Fundamenta Informaticae 73 (2006) 107–117
IOS Press

Faster Algorithm for Designing Optimal Prefix-Free Codes with Unequal Letter Costs

Sorina Dumitrescu[*]
Department of Electrical and Computer Engineering
McMaster University, Hamilton, ON, Canada L8S 4K1
sorina@mail.ece.mcmaster.ca

Abstract. We address the problem of designing optimal prefix-free codes over an encoding alphabet with unequal integer letter costs. The most efficient algorithm proposed so far has $O(n^{C+2})$ time complexity, where n is the number of codewords and C is the maximum letter cost. For the special case when the encoding alphabet is binary, a faster solution was proposed, namely of $O(n^C)$ time complexity, based on a more sophisticated modeling of the problem, and on exploiting the Monge property of the cost function. However, those techniques seemed not to extend to the r-letter alphabet. This work proves that, on the contrary, the generalization to the r-letter case is possible, thus leading to a $O(n^C)$ time complexity algorithm for the case of arbitrary number of letters.

Keywords: Prefix-free codes, unequal letter costs, lopsided trees, optimization, Monge property.

1. Introduction

Assume messages are drawn from a source alphabet $\mathcal{A} = \{a_1, a_2, \cdots, a_n\}$, each source symbol a_i having probability $p_i > 0$, $\sum_{i=1}^{n} p_i = 1$. Let the encoding alphabet be $\Sigma = \{\alpha_1, \alpha_2, \cdots, \alpha_r\}$, where each letter α_i has a cost c_i. Assume the costs are integers, and $0 \le c_1 \le c_2 \le \cdots \le c_r = C$. The cost of a word $u = u_1 \cdots u_n$, $u_1, \cdots, u_n \in \Sigma$, is defined as $cost(u) = \sum_{i=1}^{n} cost(u_i)$, where $cost(u_i)$ is the cost of the letter u_i.

A code is a subset W of Σ^* together with a one-to-one mapping from $\mathcal{A}$ to W, which assigns to each source symbol a_i a codeword $w_i \in W$. A code W is prefix-free if no codeword is a prefix of another

[*]Address for correspondence: Department of Electrical and Computer Engineering, McMaster University, Hamilton, ON, Canada L8S 4K1

codeword. The cost of the code is

$$C(W) = \sum_{i=1}^{n} cost(w_i)p_i. \tag{1}$$

The Optimal Prefix-Free Coding Problem is the problem of finding a prefix-free code of minimum cost, given the probabilities p_i and the letters' costs c_i, $1 \leq i \leq n$.

Given a fix set W of n codewords, these codewords can be assigned in various ways to the source symbols a_i, thus yielding codes with different costs. Let us assume without restricting the generality that $p_1 \geq p_2 \geq \cdots \geq p_n > 0$. Then, as noted in [3], the minimal cost is obtained by an assignment such that $cost(w_1) \leq cost(w_2) \leq \cdots \leq cost(w_n)$. Therefore we will assume from now on that the assignment of codewords to source symbols is optimal for the given set of codewords, and consequently we will identify the code with the set W.

Any prefix-free code can be associated with an r-ary lopsided tree T (a tree where the edges have lengths). Each node of the tree can have at most r children. Each edge is labeled by a letter from the alphabet Σ, distinct edges that leave from the same node, having distinct labels. Any edge labeled by α_i has the length c_i. The *depth* of a node $v \in T$, denoted by $depth(v)$, is the sum of the lengths of the edges on the path connecting the root to that node. The root has depth 0. Any leaf v can be identified with the sequence of letters $u_1 u_2 \cdots u_s$, which labels the path from the root to v. Then $depth(v) = cost(u_1 u_2 \cdots u_s)$. Throughout the rest of the paper we will refer to r-ary lopsided trees whose edge lengths are $c_1, c_2, \cdots, c_r$, as $(c_1, c_2, \cdots, c_r)$-trees, or even simply, as trees, when the r-tuple of lengths is understood.

Any prefix-free code W of size n can be associated in a one-to-one manner to a tree whose leaves correspond to the codewords in W. If v_i is the leaf associated to the codeword w_i, then clearly $cost(W) = \sum_{i=1}^{n} depth(v_i)p_i$. Therefore, given a tree T we can define its cost with respect to the non-increasing sequence of probabilities $P = (p_1, p_2, \cdots, p_n)$, as $cost(T, P) = \sum_{i=1}^{n} depth(v_i)p_i$, where $v_1, v_2, \cdots, v_n$ are its leaves ordered in increasing order of depth.

Consequently, the optimization problem becomes equivalent to the following.

Problem 1. Given the r-tuple of integers $(c_1, c_2, \cdots, c_r)$ and the non-increasing sequence of probabilities $P = (p_1, p_2, \cdots, p_n)$, find the n-leaves $(c_1, c_2, \cdots, c_r)$-tree T_{opt}, of minimum cost with respect to P.

