Lopsided Trees: Analyses, Algorithms, and Applications

Vicky Choi *   Mordecai J. Golin †

February 9, 2000

Abstract

Lopsided trees are rooted, ordered, trees in which the length of an edge from a node to its i^{th} child depends upon the value of i. These trees model a variety of problems and have therefore been extensively studied. In this paper we combine analytic and combinatorial techniques to address three open problems on such trees:

- Given n, efficiently construct a lopsided tree with n leaves that has minimal external-path-length.
- Express the cost of the minimal external-path-length tree as a function of n.
- Calculate exactly how many nodes of depth $\le x$ exist in the infinite lopsided tree.

Lopsided trees model *Varn codes*, prefix free codes in which the letters of the encoding alphabet can have different lengths. The solutions to the first and second problems above solve corresponding open problems on Varn codes. The solution to the third problem can be used to model the performance of broadcasting algorithms in the postal model of communication. Finding these solutions requires generalizing the definition of Fibonacci numbers and then using Mellin-transform techniques.

Keywords: Varn Codes, Fibonacci Recurrences, Mellin Transforms, Postal Model.

1 Introduction

In this paper we discuss some properties of lopsided trees. A tree is said to be *lopsided* if it is a rooted, ordered (i.e., the children of each node are ordered) tree with maximum arity r, in which the length of an edge from a parent to its i^{th} child is c_i where $c_1 \le c_2 \le \cdots \le c_r$ are r fixed positive reals. Figure 2 illustrates two finite lopsided trees, figure 3 illustrates an infinite one. The name *lopsided trees* was only coined in 1989 by Kapoor and Reingold [16] but the trees themselves have been implicitly present in the literature at least since 1961 when Karp [17] used them to model minimum-cost prefix-free (Huffman) codes in which the length of the edge of the letters in the encoding alphabet were unequal; c_i represented the length of the i^{th} letter in the encoding alphabet (the *idea* of such codes was already present in Shannon's seminal paper on communication theory [27]). Such trees were later used in [13] and [3] to design more efficient algorithms for the same problem.

For fixed $c_1 \le c_2 \le \cdots \le c_r$ we study three problems on these trees:

*Rutgers University. email:VCHOI@PAUL.RUTGERS.EDU.

†Department of Computer Science, Hong Kong University of Science, and Technology, Clear Water Bay, Kowloon, Hong Kong. email: GOLIN@CS.UST.HK.

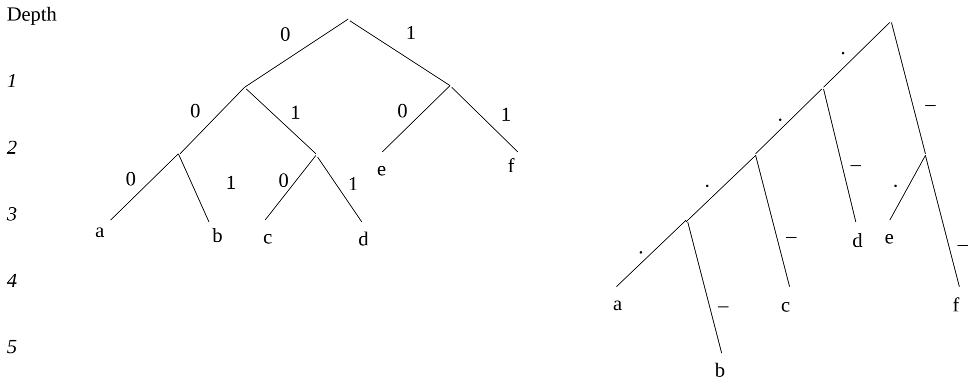


Figure 1: Two trees with 6 leaves (labelled a,b,c,d,e,f). The tree on the left has $(c_1, c_2) = (1, 1)$. It corresponds to the prefix-free code

$$a = 000, \quad b = 001, \quad c = 011, \quad d = 011, \quad e = 10, \quad f = 11$$

for alphabet $\Sigma = \{0, 1\}$ when 0 and 1 have the same length. The cost of the code and corresponding tree is $3 + 3 + 3 + 2 + 2 = 13$. The tree on the right has $(c_1, c_2) = (1, 2)$. It corresponds to the prefix-free code

$$a = \ldots., \quad b = \ldots.\_, \quad c = ..\_, \quad d = .\_, \quad e = \_., \quad f = \_\_$$

when $\Sigma = \{., \_\}$ and $length(.) = 1$, $length(\_) = 2$. The cost of the code and corresponding tree is $4 + 5 + 4 + 3 + 3 + 4 = 23$.

- Given n, efficiently construct a lopsided tree with n leaves that has minimal cost, where the cost of a tree is its external path length, i.e., the sum of the lengths of the paths from the root to all of the leaves.
- Calculate exactly how many nodes of depth $\leq x$ exist in the infinite tree.
- Express the cost of the minimal cost tree as a function of n and the c_i.

To motivate these problems we first introduce the concept of a Varn-code [29] [25]. Suppose that we wish to construct a prefix-free encoding of n symbols using an encoding alphabet of r letters, $\Sigma = \{\alpha_1, \ldots, \alpha_r\}$ in which the length of character α_i is c_i, where the c_is may all be different. As an example consider the Morse-code alphabet $\Sigma = \{., -\}$ in which the length of a "dash" may be longer than that of a "dot". By a prefix-free encoding we mean a set of n strings $\{\omega_1, \ldots, \omega_n\} \subseteq \Sigma^*$ in which no ω_i is a prefix of any ω_j.

If a symbol is encoded using string $\omega = \alpha_{i_1}\alpha_{i_2}\ldots\alpha_{i_l}$ then $cost(\omega) = \sum_{j \leq l} c_{i_j}$ is the length of the string. For example if $r = 2$, $\Sigma = \{0, 1\}$ and $c_1 = c_2 = 1$ then the cost of the string is just the number of bits it contains. This last case is the basic one encountered in regular Huffman encoding [26] [10].

Now suppose that the n symbols to be encoded are known to occur with equal frequency. The *cost* of the code is then defined to be $\sum_{i \leq n} cost(\omega_i)$ (which divided by n is the average cost of transmitting or *length* of a symbol). Given $c_1 \leq c_2 \leq \cdots \leq c_r$ a *Varn-code* for n symbols is a minimum-cost code. Varn codes have been extensively studied in the compression and coding literature; [25] contains a large bibliography and up-to-date description of what is currently known about them.

Such codes can be naturally modelled by lopsided trees in which the length of the edge from a node to its i^{th} child is c_i; we call such an edge an i^{th} *edge*. Suppose that v is a leaf in a lopsided tree and the unique path from the tree's root to v first traverses an i_1^{st} edge then an i_2^{nd} edge and so on up to an i_l^{th} edge. We can then associate with this leaf the codeword $\omega = \alpha_{i_1}\alpha_{i_2}\ldots\alpha_{i_l}$. The cost of this codeword is exactly the same as the depth of v in the tree, i.e., $\sum_{j\le l} c_{i_j}$. Using this correspondence, every tree with n leaves corresponds to a prefix-free set of n codewords and vice-versa; the cost of the code is exactly equal to the external path length of the tree which we will henceforth call the *cost* of the tree. This correspondence is extensively used, for example, in the analysis of Huffman codes. See Figure 1. A lopsided tree with minimal cost for n leaves will be called an *optimal tree.* See Figure 2.

With this correspondence and notation we see that the problems of constructing a Varn code and calculating its cost are equivalent to those of constructing an optimal tree and calculating its cost. Under these two different guises these problems have been extensively studied in both the coding/compression and computer science communities. Algorithms for finding such trees appear in [16] [5] [29] [9] [23] [15]. The first two citations are special cases; in [16] Kapoor and Reingold discuss the binary case ($r = 2$) and in [5] Choy and Wong address what they call α, β trees, trees in which r is fixed and $\forall i,\ c_i = \alpha + (i-1)\beta$. Both special case algorithms run in $O(n)$ time. The remaining citations provide algorithms for the general case; the fastest one is [15] which runs in $O(n \log^2 r)$ time. The bottleneck in these general algorithms is that there are many possible trees with n leaves and all of the algorithms work by constructing restricted (but large) sets of such trees and somehow finding the minimal-cost one in the set; this tree will be the optimal one. (When $r = 2$ this restricted set will collapse to only one candidate tree; it is when $r > 2$ that problems arise). Analyses of the costs of Varn codes or, equivalently, of the costs of optimal lopsided trees appear in [27] [22] [6] [7] [1] [16] [25]. As the authors of these papers mention, their various analyses are only tight for some special cases but in most cases provide only loose upper and lower bounds. Many of the analyses use information theoretic techniques and therefore can not be tight (since they do not fully model the tree). The most complete analysis is in [16] which derives a closed asymptotic expression but restricts itself to the binary-tree case for rational c_1/c_2, leaving both the binary irrational and the general r-ary cases open problems. Lopsided trees have also been used to model searching procedures with r-ary branching in which the cost of discovering that the ith branch must be taken is c_i [28].

This paper has two main results, one combinatorial, the other analytic. Combining them yields a a full analysis of the cost of Varn codes.

The first main result of this paper is a new combinatorial way of looking at optimal lopsided trees. We assume that $c_1 \le c_2 \le \cdots \le c_r$ are fixed and examine how the structure of optimal lopsided trees evolve as n increases. We prove that the trees evolve in a very regular and understandable fashion. This will permit us to know what an optimal tree for n nodes looks like without having to search through a large collection of lopsided trees.

An immediate consequence of this result is an algorithm for constructing optimal trees in $O(n \log r)$ time, beating the old $O(n \log^2 r)$ bound. A different analysis shows that this algorithm also runs in $n + O(Cr \log r \log^r n)$ time where C is a constant dependent upon the c_i; this second running time is therefore not directly comparable with the algorithms in the literature (although, for fixed $c_1, \ldots, c_r$, it always beats them for large enough n.).

