, MS Thesis, Electrical Eng. Dept., MIT, Cambridge, MA, 1949.