This problem was first studied by Karp in [5]. His solution is based on the problem formulation as an integer programming problem and has exponential time complexity in n. Golin and Rote [3] proposed a dynamic programming solution with polynomial time complexity in n, namely $O(n^{C+2})$, which is currently the most efficient algorithm for general r and small C. Shortly after, Bradford *et al.* [4] introduced a different approach for the case of binary trees ($r = 2$), yielding an $O(n^C)$ time algorithm to solve the problem. The authors of [4] explicitly stated that their method was limited to binary trees because the techniques employed seemed not to extend to the r-ary case. We prove in this paper that, on the contrary, the idea of [4] can be generalized to the r-ary case as well, thus leading to a $O(n^C)$ time solution for general r.

2. Recasting the Problem in Terms of Full Trees

For each $m > n$ denote by P_m the sequence of m values obtained by padding P with $m - n$ zeros at the end. A tree is called *full* tree, if any internal node has the full set of r children. Denote by $T(m)$ the minimum cost full tree among all full trees of m leaves, with respect to the sequence P_m. In [3] it was shown that Problem 1 can be recast as the problem of finding the full tree T^*_{opt} with m_0 leaves such that $n \leq m_0 \leq n(r-1)$ and

$$cost(T^*_{opt}, P_{m_0}) = \min_{m, n \leq m \leq n(r-1)} \{cost(T, P_m) \mid T \text{ is full tree with } m \text{ leaves}\}. \quad (2)$$

T_{opt} is further obtained from T^*_{opt} by peeling away the 0-probability leaves (i.e., the $m_0 - n$ deepest leaves).

We make the observation that in (2) it is not necessary to search for the optimal m_0 in the whole range between n and $n(r-1)$, but it is enough to search among the full trees with $(n-1)(r-1)+1$ leaves and those with n leaves, if such full trees exist. This observation is justified by the following lemma.

Lemma 2.1. If the optimal tree T_{opt} is not a full tree, then

$$cost(T_{opt}, P) = \min\{cost(T, P_m) \mid T \text{ is full tree with } m \text{ leaves}\}, \quad (3)$$

where $m = (n-1)(r-1)+1$.

Proof:
Let $full(T_{opt})$ be the full tree obtained by completing the missing children of the internal nodes of T_{opt}. Let ℓ be the number of internal nodes of T_{opt}. Then $\ell \leq n-1$ ([3]). Since T_{opt} is not full itself, then at least one leaf of $full(T_{opt})$ is not leaf in T_{opt}. If $\ell = n-1$ set $T' = full(T_{opt})$. Otherwise, construct a full tree with $n-1-\ell$ internal nodes. By replacing leaf v with this tree, a full tree T' with $n-1$ internal nodes is obtained. Then T' has $(n-1)(r-1)+1$ leaves. The set of leaves of T' contains the set of leaves of T_{opt}, and some new leaves. Label the new leaves by $v_{n+1}, \cdots, v_m$. Then $cost(T', P_m) \leq \sum_{i=1}^{m} depth(v_i)p_i = \sum_{i=1}^{n} depth(v_i)p_i = cost(T_{opt}, P)$, which implies

$$cost(T_{opt}, P) \geq \min\{cost(T, P_m) \mid T \text{ is full tree with } m \text{ leaves}\}. \quad (4)$$

On the other side, let T' be an arbitrary full tree with m leaves. Let T be the tree with n leaves obtained from T' by removing the deepest $m-n$ leaves and the unnecessary internal nodes (i.e., the internal nodes which are descendants of some removed leaves, but are not descendants of any retained leaf). Since the leaves of T are exactly the leaves of T' assigned to non-zero probabilities, it follows that $cost(T', P_m) = cost(T, P)$, which further implies

$$cost(T_{opt}, P) \leq \min\{cost(T, P_m) \mid T \text{ is full tree with } m \text{ leaves}\}. \quad (5)$$

Relations (4) and (5) prove the lemma. □

According to the above lemma, Problem 1 can be solved as follows:

1) If $n-1$ is not a multiple of $r-1$, find the full tree $T(m)$ of m leaves $m=(n-1)(r-1)+1$, of minimal cost with respect to P_m. Then construct T_{opt} from $T(m)$ by removing the $m-n$ deepest leaves and the unnecessary internal nodes.
2) If $n-1$ is a multiple of $r-1$, then find $T(m)$ as above, and find the full tree $T(n)$ of n leaves of minimal cost with respect to P. If $cost(T(n),P) \leq cost(T(m),P_m)$ then set $T_{opt}=T(n)$. Otherwise, construct T_{opt} from $T(m)$ as above.

Therefore, in order to prove our complexity claim, it is enough to show that the optimal m-leaves full tree $T(m)$ can be found in $O(n^C)$ time. To this aim, we will follow the general idea of [4]. Namely, we will recast the problem in terms of the so-called monotonic sequences, step which will account for a linear factor decrease in time complexity, and further use a property of the cost function, known as the Monge property, which allows for another linear factor decrease in complexity.