The second main result of this paper is an analysis of exactly how many nodes in the infinite tree have depth at most x. This problem reduces to analyzing Fibonacci-type recurrences of the form

$$L(x) = L(x - c_1) + L(x - c_2) + \cdots L(x - c_r). \tag{1}$$

This problem is easy if the c_i are integers or rational multiples of each other. It

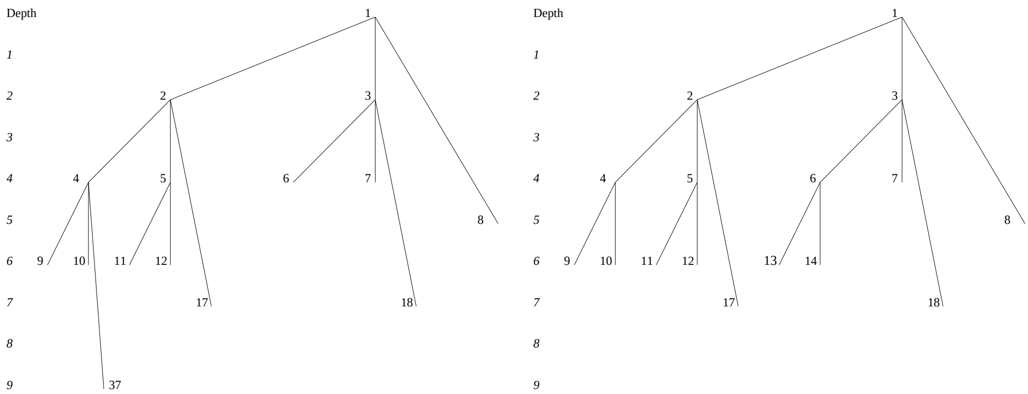


Figure 2: The two trees pictured both have $r = 3$, $(c_1, c_2, c_3) = (2, 2, 5)$ and 10 leaves each. The leftmost one has 5 internal nodes and cost 60; the rightmost one has 6 internal nodes and cost 59. We will see later that the rightmost one is optimal

gets complicated when the c_i are irrational. While the solution to the case $r = 2$ is implicit in the work of Fredman and Knuth [12] and, later, Pippenger [24], the general r-ary case does not seem to have been previously addressed. We show how to use Mellin transform and singularity analysis techniques to solve these equations. The solution to (1) will have a qualitatively different behavior depending upon whether the c_i are rational multiples of each other or not. This difference in behavior will be reflected in the ways in which the trees evolve as the number of leaves increase.

The analysis of $L(x)$ is of independent interest. We illustrate this by describing an application in the derivation of exact bounds on the time needed for broadcasting in the Postal-Model of computation, improving the bounds given in [2].

A more important application arises when we combine the analysis of $L(x)$ with the first result describing the structure of trees. This will yield the major result of this paper, an exact analysis of the cost of optimal trees or Varn-codes. Because we know exactly how the structure of the optimal tree evolves as n grows we are able to calculate how the cost of the optimal tree increases with n. This provides, once and for all, a unified analysis that gives asymptotically exact bounds for Varn-code costs in all cases.

The remainder of this paper is structured as follows. Section 2 motivates the problem and introduces the definitions that will be used. Section 3 presents (mostly without proof) the three major results of this paper: (i) a description of the combinatorial structure of optimal trees and the the algorithms that it implies; (ii) an asymptotic analysis of the solution to generalized Fibonacci recurrences and the subsequent analysis of the distribution of nodes in the infinite tree that it implies as well as the analysis of broadcasting protocols in the Postal model that follows; (iii) an asymptotic analysis of optimal Varn lopsided trees or, equivalently, Varn codes.

Sections 4, 5 and 6 will prove the respective results of parts (i), (ii) and (iii).

2 Definitions

In this section we motivate and introduce the definitions used in the rest of the paper. In what follows $\{x\} = x - \lfloor x \rfloor$ is the *fractional part* of x, e.g., $\{7.32\} = .32$.

Now let $0 < c_1 \le c_2 \le \cdots \le c_r$ be r fixed reals.

Definition 1
The **infinite r-ary tree** *is the infinite, rooted, r-ary tree such that the length of*

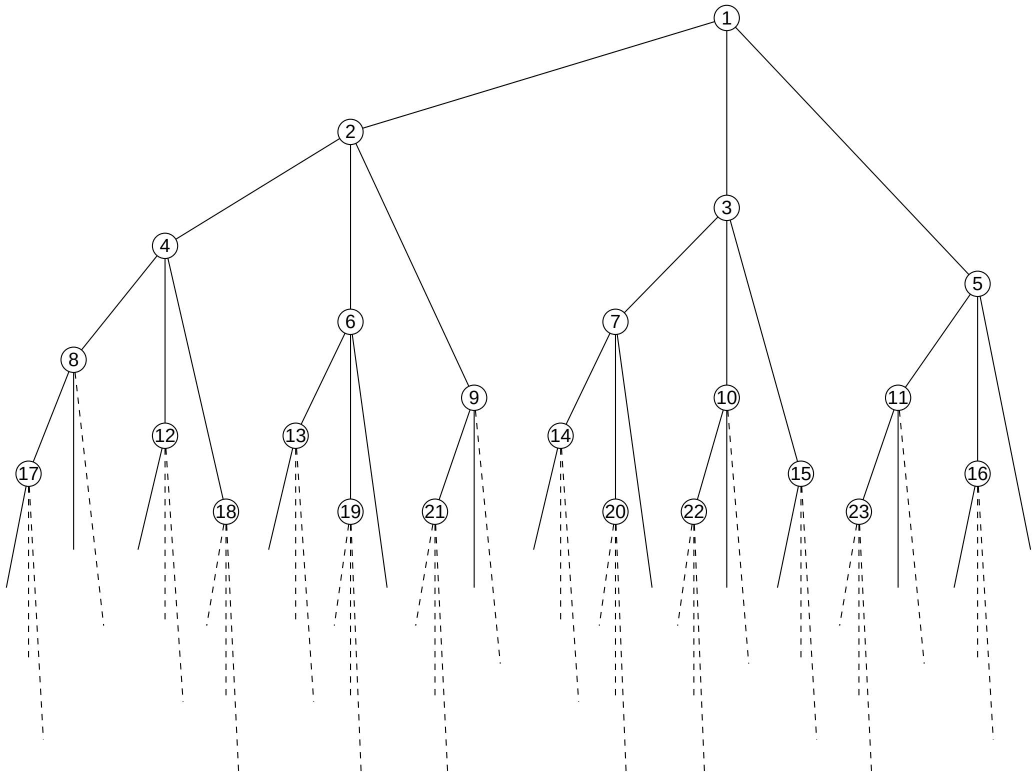


Figure 3: The labeled infinite 3-ary tree with $(c_1, c_2, c_3) = (3, 5, 7)$. Nodes are drawn so that the depth of nodes on the page corresponds to their depths in the tree. This convention will be followed throughout this paper.

the edge connecting a node to its i^{th} *child is* c_i*. See Figure 3 .*

A **lopsided tree** *is a subtree T of the infinite r-ary tree containing the root. See Figure 4.*

If u is a node in the infinite tree then $\mathbf{child_i(u)}$ *is the* i^{th} *child of u, e.g., in Figure 3* $child_1(4) = 8$, $child_2(4) = 12$ *and* $child_3(4) = 18$.

In standard trees the *depth* of a node v is defined to be the number of edges on the unique path from the root to v. The *external path length* of the tree is the sum of the depth of all external nodes. This definition can easily be extended to lopsided trees if we redefine the depth of v to be the sum of the lengths of the edges on the path connecting the root to v.

Definition 2 *Let u be a node and T be a lopsided tree.*
$\mathbf{depth}(u)$ *is the sum of the lengths of the edges on the path connecting the root to u.*
The **external path length** *or* **cost** *of T is* $C(T) = \sum_{v \text{ a leaf of } T} depth(v)$*, the sum of the depths of all external nodes.*
The **height** *of T is* $H(T) = \max_{u \in T} depth(u)$.

For example, in the tree T in Figure 4, $depth(4) = c_1 + c_1 = 3 + 3 = 6$, $depth(6) = c_1 + c_2 = 3 + 5 = 8$, and $C(T) = 6 + 8 + 10 + 8 + 10 + 7 = 49$. $H(T) = 10 = depth(9) = depth(10)$.

Definition 3 *A tree T with n leaves is* **optimal** *if it has the minimal cost among all lopsided trees with n leaves. We denote such an optimal tree by T_n (note that it might not be unique). See Figure 5.*

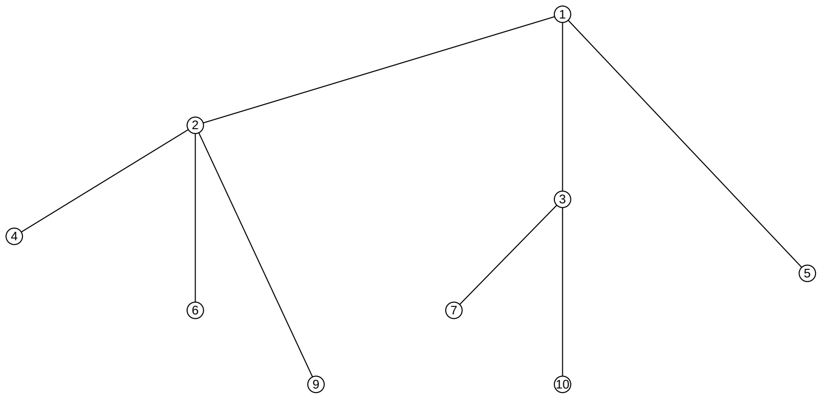


Figure 4: A lopsided tree with $r = 3$ and $(c_1, c_2, c_3) = (3, 5, 7)$.