3. Problem Formulation in Terms of Monotonic Sequences

From now on we will only consider full trees, and will refer to them simply as trees. The r children of an internal node v of a tree are ordered from left to right in increasing order of the edge length from v, and we will refer to them by using their position in this sequence. Therefore, the length from v to its j-th child is c_j. The rightmost child of v is its r-th child. The depth of the tree T, denoted by $depth(T)$, is the largest depth of all nodes. A node v of the tree T is said to be at level i if $i=depth(T)-depth(v)$ (i.e., v is i levels far from the bottom of the tree).

The following sequences were introduced in [4]. Occasionally we use slightly different notation.

The *numbers-of-leaves sequence* of a tree T is $\Delta(T)=(\delta_0(T),\delta_1(T),\cdots,\delta_{d-1}(T))$, where d is the depth of T and $\delta_i(T)$ is the number of leaves below or at the level i, for $1\leq i\leq d-1$.

The *characteristic sequence* of a tree T is $\Gamma(T)=(\gamma_0(T),\gamma_1(T),\cdots,\gamma_{d-1}(T))$, where $d=depth(T)$ and $\gamma_i(T)$ is the number of right children at or below level i, for $1\leq i\leq d-1$. Clearly, since T is a $(c_1,c_2,\cdots,c_r)$-tree, the last c_r components of the sequence are all equal because the highest level where a rightmost child can appear is $d-c_r$.

For any positive integer k, define the set $\mathcal{M}_k$ of *k-ended monotonic sequences* as the set of all finite sequences with at least c_r components, whose components are non-negative integers, in nondecreasing order, and the last c_r components are equal to k. In other words $\mathcal{M}_k$ is the set of sequences $B=(b_0,\cdots,b_{d-1})$ for some $d\geq c_r$, and $0\leq b_0\leq b_1\leq\cdots\leq b_{d-c_r}=\cdots=b_{d-1}=k$.

Let T be a tree of m leaves and let $Q=(q_1,\cdots,q_m)$ be a non-increasing sequence of m probabilities (i.e., $q_1\geq\cdots\geq q_m\geq 0$, $\sum_{i=1}^{n}=1$) . Clearly, the cost $cost(T,Q)$ depends only on the number of leaves at each level of T and on the sequence of probabilities Q. Moreover, as proved in [4] $cost(T,Q)$ can be expressed in terms of the sequences $\Delta(T)$ and Q.

Definition 3.1. For $i, 1\leq i\leq m$, let $Suf_i(Q)$ denote the sum of the last i values in the sequence Q, i.e., $Suf_i(Q)=\sum_{j=m-i+1}^{m}q_j$. For $i>m$, let $Suf_i(Q)=\infty$.

The following equality was proved in [4] (even if the referenced paper treats only the case of binary trees, the proof is valid for r-ary trees):

$$cost(T,Q)=\sum_{k=0}^{depth(T)-1} Suf_{\delta_k(T)}(Q). \tag{6}$$

Further we show that the number-of-leaves sequence of a tree can be obtained from its characteristic sequence.

Lemma 3.1. For any tree T and for all $i, 0 \leq i \leq depth(T) - 1$, the following relation holds

$$\delta_i(T) = \sum_{j=1}^{r} \gamma_{i+c_j-c_r}(T) - \gamma_{i-c_r}(T), \tag{7}$$

where, by convention, $\gamma_k(T) = 0$ if $k < 0$.

Proof:
Let us fix some $i, 0 \leq i \leq depth(T) - 1$. The nodes at or below level i, together with the connecting edges, form a forest (i.e., a union of trees), denoted by $\mathcal{F}_i$. The quantity $\delta_i(T)$ equals the number of leaves of this forest, which further equals $n_i + (r-1)int_i$, where n_i is the number of trees of forest $\mathcal{F}_i$, and int_i is the number of its internal nodes.

Clearly, int_i coincides with the number of internal nodes of T, situated at or below level i. Since for each k there is a one-to-one correspondence between internal nodes at level k and the rightmost children situated at level $k - c_r$, it follows that $int_i = \gamma_{i-c_r}(T)$.

Let us identify now the number of trees n_i. The roots of these trees are those nodes of T whose parents are above level i. The set of roots coincides with the union $\cup_{1 \leq j \leq r} \mathcal{V}_j$, where $\mathcal{V}_j$ denotes the set of nodes at or below level i, which are the j-th child of some node above level i, for all $1 \leq j \leq r$. Clearly, the parent of some node in $\mathcal{V}_j$ is an internal node situated at a level between $i+1$ and $i+c_j$ (including both). Conversely, any internal node situated at some level between $i+1$ and $i+c_j$ (including both) has its j-th child below or at level i. Therefore, $\mid \mathcal{V}_j \mid = int_{i+c_j} - int_i = \gamma_{i+c_j-c_r}(T) - \gamma_{i-c_r}(T)$. It follows that $n_i = \sum_{j=1}^{r} \mid \mathcal{V}_j \mid = \sum_{j=1}^{r}(\gamma_{i+c_j-c_r}(T) - \gamma_{i-c_r}(T)) = \sum_{j=1}^{r} \gamma_{i+c_j-c_r}(T) - r\gamma_{i-c_r}(T)$. Thus,

$$\delta_i(T) = \sum_{j=1}^{r} \gamma_{i+c_j-c_r}(T) - r\gamma_{i-c_r}(T) + (r-1)\gamma_{i-c_r}(T) = \sum_{j=1}^{r} \gamma_{i+c_j-c_r}(T) - \gamma_{i-c_r}(T). \tag{8}$$