In our analysis of the structure of the trees we will be interested in knowing at what depths nodes can appear in the infinite tree. It is obvious that nodes will appear at exactly the set of depths $\{\sum_{i=1}^{r} a_i c_i\}$ where the a_i range over all nonnegative integers. For example, if $(c_1, c_2) = (15, 25)$ then nodes can only have depths that are multiples of $5 = gcd(15, 25)$ and, deep enough in the tree, nodes will appear on *every* level with depth a multiple of 5. If though, $(c_1, c_2) = (3, \pi)$, then nodes appear on all depths that can be written in the form $a_1 c_1 + a_2 c_2$, $a_1, a_2 \geq 0$ integers, and general theorems about irrational numbers [21] imply that the depth difference between (neighboring) successive levels upon which nodes appear tends to zero. This will be discussed in more detail in Section 3.2. To formalize this distinction we introduce the following definitions:

Definition 4 *Let $(c_1, \ldots, c_r)$ be a tuple of r positive reals:*

1. *The tuple is* rationally related *if there exists $d > 0$ and positive integers $(c_1', \ldots, c_r')$ such that*
$$(c_1, \ldots, c_r) = d \cdot (c_1', \ldots, c_r') \qquad \text{and} \qquad gcd(c_1', \ldots, c_r') = 1.$$
2. *If $(c_1, \ldots, c_r)$ is rationally related we define the gcd of the tuple by $gcd(c_1, \ldots, c_r) = d$.*
3. *If $(c_1, \ldots, c_r)$ is not rationally related it is said to be* irrationally related.

For example, triple $(2\pi, 4\pi, 6\pi)$ is rationally related because $(2\pi, 4\pi, 6\pi) = 2\pi(1, 2, 3)$ but the triple $(1, 4, \pi)$ is not-rationally related. Note too that if the $(c_1, \ldots, c_r)$ are all integers then the gcd defined above is the standard *greatest common divisor,* e.g., $(10, 20, 35) = 5(2, 4, 7)$ so $\gcd(10, 20, 35) = 5$.

There is one more definition that we will need. It will turn out that optimal trees of a certain size will have a bottom "fringe" of size h where h is defined in the following lemma (which will be proved in Section 4):

Lemma 1 *Let $x_m = \frac{\sum_{i=1}^{m} c_i}{m-1}$ for $m = 2, \ldots, r$. There exists $k \geq 2$ such that*

$$x_2 \geq x_3 \geq \cdots \geq x_{k-1} \geq x_k < x_{k+1} < \cdots < x_r. \tag{2}$$

(If $x_2 \leq x_3$ set $k = 2$. If $x_2 \geq x_3 \geq \cdots \geq x_{r-1} \geq x_r$ set $k = r$.) Letting k be this value and setting $h \stackrel{\text{def}}{=} x_k$ we have, further, that, if $k < r$ then $c_k \leq h < c_{k+1}$.

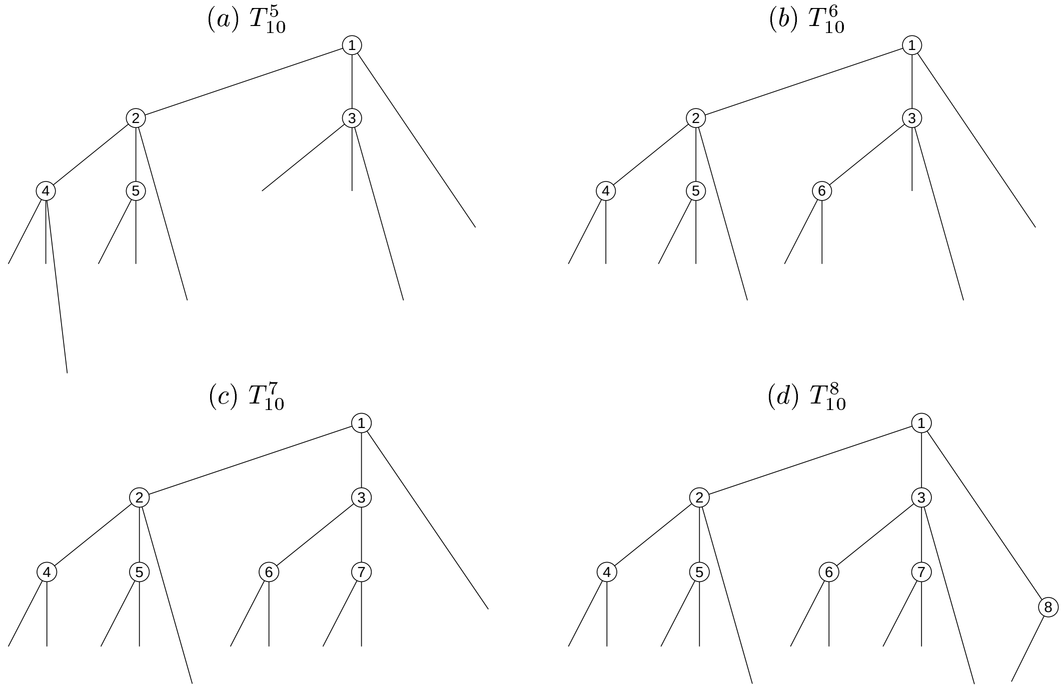


Figure 5: For $r = 3$, $(c_1, c_2, c_3) = (2, 2, 5)$. The cost of the trees are, respectively, 60, 59, 60, 62. It can be shown that (b) is an optimal tree for $n = 10$ leaves.

3 The Results

In this section we present the major results of this paper. Proofs of most of the results are deferred until later.

Before starting the discussion of *lopsided* trees we try to give some intuition by quickly reviewing what is know about the standard, *non-lopsided*, tree, i.e., $r - 2$ and $c_1 = c_2 = 1$. It is well known e.g., [19, 5.3.1], that such a tree with minimal external path length for n leaves must have the following property: all leaves must appear on level $l = \lfloor \log_2 n \rfloor$ or level $l + 1$. This means that there will be $2^{l+1} - n$ leaves on level l and $2n - 2^{l+1}$ leaves on level $l + 1$. The external path length is then:

$$l(2^{l+1} - n) + (l + 1)(2n - 2^{l+1}) = ln + 2n - 2^{l+1}.$$

The question now is how to rewrite this as a function of n. The standard way of doing so is to note that

$$l = \log_2 n - \{\log_2 n\}, \quad 2^{l+1} = n2^{1-\{\log_2 n\}}.$$

The external path length can therefore be rewritten as

$$\begin{aligned} ln + 2n - 2^{l+1} &= n \log_2 n - n \left[2 - \{\log_2 n\} - 2^{1-\{\log_2 n\}}\right] \\ &= n \log_2 n + nB(\log_2 n) \end{aligned}$$

where $B(\theta) = 2 - \{x\} - 2^{1-\{x\}}$ is periodic with period 1, i.e., $B(1 + x) = B(x)$. See Figure 6.

Two important observations to keep in mind about this example are: (i) the optimal tree was one that tried to keep the leaves as "balanced" as possible; (ii) the

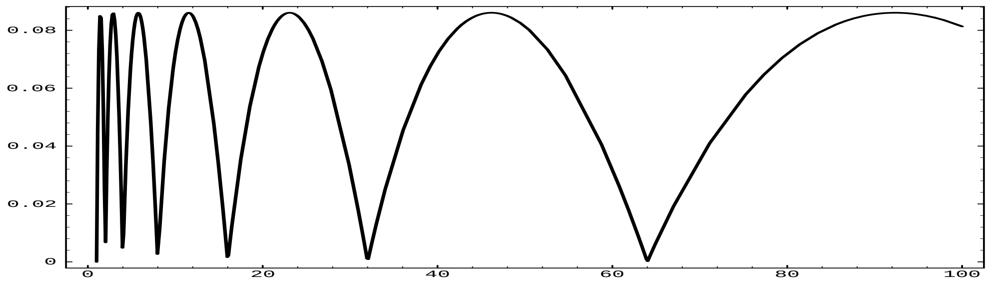


Figure 6: The function $B(\theta) = 2 - \{x\} - 2^{1-\{x\}}$.

fact that the leaves can occur on two distinct levels a unit distance apart introduced the periodic $B(\theta)$ term into the expression for path length. This periodic term "corrects" for the fact that there is a discrete jump between successive levels. Both of these observations will prove helpful in understanding the results in the remainder of this section.

3.1 The Structure of Optimal Trees

In this subsection we describe trees T_n that have minimal external path length among all trees with n leaves. More particularly we examine how the structure of the optimal trees change as n grows.

We start by labelling the nodes of the infinite tree as 1,2,3, $\ldots$, in order of increasing depth, breaking ties arbitrarily. That is, if u and v are two nodes with $depth(u) < depth(v)$, then $u < v$; if $depth(u) = depth(v)$, break ties arbitrarily. Figures 3 and 7 provide examples of such labellings.

Definition 5 *Let V be any set of nodes. Set*

$$LEAF(V) = \{u : parent(u) \in V, u \notin V\}.$$

For $n \leq |LEAF(V)|$, let

$$LEAF_n(V) = \textit{the } n \textit{ smallest labelled nodes in } LEAF(V).$$

$LEAF_n(V)$ is the set of n smallest labelled nodes that are children of nodes in V but are not in V themselves. $LEAF_n(V)$ contains n highest (smallest depth) in $LEAF(V)$.

For example, in Figure 7, $LEAF_5(\{1,2,3\}) = \{4,5,6,7,8\}$

It is obvious that an optimal tree must be *proper* (each of its internal nodes must have at least two children) otherwise some internal node can be replaced by its child, decreasing the cost. This property of being *proper* will be useful later.

For any given n, let m be the number of internal nodes of some proper tree having n leaves. The total number of edges in the tree is $n + m - 1$. Since every internal node has between 2 and r children, $2m \leq n + m - 1 \leq rm$, or $\lceil \frac{n-1}{r-1} \rceil \leq m \leq n - 1$.