□

Relation (6) together with Lemma 3.1 imply that the optimization problem can be recast in terms of characteristic sequences rather than trees. Note first that if T is a tree with $n-1$ internal nodes (i.e., with $m = 1 + (r-1)(n-1)$ leaves), then $\Gamma(T) \in \mathcal{M}_{n-1}$. However, not any $(n-1)$-ended monotonic sequence $B \in \mathcal{M}_{n-1}$ is the characteristic sequence of some tree with $n-1$ internal nodes. On the other side, following the idea of [4] we will define a cost for any $(n-1)$-ended monotonic sequence, and we will show that the minimum cost over $(n-1)$-ended monotonic sequences coincides with the minimum cost over m-leaves trees.

Definition 3.2. For any $n \geq 2$, and any $B = (b_0, b_1, \cdots, b_{d-1})$ monotonic sequence in $\mathcal{M}_{n-1}$, denote $N_k(B) = \sum_{j=1}^{r} b_{k+c_j-c_r} - b_{k-c_r}$, for all $k, 0 \leq k \leq d-1$, where, by convention, $b_j = 0$ if $j < 0$.

Let $m = 1 + (r-1)(n-1)$. For any non-increasing sequence of m probabilities $Q = (q_1, \cdots, q_m)$, define the cost of B with respect to Q as $cost(B, Q) = \sum_{k=0}^{d-1} Suf_{N_k(B)}(Q)$.

Remark 3.1. It is easy to check that by adding leading zeros to a monotonic sequence, its cost does not change.

Clearly, applying the above definition to the characteristic sequence of a tree T, we obtain that $cost(\Gamma(T), Q) = cost(T, Q)$, which further implies

$$\min_{B \in \mathcal{M}_{n-1}} cost(B, Q) \leq \min_{T \in \mathcal{T}_{n-1}} cost(T, Q). \tag{9}$$

Denote by $\mathcal{T}_k$ the set of trees with k internal nodes. The following proposition is essential to our development.

Proposition 3.1. For any $n \geq 2$ and $Q = (q_1, \cdots, q_m)$, where $m = (n-1)(r-1)+1$,

$$\min_{B \in \mathcal{M}_{n-1}} cost(B, Q) \geq \min_{T \in \mathcal{T}_{n-1}} cost(T, Q). \tag{10}$$

In order to prove the above proposition, the following two lemmas are needed.

Lemma 3.2. There is a minimum cost $(n-1)$-ended monotonic sequence $B_{opt} = (b_0, b_1, \cdots, b_{d-1})$ such that $b_i \neq b_{i+c_r}$ for all $i, 1 \leq i \leq d-1-c_r$.

Proof:
Let $B_{opt} = (b_0, b_1, \cdots, b_{d-1})$ be optimal and assume that there is some i such that $b_i = b_{i+c_r}$ for some i. Then $b_i = b_{i+1} = \cdots = b_{i+c_r}$. Construct B' by deleting b_{i+c_r} from the sequence B_{opt}. In other words, $B' = (b'_0, \cdots, b'_{d-2})$, where $b'_j = b_j$ for $j \leq i + c_r - 1$, and $b'_j = b_{j+1}$ for $j \geq i + c_r$. Clearly, $B' \in \mathcal{M}_{n-1}$, too. Since $b_i = b_{i+1} = \cdots = b_{i+c_r}$, it also follows that $b'_j = b_{j+1}$ for $j \geq i$.

Further we have $N_k(B') = \sum_{j=1}^{r} b'_{k+c_j-c_r} - b'_{k-c_r}$. When $k \leq i + c_r - 1$, $N_k(B') = N_k(B_{opt})$. When $k \geq i + c_r$, we have $k + c_j - c_r \geq i$ and $k - c_r \geq i$, therefore, $N_k(B') = N_{k+1}(B_{opt})$. Thus, $cost(B_{opt}, Q) = cost(B', Q) + Suf_{N_{i+c_r}}(Q)$. Since $Suf_{N_{i+c_r}}(Q) \geq 0$, it follows that $cost(B', Q) \leq cost(B_{opt}, Q)$. Therefore, B' is optimal, too. □

Lemma 3.3. For all $l, 1 \leq l \leq c_r$, denote $n_l = |\ \{j \mid c_j = l\}\ |$. For any $(n-1)$-ended monotonic sequence $B = (b_0, b_1, \cdots, b_{d-1})$, with $b_0 \geq 1$, we have $N_0(B) \geq n_{c_r}$ and

$$N_{c_r - l}(B) - N_{c_r - l - 1}(B) \geq n_l, \tag{11}$$

for all $l, 1 \leq l \leq c_r - 1$.