Definition 6 *Let $\lceil \frac{n-1}{r-1} \rceil \leq m \leq n-1$, and $V_m = \{1, 2, \ldots, m\}$. Set $T_n^m = V_m \cup LEAF_n(V_m)$.*

By definition, if $u \in T_n^m$ is not a root, then $parent(u) \in T_n^m$ so T_n^m is a tree. These trees are called *shallow trees* in [15] which uses the simple observation that there must be some shallow tree that is also optimal as the basis of an algorithm for constructing optimal trees. Figure 5 illustrates the shallow trees T_{10}^5, T_{10}^6, T_{10}^7 and T_{10}^8 when $(c_1, c_2, c_3) = (2, 2, 5)$.

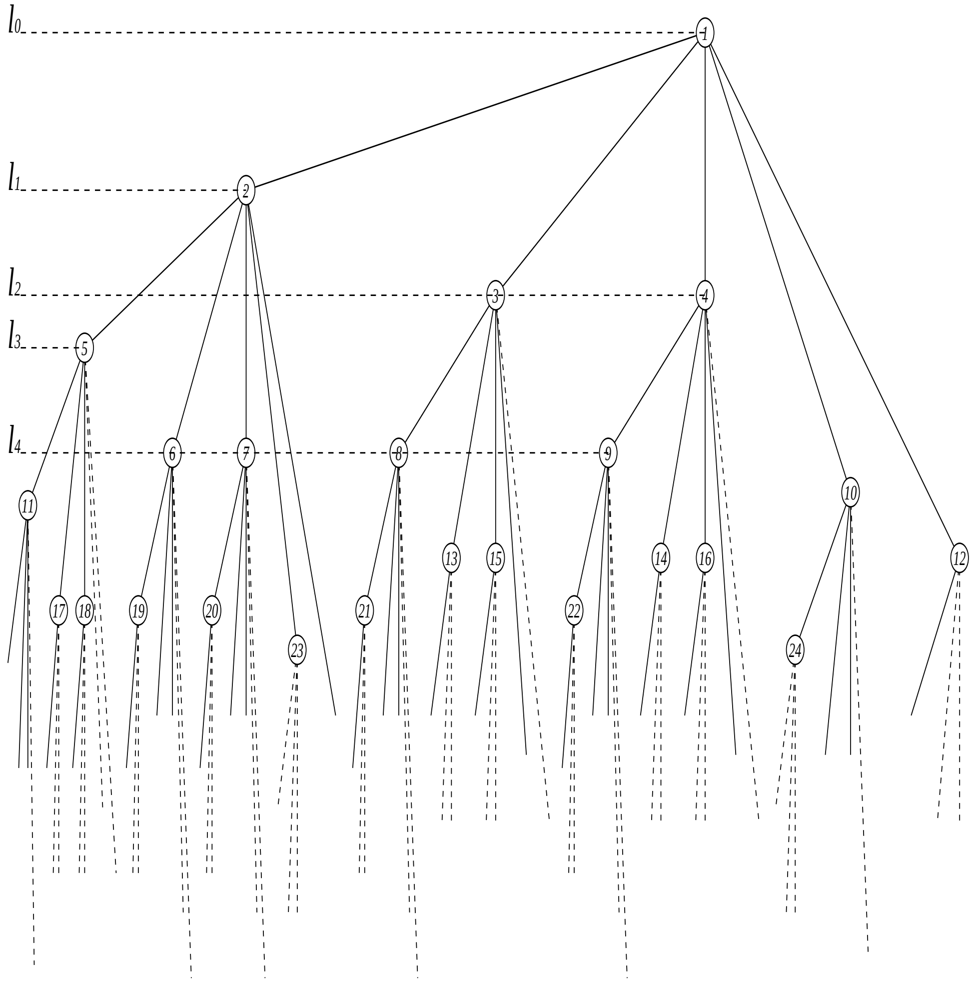


Figure 7: The top of the infinite tree with $(c_1, c_2, c_3, c_4, c_5) = (3, 5, 5, 8.75, 10)$. $(r = 5)$. Note that $l_0, l_1, l_2, l_3, l_4, \ldots = 0, 3, 5, 6, 8\ldots$ and $m_0, m_1, m_2, m_3, m_4 = 1, 2, 4, 5, 9$.

Return now to the infinite tree and let $l_0, l_1, l_2, \ldots,$ be the consecutive levels upon which nodes appear, i.e., $l_0 = 0$ and

$$\forall i > 0,\, l_i = \min\{\mathit{depth}(v) \,:\, v \text{ a node with } \mathit{depth}(v) > l_{i-1}\}.$$

Thus, for example, $l_1 = c_1$, $l_2 = \min(2c_1, c_2)$, etc..

Also let m_j be the number of nodes higher than or on depth l_j (Figure 7):

$$m_j = |\{v \text{ a node } : \mathit{depth}(v) \le l_j\}.$$

be the number of nodes higher than or on depth l_j. See Figure 7.

With these definitions, we can now state our main combinatorial result. Figures 8, 9, 10 and 11 illustrate the various cases of the theorem for $r = 5$ and $(c_1, c_2, c_3, c_4, c_5) = (3, 5, 5, 8.75, 10)$. The proof of the theorem is given in Section 4.

Theorem 1 *Given $c_1 \le c_2 \ldots \le c_r$ let k and h be as defined in Lemma 1 and l_j and m_j be as defined above. Set*

$$A_j = \{v \in \mathit{LEAF}(V_{m_j}) : \mathit{depth}(v) \le l_j + h\}$$

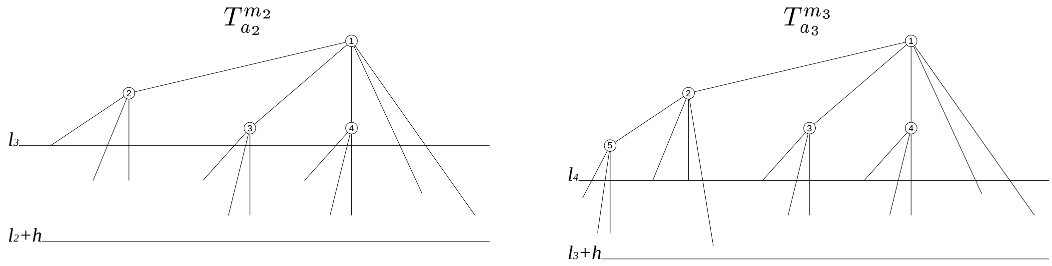


Figure 8: In the first case of Theorem 1 when $n = a_j$, the optimal tree $T_n = T^{m_j}_{a_j}$. In this example, $a_2 = 11$, $a_3 = 14$. Therefore $T_{11} = T^{m_2}_{a_2}$ and $T_{14} = T^{m_3}_{a_3}$.

and $a_j = |A_j|$ *the number of nodes in* A_j*, set*

$$B_j = A_j \cup \{v \in LEAF(V_{m_j}) : l_j + h < depth(v) \leq l_{j+1} + h\}$$

and $b_j = |B_j|$ *the number of nodes in* B_j*.*

1. *If* $n = a_j$*, then the tree* $T^{m_j}_{a_j} = V_{m_j} \cup A_j$ *is optimal;*
2. *If* $a_j < n \leq b_j$*, then the tree* $T^{m_j}_n$ *is optimal; furthermore,* $T^{m_j}_{b_j} = V_{m_j} \cup B_j$*.*
3. *If* $b_j < n \leq a_{j+1}$*, then*
   (a) $n = b_j + p(k-1)$*, then* $T^{m_j+p}_n$ *is optimal;*
   (b) $n = b_j + p(k-1) + q$ *with* $q < k-1$*, then one of* $T^{m_j+p}_n$ *and* $T^{m_j+p+1}_n$ *is optimal.*

Intuitively this theorem reverses the problem. Instead of asking "how many internal nodes are in an optimal tree with n leaves" it instead asks, and answers, the question "if T is an optimal tree with m internal nodes, how many leaves n_m can T have?".

We now briefly sketch the theorem's implications. Figures 8, 9, 10 and 11 illustrate the discussion. To build tree $T^{m_j}_{a_j}$ draw a horizontal line across the infinite tree at depth l_j; the nodes on or above this line are the m_j nodes in V_{m_j}, the internal nodes in $T^{m_j}_{a_j}$. Next draw a second horizontal line at depth $l_j + h$. The nodes in V_{m_j} have exactly a_j children on or above this line. These nodes, the A_j, will be the leaves of $T^{m_j}_{a_j}$. The tree $T^{m_j}_{a_j}$ will be optimal.

Now draw a third horizontal line at depth $l_{j+1} + h$. There are $b_j - a_j$ children of the V_{m_j} between the second and third lines (actually, below the second and on or above the third). If n satisfies $a_j < n \leq b_j$ then construct $T^{m_j}_n$ by taking the $n - a_j$ highest nodes between the second and third lines and adding them to $T^{m_j}_{a_j}$ as leaves. The tree $T^{m_j}_n$ will be optimal. The largest such tree constructed this way is $T^{m_j}_{b_j}$.

If $b_j < n \leq a_{j+1}$ the theorem says that optimal T^m_n can be constructed as follows. If n is of the form $n = b_j + p(k-1)$ take the p highest leaves in $T^{m_j}_{b_j}$, make them internal and add their k highest children to the tree.

If $b_j < n < a_{j+1}$ but $n = b_j + p(k-1) + q$ with $0 < qk - 1$ then we do not know a-priori what the optimal tree must be. By looking at the definitions we do know, though, that it must be *either* the tree that results from starting with $T^{m_j+p}_{b_j+p(k-1)}$ and adding the q smallest unused leaves in V_{m_j+p} to the tree or starting with $T^{m_j+p+1}_{b_j+(p+1)(k-1)}$ and erasing the $k-1-q$ deepest leaves in that tree.