Proof:
For each $k, 0 \leq k \leq c_r - 1$, by removing the terms with negative subscripts (which are 0 by convention) from the expression of $N_k(B)$ in Definition 3.2, we obtain

$$N_k(B) = \sum_{j, 1 \leq j \leq r, c_j \geq c_r - k} b_{k+c_j-c_r}. \tag{12}$$

By replacing c_j by t we get further

$$N_k(B) = \sum_{t=c_r-k}^{c_r} b_{t+k-c_r} n_t. \tag{13}$$

Then $N_0(B) = b_0 n_{c_r} \geq n_{c_r}$. Further, for $l, 1 \leq l \leq c_r - 1$, we have

$$N_{c_r-l}(B) - N_{c_r-l-1}(B) = \sum_{t=l}^{c_r} b_{t-l} n_t - \sum_{t=l+1}^{c_r} b_{t-l-1} n_t = b_0 n_l + \sum_{t=l+1}^{c_r} (b_{t-l} - b_{t-l-1}) n_t. \quad (14)$$

The last sum is non-negative because $b_{t-l} \geq b_{t-l-1}$. Since $b_0 \geq 1$, the conclusion follows. □

We are prepared now to present the proof of Proposition 3.1.

Proof of Proposition 3.1:
We will give the proof by induction on n.
Base case. Let $n = 2$. Any 1-ended monotonic sequence without leading 0's, has all components equal to 1. Further, Lemma 3.3 implies that the sequence with exactly c_r components, all equal to 1 has minimum cost. This monotonic sequence is the characteristic sequence of the tree with one internal node.
Inductive step. Assume Proposition 3.1 is satisfied for $n - 1$. We will show that it is satisfied for n too. Let $B_{opt} = (b_0, \cdots, b_{d-1}) \in \mathcal{M}_{n-1}$ be the $(n-1)$-ended monotonic sequence of minimum cost with respect to Q. Assume all possible leading zeros have been removed. According to Remark 3.1, by removing leading zeros the cost is not affected. The optimality of the sequence B_{opt} implies that $m \geq N_k(B_{opt})$ for each $k, 0 \leq k \leq d-1$ (otherwise $Suf_{N_k(B_{opt})}(Q)$ would be ∞). Moreover, Lemma 3.3 implies that

$$m - N_{c_r-1}(B_{opt}) \leq m - N_{c_r-2}(B_{opt}) \leq \cdots \leq m - N_0(B_{opt}), \quad (15)$$

and that, for $j, 1 \leq j \leq r$, with $c_j < c_r$, we have

$$m - N_{c_r-c_j}(B_{opt}) + n_{c_j} \leq m - N_{c_r-c_j-1}(B_{opt}), \quad (16)$$

and for $j, 1 \leq j \leq r$, with $c_j = c_r$, we have

$$m - N_{c_r-c_j}(B_{opt}) + n_{c_j} \leq m. \quad (17)$$

Construct $k_1, k_2, \cdots k_r$ as follows. For each $j, 1 \leq j \leq r$, let i_j denote the minimal i such that $c_i = c_j$. Then define $k_{i_j} = m - N_{c_r-c_j}(B_{opt}) + 1$ and $k_{i_j+i} = k_{i_j} + i$ for all $i, 1 \leq i \leq n_{c_j} - 1$. Relation (16) implies that

$$m - N_{c_r-c_j}(B_{opt}) + 1 \leq k_j \leq m - N_{c_r-c_j-1}(B_{opt}), \quad (18)$$

for $1 \leq j \leq r$ such that $c_j < c_r$, and

$$m - N_{c_r-c_j}(B_{opt}) + 1 \leq k_j \leq m, \quad (19)$$

for $1 \leq j \leq r$ such that $c_j = c_r$. Corroborating further with (15) we obtain that $1 \leq m - N_{c_r-1}(B_{opt}) < k_1 < k_2 < \cdots < k_r \leq m$.

Let $q' = q_{k_1} + \cdots + q_{k_r}$ and choose k_0 such that

$$q_{k_0-1} > q' \geq q_{k_0}. \quad (20)$$

Since $q' \geq q_{k_1}$, and the sequence Q is non-increasing, it follows that $k_0 \leq k_1$.

Construct the sequence Q' of $m' = m - (r-1)$ nonnegative values by applying the following list operations to the list Q: delete the entries $q_{k_1}, \cdots, q_{k_r}$, and insert the new value q' at position k_0. Then Q' is also sorted in non-increasing order, and the sum of all its elements equals the sum of elements of Q, i.e., 1. Moreover, denote by $Pref_i(Q)$, $Pref_i(Q')$ the sum of the first i elements in the sequence Q, respectively Q'. After a moment of thought it can be seen that

$$Pref_i(Q') \geq Pref_i(Q) \tag{21}$$

for all $i, 0 \leq i \leq m - (r-1)$.