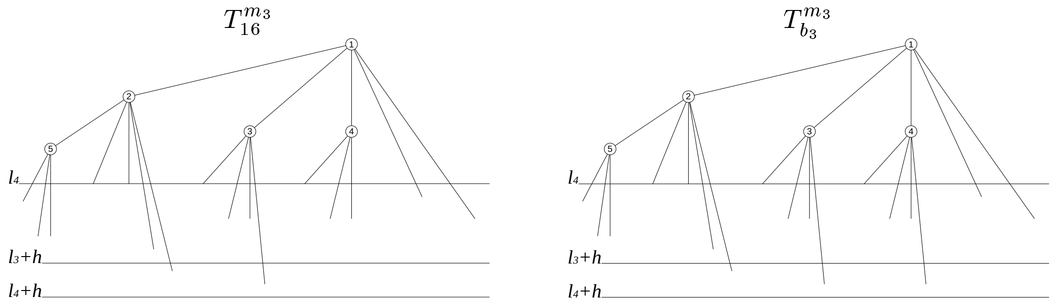


Figure 9: In the second case of Theorem 1 when $a_j < n \leq b_j$, the optimal tree $T_n = T_n^{m_j}$. In this example, $a_3 = 14$, $b_3 = 17$. So $T_{15} = T_{15}^{m_3}$ (not pictured), $T_{16} = T_{16}^{m_3}$ and $T_{17} = T_{17}^{m_3}$.

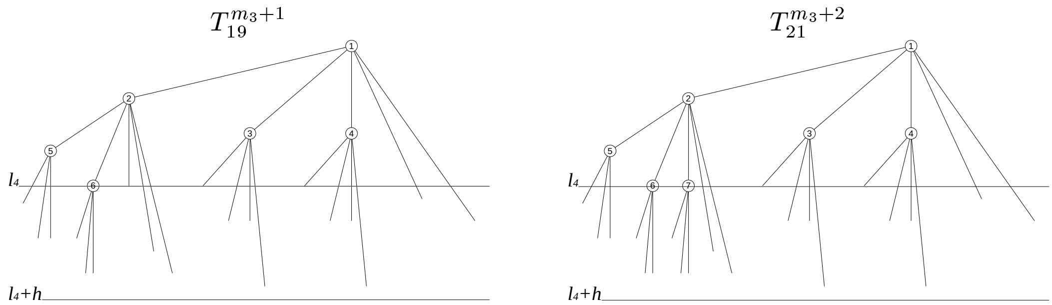


Figure 10: In the third case of Theorem 1 $b_j < n < a_{j+1}$. If $n = b_j + p(k-1)$, the optimal tree for n leaves is $T_n = T_n^{m_j+p}$. In this example, $b_3 = 17$, $19 = 17+1\cdot(3-1)$, and $21 = 17 + 2 \cdot (3-1)$. So $T_{19} = T_{19}^{m_3+1}$ and $T_{21} = T_{21}^{m_3+2}$.

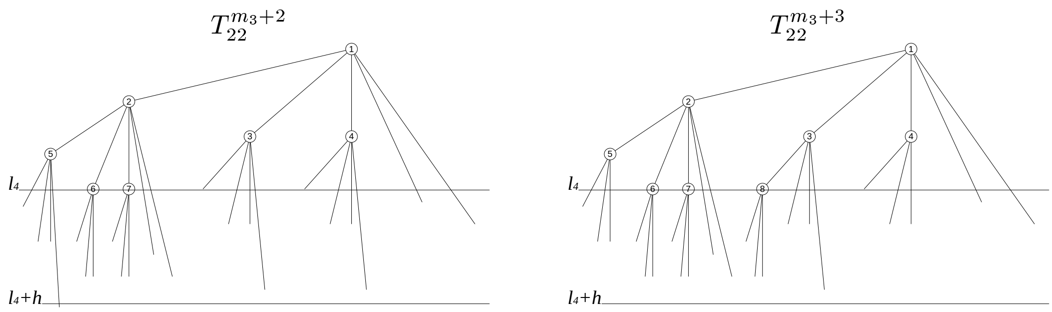


Figure 11: In the second part of the third case of Theorem 1 $b_j < n < a_{j+1}$ and $n = b_j + p(k-1) + q$ with $0 < q < k-1$. the optimal tree for n leaves is either $T_n^{m_j+p}$ or $T_n^{m_j+p+1}$. In this example, $b_3 = 17$, $22 = 17 + 2 \cdot (3-1) + 1$. The cost of $T_{22}^{m_3+2}$ is 246.75 while the cost of $T_n^{m_3+3}$ is 247.25. So $T_{22} = T_{22}^{m_3+2}$.

3.2 The Algorithm

In this section we use the results of the previous section to develop an algorithm that, given n, constructs an optimal tree with n leaves. This algorithm will run in $O(n \log r)$ time, improving upon the best previously known ones which ran in $O(nr^2)$ [23] and $O(n\log^2 r)$ [15] time respectively. With a more delicate analysis, this algorithm can be shown to run in $n + O(r \log r \log^r n)$ time. In this second analysis, the $O()$ term hides constants dependent upon the values $(c_1, \ldots, c_r)$ and so cannot be directly comparable to the running times of the previously known algorithms which were independent of $(c_1, \ldots, c_r)$.

There are two parts to our algorithm. The first part constructs the tree

$$T_{part_I} = \begin{cases} T_n^{m_j}, & \text{if } a_j \le n \le b_j; \\ T_{b_j+p(k-1)}^{m_j+p}, & \text{if } b_j < n < a_{j+1}, \text{ where } p = \left\lfloor \frac{n-b_j}{k-1} \right\rfloor. \end{cases}$$

where a_j and b_j are as defined in Theorem 1. According to that theorem, if $a_j \le n \le b_j$, then $T = T_n^{m_j}$ is an optimal tree for n leaves and the algorithm concludes, having found the optimal tree. Otherwise $b_j < n < a_{j+1}$, and the optimal tree for n leaves is the minimum of $\{T_n^{m_j+p}, T_n^{m_j+p+1}\}$, where $p = \lfloor \frac{n-b_j}{k-1} \rfloor$. The second part of our algorithm thus constructs both $T_n^{m_j+p}$ and $T_n^{m_j+p+1}$ from the tree $T = T_{b_j+p(k-1)}^{m_j+p}$ which had been constructed in the first part and returns the one with minimal cost.

The algorithm and its analyses will need the following definitions:

Definition 7 *Let T_n^m be as defined in Definition 6 and proper. Then*

1. $IN(T) \stackrel{\text{def}}{=}$ *the set of internal nodes of T;*

2. $EX(T) \stackrel{\text{def}}{=}$ *the set of leaves of T;*

3. $UN(T) \stackrel{\text{def}}{=}$ *the set of children of internal nodes of T that are not themselves in T.*

A special case that we will use very often is $T = T_n^m$; for this we have $IN(T_n^m) = V_m$, $EX(T_n^m) = LEAF_n(V_m)$, and $UN(T_n^m) = LEAF(V_m) \setminus LEAF_n(V_m)$.

We will also need two tree construction primitives:

$Branch(T, k)$: Take the minimum node in $EX(T)$ and make it an internal node with k children;

$Add(T)$: Add the minimum node in $UN(T)$ to T.

Clearly, each $Branch(T, k)$ adds $(k-1)$ leaves to T, while each $Add(T)$ adds 1 leaf to T. See Figure 12 for examples.

3.2.1 Part I

We start by deriving the relationship between $T_{a_t}^{m_t}$ and $T_{b_t}^{m_t}$: From Theorem 1

$$T_{b_{t-1}}^{m_{t-1}} = V_{m_{t-1}} \cup \{v \in LEAF(V_{m_{t-1}}) : depth(v) \le l_t + h\}$$

and

$$\begin{aligned} T_{a_t}^{m_t} &= V_{m_t} \cup \{v \in LEAF(V_{m_t}) : depth(v) \le l_t + h\} \\ &= T_{b_{t-1}}^{m_{t-1}} \cup \{v \in LEAF(V_{m_t} \setminus V_{m_{t-1}}) : depth(v) \le l_t + h\} \\ &= T_{b_{t-1}}^{m_{t-1}} \cup \{child_i(m) : i = 1, \ldots, k, m \in V_{m_t} \setminus V_{m_{t-1}}\} \\ &= T_{b_{t-1}}^{m_{t-1}} \cup \{child_i(m) : i = 1, \ldots, k, depth(m) = l_t\} \\ &\quad \text{(since } c_k \le h < c_{k+1}). \end{aligned}$$

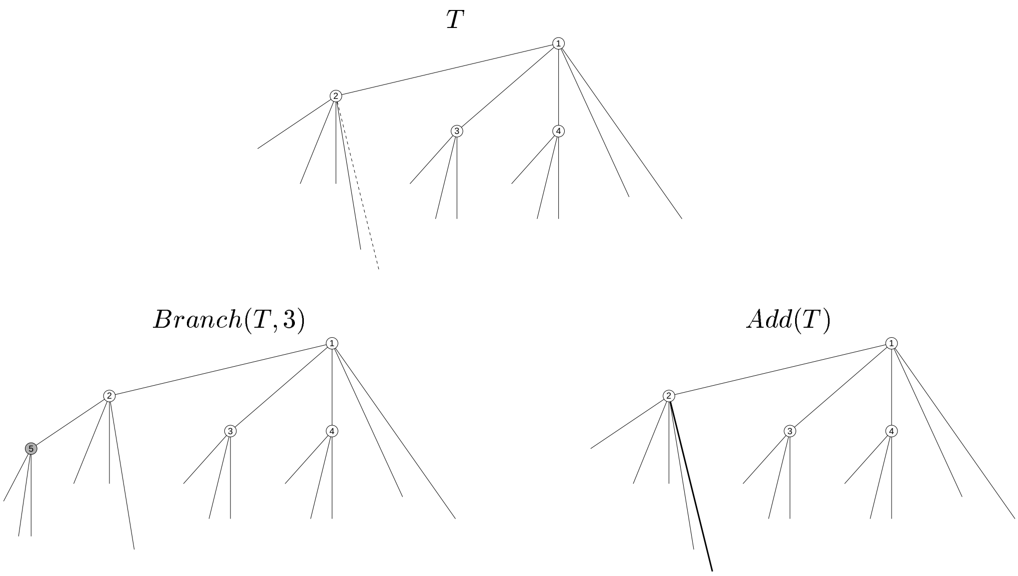


Figure 12: Examples of $Branch(T, k)$ and $Add(T)$.