Consider now the $(n-2)$-ended monotonic sequence $B' = (b'_0, \cdots, b'_{d-1}) \in \mathcal{M}_{n-2}$ where $b'_k = b_k - 1$ for all $k, 1 \leq k \leq d-1$. Because the number of elements in the sequence Q' is $m' = (n-2)(r-1)+1$ and the sequence is non-increasing, $cost(B', Q')$ is well defined according to Definition 3.2.

Next we will show that

$$cost(B_{opt}, Q) \geq cost(B', Q') + \sum_{j=1}^{r} q_{k_j} c_j. \tag{22}$$

For this we will investigate the relation between $Suf_{N_l(B')}(Q')$ and $Suf_{N_l(B_{opt})}(Q)$ for all $l, 1 \leq l \leq d-1$.

Case 1: $l \geq c_r$. In this case we have $N_l(B') = \sum_{j=1}^{r} b'_{l-(c_r-c_j)} - b'_{l-c_r} = \sum_{j=1}^{r}(b_{l-(c_r-c_j)} - 1) - (b_{l-c_r} - 1) = N_l(B_{opt}) - (r-1)$. Then

$$\begin{aligned} Suf_{N_l(B')}(Q') = 1 - Pref_{m-r+1-N_l(B')}(Q') = 1 - Pref_{m-N_l(B_{opt})}(Q') \leq \\ 1 - Pref_{m-N_l(B_{opt})}(Q) = Suf_{N_l(B_{opt})}(Q). \end{aligned} \tag{23}$$

The inequality in the above sequence of relations follows from (21).

Case 2: $0 \leq l \leq c_r - 1$. It is more convenient to write $l = c_r - s$, where $1 \leq s \leq c_r$. Then the following sequence of equalities follows by using relation (13): $N_l(B') = N_{c_r-s}(B') = \sum_{t=s}^{c_r} b'_{t-s} n_t = \sum_{t=s}^{c_r} b_{t-s} n_t - \sum_{t=s}^{c_r} n_t = N_{c_r-s}(B_{opt}) - \sum_{t=s}^{c_r} n_t$. Denote by j_s the smallest j such that $c_j \geq s$. In other words, $j_s = \sum_{t=1}^{s-1} n_t + 1$. According to (15), (18) and (19), j_s is the smallest j such that $m - N_{c_r-s}(B_{opt}) < k_j$. We further distinguish between two subcases.

Subcase 2.a: $k_0 \leq m - N_{c_r-s}(B_{opt}) + 1$. Then the last $N_{c_r-s}(B) - \sum_{t=s}^{c_r} n_t$ elements of Q' are obtained from the last $N_{c_r-s}(B_{opt})$ elements of Q, by removing $q_{k_{j_s}}, \cdots, q_{k_r}$. Therefore,

$$Suf_{N_l(B')}(Q') = Suf_{N_l(B_{opt})}(Q) - (q_{k_{j_s}} + \cdots + q_{k_r}). \tag{24}$$

Subcase 2.b: $m - N_{c_r-s}(B_{opt}) + 1 < k_0$. Then the last $N_{c_r-s}(B_{opt}) - \sum_{t=s}^{c_r} n_t$ elements of Q' are obtained from the last $N_{c_r-s}(B_{opt}) - 1$ elements of Q by removing $q_{k_{j_s}}, \cdots, q_{k_r}$, and adding q'. Since $q' < q_{m-N_{c_r-s}(B_{opt})+1}$ it follows that

$$Suf_{N_l(B')}(Q') \leq Suf_{N_l(B_{opt})}(Q) - (q_{k_{j_s}} + \cdots + q_{k_r}). \tag{25}$$

Summarizing we obtain

$$\begin{aligned} cost(B', Q') = \textstyle\sum_{l=0}^{d-1} Suf_{N_l(B')}(Q') \leq \\ \textstyle\sum_{l=0}^{d-1} Suf_{N_l(B_{opt})}(Q) - \sum_{s=1}^{c_r} \sum_{j=j_s}^{r} q_{k_j} = \\ cost(B_{opt}, Q) - \textstyle\sum_{s=1}^{c_r} \sum_{j=j_s}^{r} q_{k_j} = \\ cost(B_{opt}, Q) - \textstyle\sum_{j=1}^{r} q_{k_j} c_j, \end{aligned} \tag{26}$$

which proves (22).