Tree $T_{a_t}^{m_t}$ is therefore obtained from $T_{b_{t-1}}^{m_{t-1}}$ by making all leaves on level t (i.e. of depth l_t) into internal nodes with k children. This can be done by starting with $T = T_{b_{t-1}}^{m_{t-1}}$ and repeating $Branch(T, k)$ until the depth of the minimum node in $EX(T)$ is larger than l_t (i.e. equals l_{t+1}) or the remaining number of leaves n is less than $(k-1)$ We call this procedure *BRANCHING*. See Figure 13 for examples.

To find $T_{b_t}^{m_t}$ from $T_{a_t}^{m_t}$ recall that

$$\begin{aligned}
T_{b_t}^{m_t} &= V_{m_t} \cup B_t \\
&= V_{m_t} \cup A_t \cup \{v \in LEAF(V_{m_t}) : l_t + h < depth(v) \leq l_{t+1} + h\} \\
&= T_{a_t}^{m_t} \cup \{v \in LEAF(V_{m_t}) : l_t + h < depth(v) \leq l_{t+1} + h\} \\
&= T_{a_t}^{m_t} \cup \{u \in UN(T_{a_t}^{m_t}) : depth(u) \leq l_{t+1} + h\}
\end{aligned}$$

Tree $T_{b_t}^{m_t}$ is therefore obtained by starting with $T_{a_t}^{m_t}$ and adding all of the children of internal nodes of $T_{a_t}^{m_t}$, i.e., nodes in $UN(T_{a_t}^{m_t})$, whose depths are less than or equal to $l_{t+1} + h$. This can be done by starting with $T = T_{a_t}^{m_t}$ and repeating $Add(T)$ until the depth of the minimum node in $UN(T)$ is larger than $l_{t+1} + h$ or the remaining number of leaves n is zero. We call this procedure *ADDING*. See Figure 14 for examples.

Tree T_{part_I} is obtained simply by iteratively *BRANCHING* and *ADDING* as shown in Table 1..

3.2.2 Part II

The second part of our algorithm starts with the output $T_{b_j+p(k-1)}^{m_j+p}$ of Part I, constructs $T_n^{m_j+p}$ (if it exists) and $T_n^{m_j+p+1}$ and returns the minimum of the two.

In the example illustrated by the figures, $b_3 = 17$ and $a_4 = 25$, so for $n = 22$, the first part of our algorithm constructs T_{21}^7, the second part constructs T_{22}^7 and T_{22}^8 and returns the minimal one T_{22}^7. See Figures 10 and 11.

Let $T = T_{b_j+p(k-1)}^{m_j+p}$ and $m = m_j + p + 1$. Our algorithm adds the minimum nodes from $UN(T)$ until $n = 0$ or $UN(T)$ is empty. That is, it computes $T = T_{b_j+p(k-1)+q'}^{m_j+p}$ for some $q' \leq q$, with $q' = q$, if $T_n^{m_j+p}$ exists.

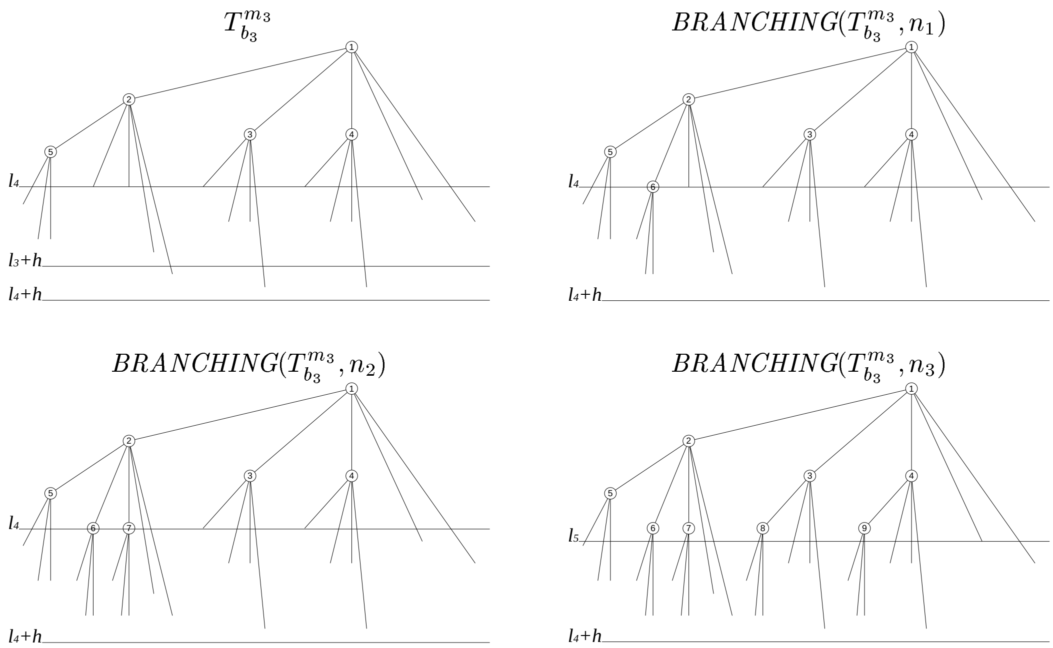


Figure 13: $r = 5$, $(c_1, c_2, c_3, c_4, c_5) = (3, 5, 5, 8.75, 10)$, $k = 3$ and $h = 6.5$. In this example, $b_3 = 17$ and $a_4 = 17 + 4 * (3 - 1)$. These are the trees returned by *BRANCHING* for $2 \leq n_1 < 4$, $4 \leq n_2 < 6$ and $n_3 \geq 8$.

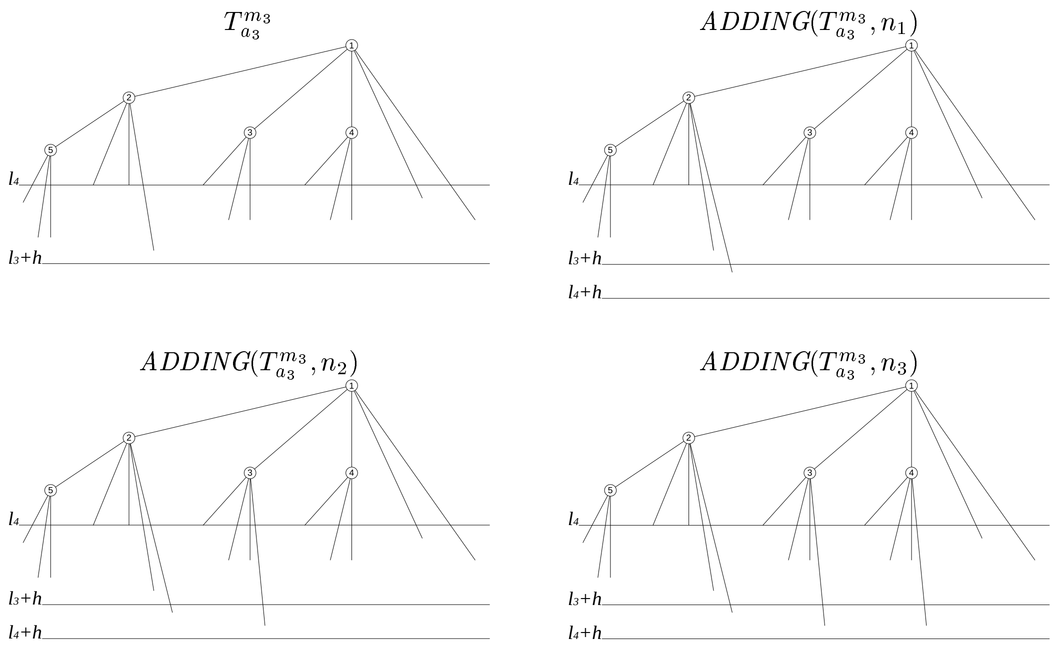


Figure 14: For $r = 5$, $(c_1, c_2, c_3, c_4, c_5) = (3, 5, 5, 8.75, 10)$, $k = 3$ and $h = 6.5$. In this example, $a_3 = 14$ and $b_3 = 17$. These are the trees returned by *ADDING* for $n_1 = 1$, $n_2 = 2$, and $n_3 \geq 3$.

Procedure *BRANCHING(T,n)*
Input : $T = T^{m_t}_{b_t}$
Output : $T^{m_{t-1}+p}_{b_{t-1}+p\cdot(k-1)}$, where $p = \min\{\lfloor \frac{n}{k-1} \rfloor, M_t\}$.
repeat
$Branch(T, k)$; $n = n - (k-1)$;
$u =$ *the minimum node in* $EX(T)$;
until $depth(u) > l_t$ or $n < (k-1)$

Procedure *ADDING*(T, n)
Input : $T = T^{m_t}_{a_t}$
Output : $T^{m_t}_{a_t+q}$, where $q = \min\{n, b_t - a_t\}$.
while $depth(u) \leq l_{t+1} + h$ and $n > 0$
$Add(T)$; $n = n - 1$;
$u =$ *be the minimum node in* $UN(T)$;
endwhile.