Let now T'_{opt} be the tree with $n-2$ internal nodes such that $cost(T'_{opt}, Q') = \min_{T' \in \mathcal{T}_{n-2}} cost(T', Q')$. According to the inductive hypothesis, we have

$$cost(B', Q') \geq cost(T'_{opt}, Q'). \tag{27}$$

Now construct the tree T by transforming the leaf corresponding to q' into an internal node whose all children are leaves. Thus T is a tree with $n-1$ internal nodes. By assigning the probabilities $q_{k_1}, \cdots, q_{k_r}$ to the new leaves, and keeping the old assignments of probabilities for the old leaves, possibly a suboptimal assignment of probabilities of Q to the leaves of T is obtained, whose cost is $cost(T'_{opt}, Q') + \sum_{j=1}^{r} q_{k_j} c_j$. This implies that

$$cost(T, Q) \leq cost(T'_{opt}, Q') + \sum_{j=1}^{r} q_{k_j} c_j. \tag{28}$$

Finally, relations (22), (27) and (28) lead to

$$cost(T, Q) \leq cost(B_{opt}, Q), \tag{29}$$

which concludes the inductive step and the proof. □

Proposition 3.1 together with (9) show that the minimum cost tree $T(m)$ of m leaves (or $n-1$ internal nodes) can be constructed by first finding the $(n-1)$-ended monotonic sequence of minimum cost, B_{opt}, and then applying the recursive procedure $OptTree(n, B_{opt}, Q)$ described as follows:

$OptTree(n, B_{opt}, Q)$
1) Identify $k_0, k_1, \cdots, k_r$.
2) Construct Q' from Q as described in the proof of Proposition 3.1.
3) Construct B' by decrementing by 1 each component of the sequence B_{opt}.
4) If $n = 2$ set T' to be the tree with one internal node. If $n \neq 2$ set T' to be $OptTree(n-1, B', Q')$.
5) Build T'' from T' by transforming the leaf corresponding to probability q'_{k_0} into an internal node with r children. Assign probabilities $q_{k_1}, \cdots, q_{k_r}$ to the new leaves.
6) Return T''.

It is easy to see that $T(m)$ can be constructed from B_{opt} as described above in $O(n^2)$ time.

4. $O(n^C)$ Time Algorithm for Finding the Minimum Cost Monotonic Sequence

Construct the weighted directed acyclic graph $G = (V, E)$, where the set of vertices is $V = \{(u_0, u_1, \cdots, u_{c_r-1}) \mid 0 \leq u_0 \leq u_1 \leq \cdots \leq u_{c_r-1} \leq n-1\}$, and the set E of edges consists of all ordered pairs of vertices $[(u_0, u_1, \cdots, u_{c_r-1}), (u_1, \cdots, u_{c_r-1}, u_{c_r})]$ such that $u_0 \neq u_{c_r}$. Such an edge will be simply denoted by $e(u_0, u_1, \cdots, u_{c_r})$. The weight of the edge $e(u_0, u_1, \cdots, u_{c_r})$ is

$$\omega(u_0, u_1, \cdots, u_{c_r}) = Suf_{\sum_{j=1}^{r} u_{c_j} - u_0}(Q) = Suf_{\sum_{t=1}^{c_r} u_t n_t - u_0}(Q). \tag{30}$$

Let the source of the graph be the vertex with all components 0, and the let the final node be the vertex with all components $n-1$. Let $B = (b_0, b_1, \cdots, b_{d-1})$ be an arbitrary $(n-1)$-ended monotonic sequence with no leading 0's (i.e., $b_1 \geq 1$) and with the additional property that $b_i \neq b_{i+c_r}$ for all i. Denote by $Path(B)$ the following path in the graph G, from the source to the final node

$$\begin{aligned}(0,0,\cdots,0,0,0) \to (0,0,\cdots,0,0,b_0) \to (0,0,\cdots,0,b_0,b_1) \to \cdots \to \\ (b_0,b_1,\cdots,b_{c_r-1}) \to (b_1,b_2,\cdots,b_{c_r}) \to \cdots \to \\ (b_i,b_{i+1},\cdots,b_{i+c_r-1}) \to (b_{i+1},b_{i+2},\cdots,b_{i+c_r}) \to \cdots \to \\ (b_{d-c_r-1},n-1,\cdots,n-1) \to (n-1,n-1,\cdots,n-1). \end{aligned} \quad (31)$$

It is easy to check that the weight of the above path (i.e., the sum of the weights of its edges) equals $cost(B,Q)$. Moreover, the mapping $Path(\cdot)$ defines a one-to-one correspondence between the $(n-1)$-ended monotonic sequences with no leading 0's (i.e., $b_1 \geq 1$) and with the additional property that $b_i \neq b_{i+c_r}$ for all i, and the paths in G from the source to the final node. Lemma 3.2 implies that there exists a minimum cost monotonic sequence with the property mentioned above. Therefore, finding an optimal sequence reduces to solving the shortest path problem in the graph G.

Note that the graph has $O(n^{c_r})$ vertices and $O(n^{c_r+1})$ edges, consequently, the shortest path problem can be solved by standard algorithms in $O(n^{c_r+1}) = O(n^{C+1})$ time. In order to solve it faster we start from the dynamic programming solution and further show that it can be sped up by using the fast matrix search technique in totally monotone matrices introduced in [1].

For each vertex $(u_0,u_1,\cdots,u_{c_r-1})$ denote by $\bar{\omega}(u_0,u_1,\cdots,u_{c_r-1})$ the weight of the minimum path from the source to that vertex.