Algorithm : Construct T_{part_I}
Input : $c_1, \ldots, c_r$, n
Output : optimal tree for n leaves
compute k, h;
create T containing the single, root, node.
$n = n - 1$; $l_0 = 0$;
—*Part I*—
while $n \geq (k-1)$ do
$T^{m_{t-1}+p}_{b_{t-1}+p\cdot(k-1)} \leftarrow$*BRANCHING*$(T^{m_{t-1}}_{b_{t-1}}, n)$;
$l_{t+1} =$ the depth of minimum leaf in $T^{m_{t-1}+p}_{b_{t-1}+p\cdot(k-1)}$;
—*Test if* $T^{m_{t-1}+p}_{b_{t-1}+p\cdot(k-1)} = T^{m_t}_{a_t}$ —
if $l_{t+1} > l_t$ then $T^{m_t}_{a_t+q} \leftarrow$*ADDING*$(T^{m_t}_{a_t}, n)$;
endwhile;

Table 1: The upper level descriptions of the procedures ADDING, BRANCHING and the first part of the algorithm

We then compute $T^{m_j+p+1}_n$ from T, by first performing $Branch(T, n+1)$, labeling the new internal node m, and then replacing c maximum external nodes in T by adding c more children to m where c has the maximum value such that the c children just added are all higher than those external nodes removed.

The second part of the algorithm is given in Table 2

Implementation Details Until now we have only given a high level description of how to construct the optimal trees. We now discuss how to implement the algorithm efficiently.

Recall that in Part I, procedures *BRANCHING* (*ADDING* resp.) requires identifying the minimum node in $EX(T)$ ($UN(T)$ resp.); in Part II, we needed to identify the maximum node in $EX(T)$. To implement these operations efficiently, we use the following representation of nodes as given in [15]:

We label the nodes of the infinite tree in order of increasing depth as stated earlier but now, further refine the ordering so that ties are broken as follows: if $depth(u) = depth(v)$, $u = child_i(u')$ and $v = child_j(v')$, then

Part II

```
    —construct T_n^{m_j+p} and T_n^{m_j+p+1} —
if (n > 0) then
        while n > 0 and LowQ2 ≠ ∅ do
                child_i(v_i) = the minimum node in LowQ2;
                —Add child_i(v_i) to T —
                Add(T); n = n − 1;
                UPDATE(LowQ2,i);
        endwhile;
        —if n = 0 then T_n^{m_j+p} = T —
        for i = 1 to k do w_i = m;
        for i = k + 1 to r do w_i = v_i − 1;
        Create HighQ;
        —construct T_n^{m_j+p+1} —
        child_i(u_i) = the minimum node in LowQ1;
        Branch(T, n + 1); m = m + 1; depth(m) = depth(u_i) + c_i;
        m_deg = n + 2;
        child_i(w_i) = the maximum node in HighQ.
        while (depth(m) + c_{m_deg}) < (depth(w_i) + c_i) do
                add child_{m_deg}(m); m_deg = m_deg + 1;
                remove child_i(w_i);
                UPDATE(HighQ,i);
                child_i(w_i) = the maximum node in HighQ.
        endwhile;
endif.
```

Table 2: Part II of the algorithm

1. if $depth(u') < depth(v')$ (i.e. $u' < v'$), then $u < v$;
2. else $depth(u') = depth(v')$;
   (a) if $i < j$, then $u < v$;
   (b) if $i = j$, then $u < v$ if $u' < v'$.

This rule imposes a unique ordering of the nodes of the infinite tree. Figure 7 gives an example of such a labeling.

For each i, let

$$u_i = min\{v : child_i(v) \in EX(T)\},$$

$$v_i = min\{v : child_i(v) \in UN(T)\},$$

$$w_i = max\{v : child_i(v) \in EX(T)\}.$$

Then the minimum node in $EX(T)$ is the minimum among $\{child_i(u_i) : i = 1, \ldots, r\}$, which is maintained by a priority queue, called $LowQ1$. Similarly, we maintain priority queue $LowQ2$ for the minimum node in $UN(T)$ and a priority queue $HighQ$ for the maximum node in $EX(T)$.

Furthermore, with the tie-breaking rule, we have the following:

Lemma 2 *For each i, $\{v : child_i(v) \in EX(T)\} = \{u_i, u_i + 1, \ldots, w_i\}$.*

Proof. For any $u_i < u < w_i$, by the tie-breaking rule, we have $child_i(u_i) < child_i(u) < child_i(w_i)$, and therefore, by the definition of T, we have $child_i(u) \in EX(T_n^m)$. □

Procedure *BRANCHING*(T, n)

Input : $T_{b_{t-1}}^{m_{t-1}}$

Output : $T_{b_{t-1}+p(k-1)}^{m_{t-1}+p}$, where $p = min\{\lfloor \frac{n}{k-1} \rfloor, M_t\}$

$child_i(u_i) =$ *the minimum node in* $LowQ1$;
repeat
  repeat
    —*Make* $child_i(u_i)$ *an internal node with* k *children* —
    $Branch(T, k)$; $m = m + 1$;
    —*Label* $child_i(u_i)$ *as node* m *and assign its depth* —
    $depth(m) = l_t$; $n = n - (k - 1)$;
    $u_i = u_i + 1$;
  until $depth(u_i) > depth(u_i - 1)$ or $n < (k - 1)$;
  UPDATE($LowQ1$,i);
  $child_i(u_i) =$ *the minimum node in* $LowQ1$;
until $depth(u_i) + c_i > l_t$ or $n < (k - 1)$.

Procedure *ADDING*(T, n)

Input : $T_{a_t}^{m_t}$

Output : $T_{a_t+min\{n, b_t - a_t\}}^{m_t}$

$child_i(v_i) =$ *the minimum node in* $LowQ2$;
while $depth(v_i) + c_i \leq l_{t+1} + h$ and $n > 0$ do
  repeat
    —*Add* $child_i(v_i)$ *to the tree* —
    $ADD(T)$; $n = n - 1$;
    $v_i = v_i + 1$;
  until $depth(v_i) > depth(v_i - 1)$ or $n = 0$;
  UPDATE($LowQ2$,i);
  $child_i(v_i) =$ *the minimum node in* $LowQ2$;
endwhile.

Table 3: Priority queue implementations of the *BRANCHING* and *ADDING* procedures.

In particular, we have

$$\{v : \ child_i(v) \in EX(T), depth(u_i) = depth(v)\} = \{u_i, u_i + 1, \ldots, u_i + s\}$$

for some s. Suppose that $child_i(u_i)$ is a leaf in $EX(T_{b_t}^{m_t})$. Then the ith children of nodes in $\{u_i, u_i + 1, \ldots, u_i + s\}$ are also in $EX(T_{b_t}^{m_t})$. Consequentially, in *BRANCHING*, we do not need to update the priority queue $LowQ1$ immediately after each *Branch* operation. Instead, we only need to update the priority queue after performing all *Branch*es on "one level": after *Branch* at $child_i(u_i)$, if $depth(u_i + 1) = depth(u_i)$, let $u_i = u_i + 1$ and *Branch* at $child_i(u_i)$, repeat until $depth(u_i + 1) > depth(u_i)$ and only then update the priority queue. Alternatively, this can be visualized as branching nodes "level by level" by walking through nodes on $depth(u_i)$ and *branching* at their ith children. See Figure 15 for an example and Table 3 for psuedocode.

Similarly, we have

$$\{v : \ child_i(v) \in EX(T), depth(v) = depth(v_i)\} = \{v_i, v_i + 1, \ldots, v_i + s\}$$

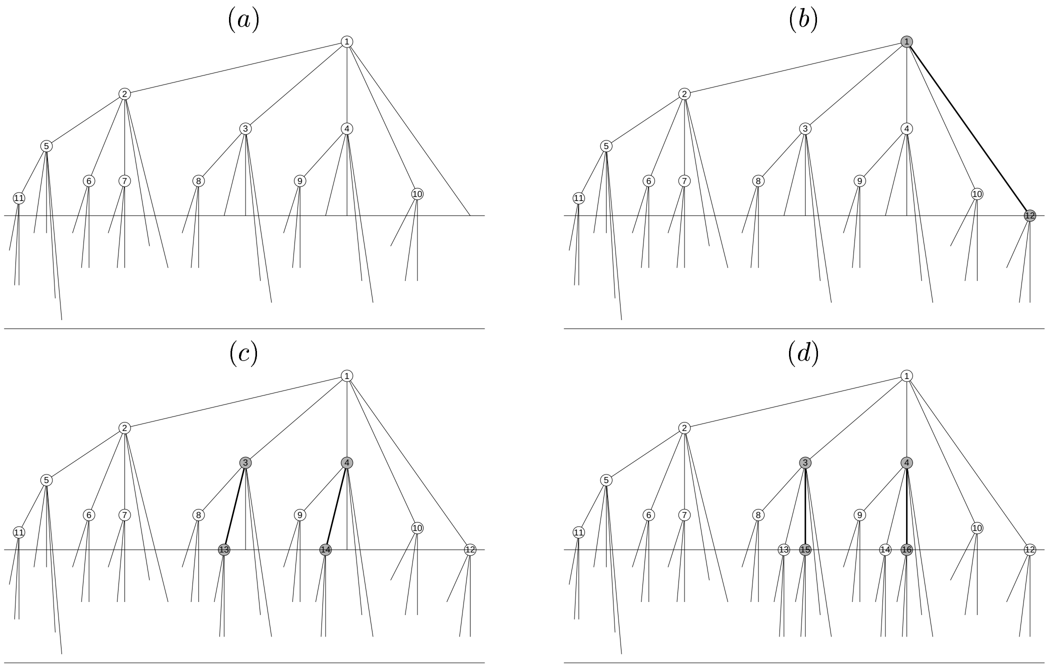


Figure 15: An illustration of *BRANCHING* "level by level".