For each (c_r-1)-tuple $\mathbf{u} = (u_1,\cdots,u_{c_r-1})$ with $0 \leq u_1 \leq \cdots \leq u_{c_r-1} \leq n-1$, consider the matrix $A_{\mathbf{u}}$ with elements $A(u_0,u_{c_r})$, $0 \leq u_0 \leq u_1$, $u_{c_r-1} \leq u_{c_r} \leq n-1$, defined as follows: $A(u_0,u_{c_r}) = \bar{\omega}(u_0,\mathbf{u}) + \omega(u_0,\mathbf{u},u_{c_r})$. Then, for any u_{c_r}, $u_{c_r-1} \leq u_{c_r} \leq n-1$, we have

$$\bar{\omega}(\mathbf{u},u_{c_r}) = \min_{u_0, 0 \leq u_0 \leq u_1, u_0 \neq u_{c_r}} A(u_0,u_{c_r}). \quad (32)$$

This implies that finding $\bar{\omega}(\mathbf{u},u_{c_r})$ for all u_{c_r} is finding all column minima in the matrix A. This problem would be normally solved in $O(n^2)$ time. However, there are situations when it can be solved faster. Such a situation is the case of totally monotone matrices defined in [1], for which all column minima can be solved in $O(n)$ time as shown in [1]. The matrix A is said to be totally monotone if the following implication holds

$$A(u_0,u_{c_r}) \geq A(u_0',u_{c_r}) \Rightarrow A(u_0,u_{c_r}') \geq A(u_0',u_{c_r}') \quad (33)$$

for all integers $0 \leq u_0 < u_0' \leq u_1$, $u_{c_r-1} \leq u_{c_r} < u_{c_r}' \leq n-1$.

It is known that the total monotonicity for matrix A is satisfied if the following property holds (also known as the Monge property) [2]:

$$A(u_0,u_{c_r}) + A(u_0+1,u_{c_r}+1) \leq A(u_0+1,u_{c_r}) + A(u_0,u_{c_r}+1) \quad (34)$$

for all u_0 and u_{c_r}. Ref (34) is equivalent to

$$\begin{aligned}&\bar{\omega}(u_0,\mathbf{u}) + \omega(u_0,\mathbf{u},u_{c_r}) + \bar{\omega}(u_0+1,\mathbf{u}) + \omega(u_0+1,\mathbf{u},u_{c_r}+1) \\ &\leq \bar{\omega}(u_0+1,\mathbf{u}) + \omega(u_0+1,\mathbf{u},u_{c_r}) + \bar{\omega}(u_0,\mathbf{u}) + \omega(u_0,\mathbf{u},u_{c_r}+1),\end{aligned} \quad (35)$$

further equivalent to

$$Suf_{\sum_{t=1}^{c_r} u_t n_t - u_0}(Q) + Suf_{\sum_{t=1}^{c_r} u_t n_t + n_{c_r} - u_0 - 1}(Q) \leq Suf_{\sum_{t=1}^{c_r} u_t n_t - u_0 - 1}(Q) + Suf_{\sum_{t=1}^{c_r} u_t n_t + n_{c_r} - u_0}(Q). \quad (36)$$

Denote $\alpha = \sum_{t=1}^{c_r} u_t n_t - u_0$. Then (36) can be written as

$$Suf_{\alpha}(Q) + Suf_{\alpha+n_{c_r}-1}(Q) \leq Suf_{\alpha-1}(Q) + Suf_{\alpha+n_{c_r}}(Q), \quad (37)$$

which is equivalent to $q_{m-\alpha+1} \leq q_{m-\alpha-n_{c_r}+1}$,which is true.

We conclude that the fast matrix search technique of [1] can be applied to solve (32) for given $\mathbf{u}$ and all u_{c_r}, in $O(n)$ time. By processing all $(c_r - 1)$-tuples $\mathbf{u}$ in lexicographical order, the shortest path can be computed in $O(n^{c_r}) = O(n^C)$ time.

References

[1] A. Aggarwal, M. Klave, S. Moran, P. Shor, R. Wilber: Geometric applications of a matrix-searching algorithm. *Algorithmica*, 2 (1987), 195–208.

[2] A. Apostolico, Z. Galil, eds.: *Pattern Matching Algorithms*. Oxford Univ. Press., New York, 1997.

[3] M. Golin, G. Rote: A dynamic programming algorithm for constructing optimal prefix-free codes for unequal letter costs. *IEEE Trans. Inform. Th.*, 44 (1998), 1770–1781.

[4] P. Bradford, M. Golin, L. Larmore, W. Rytter: Optimal prefix-free codes for unequal letter costs and dynamic programming with the Monge property. *J. Algorithms*, 42 (2002), 277–303.

[5] R.M. Karp: Minimum-redundancy coding for the discrete noiseless channel. *IRE Transactions on Inform. Th.*, 7 (1961), 27–39.