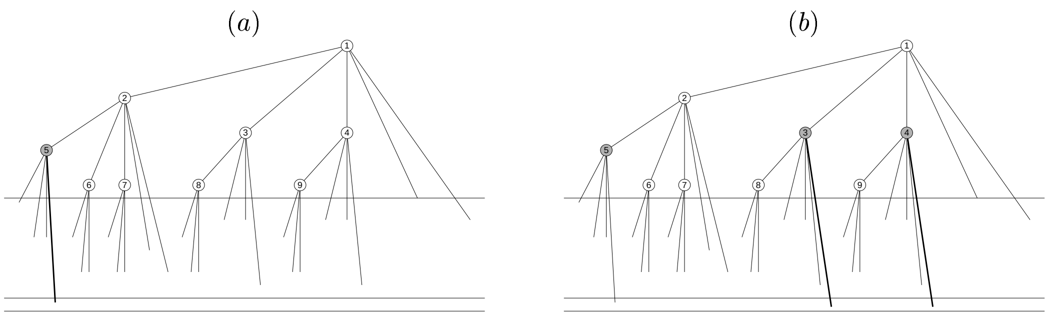


Figure 16: An illustration of *ADDING* "level by level".

for some s. In $T_{a_t}^{m_t}$, since every internal node has at least k children,

$$\{v \in V_m : child_i(v) \in UN(T_n^m)\} = \emptyset,$$

for $i = 1, \ldots, k$. Thus $LowQ2$ contains $\{child_i(v_i) : i = k+1, \ldots, r\}$. If $child_i(v_i)$ is a leaf in $UN(T_{a_t}^{m_t})$ whose depth is less than $l_{t+1} + h$ then the ith child of nodes in $\{v_i, v_i + 1, \ldots, v_i + s\}$ are also in $UN(T_{b_t}^{m_t})$. Therefore, in *ADDING*, we do not need to update the priority queue $LowQ2$ immediately after each *Add* operation. Instead, we only need to update the priority queue after performing all *Add*s on "one level": after performing *Add* $child_i(v_i)$, if $depth(v_i + 1) = depth(v_i)$, let $v_i = v_i + 1$ and *Add* $child_i(v_i)$, repeat until $depth(v_i + 1) > depth(v_i)$ and only then update the priority queue. Alternatively, this can be visualized as adding nodes "level by level" by walking through nodes on $depth(v_i)$ and adding their ith child. See Figure 16 for an example and Table 3 for psuedocode.

3.2.3 Time Complexity

Note that a priority queue containing r items can be created in $O(r)$ time and updated in $O(\log r)$ time [26]. The time required for $BRANCHING(T_{b_{t-1}}^{m_{t-1}}, n)$ is thus $p_t \cdot (k-1) + \alpha_t O(\log r)$ where $p_t = \min\{\lfloor \frac{n}{k-1} \rfloor$, number of nodes at level $t\}$ and α_t is the number of times we need to update the priority queue, $LowQ1$; the time required for $ADDING(T_{a_t}^{m_t}, n)$ is $q_t + \beta_t O(\log r)$ where $q_t = \min\{n, b_t - a_t\}$ and β_t is the number of times we need to update the priority queue, $LowQ2$. For $a_j \le n < a_{j+1}$, the time complexity of the first part is

$$\sum_{t=1}^{j} \left(p_t \cdot (k-1) + \alpha_t O(\log r) + q_t + \beta_t O(\log r)\right).$$

Since $\alpha_t \le p_t \cdot (k-1)$ and $\beta_t \le q_t$, the time complexity of first part is $O(\sum_{t=1}^{j}(p_t \cdot (k-1) + q_t) \cdot \log r) = O(n \log r)$.

In the second part, constructing $T_n^{m_j+p}$ takes $O(r \log r)$ time; creating the priority queue $HighQ$ takes $O(r)$ time; constructing $T_n^{m_j+p+1}$ also takes $O(r \log r)$ time.

The total time complexity of the algorithm is therefore $O(n \log r)$.

For a more delicate analysis, note that $\alpha_t \le r$ and $\sum_{t=1}^{j} \beta_t \le j(r-k)$, the time complexity for the first part of the algorithm is $n + j \cdot O(r \log r)$. To bound the running time, we therefore need to bound j.

It can be shown that $a_j = c\varphi^{l_j} + o(\varphi^{l_j})$ where φ and c depend upon the values $(c_1, \ldots, c_r)$ (this will be proven later in Lemma 13 and Lemma 16). Since $a_j < n$, this immediately implies that $l_j = O(\log_\varphi n)$. Now, nodes can appear on depth l_j if and only if there exists non-negative integers γ_i such that $\sum_{i=1}^{r} \gamma_i c_i = l_j$. This implies that $\gamma_i \le l_j/c_i \le l_j/c_1$, for all $i = 1, \ldots, r$. Therefore, $j \le |\{(\gamma_1, \ldots, \gamma_r) : \gamma_i \le l_j/c_1\}| = (l_j/c_1)^r$ and $j = O(\log_\varphi^r n)$. The time complexity of the algorithm will therefore be $n + O(r \log r \log_\varphi^r n)$.

Actually, for rationally related c_i's, the result is even better. If the c_i's are rationally related, there exists a real number d such that the infinite tree can only have nodes on levels that are integer multiples of d. Since $l_j = O(\log_\varphi n)$, this implies that $j = O(1/d \cdot \log_\varphi n) = O(\log_\varphi n)$. Therefore, in this case, the time complexity of the algorithm is $n + O(r \log r \log_\varphi n)$.

3.3 Growth of the Infinite Tree

We will need to understand how the infinite tree evolves as its depth grows. Set $A_x = \{v \text{ a node } : depth(v) \le x\}$ to be the tree containing all nodes of depth at

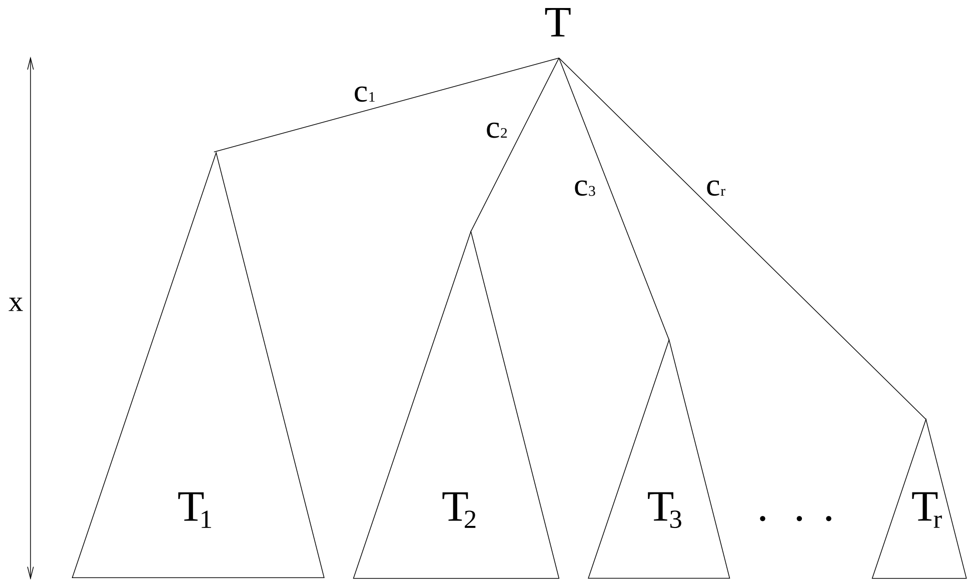


Figure 17: Tree T_i is the subtree rooted at the ith child of the root of T. If T has height x, then T_i has height $x - c_i$ and contains $F(x - c_i)$ nodes.

most x in the infinite tree. Then set

$$F(x) = \text{ number of nodes in } A_x, \qquad L(x) = \text{number of leaves in } A_x.$$

Tree A_x has a root and subtrees falling off of each of its r children. The subtree falling off of the i^{th} child has the same structure as A_{x-c_i} so (Figure 17) the equations satisfy the Fibonacci-type recurrences

$$F(x) = \begin{cases} 1 + F(x-c_1) + F(x-c_2) + \cdots + F(x-c_r), & \text{if } x \geq c_1; \\ 1, & \text{if } 0 \leq x < c_1; \\ 0, & \text{if } x < 0; \end{cases} \tag{3}$$

and

$$L(x) = \begin{cases} L(x-c_1) + L(x-c_2) + \cdots + L(x-c_r), & \text{if } x \geq c_1; \\ 1, & \text{if } 0 \leq x < c_1; \\ 0, & \text{if } x < 0. \end{cases} \tag{4}$$

Both of these equations are defined for all real x.

Now notice that if a node $u \in A_x$ has $depth(u) > x - c_1$ then any child of u will have depth greater than x; u therefore can have no children and is a leaf. If $depth(u) \geq x - c_1$, then at least the first child of u is in A_x and u is not a leaf. We have just shown that u is a leaf if and only if $depth(u) > x - c_1$ and thus

$$L(x) = F(x) - F(x - c_1). \tag{5}$$

We will now see that the behavior of the solution of $F(x)$, and thus $L(x)$ as well, will depend very strongly upon whether the c_i are rationally related or not. This qualitative difference will strongly influence the analysis of optimum lopsided trees.

Suppose that the $(c_1, \ldots, c_r)$ are given. A node can exist on level x of the infinite tree if and only if there is a path of length x in the tree; this in turn only happens if there exist nonnegative integers $a_1, \ldots, a_r \geq 0$ such that $\sum_i a_i c_i = x$.

The existence of such integers corresponds to the existence of a path containing exactly a_1 first edges from a nodes, a_2 second edges, etc.. The number of nodes on level x will be the number of paths that can be built using a_1 first edges, a_2 second edges, etc., which is

$$\sum_{\substack{a_1c_1+\cdots+a_rc_r=x\\ a_1,\ldots a_r,\geq 0}} \binom{a_1+a_2+\cdots+a_r}{a_1\ a_2\ \ldots\ a_r}$$

so

$$F(x)=\sum_{\substack{a_1c_1+\cdots+a_rc_r\leq x\\ a_1,\ldots,a_r\geq 0}} \binom{a_1+a_2+\cdots+a_r}{a_1\ a_2\ \ldots\ a_r}. \tag{6}$$

Suppose now that $(c_1, c_2) = (3, 5)$. Then $F(x)$ can only change at integer values of x that can be written in the form $3a_1 + 5a_2$, i.e., $x = 1, 3, 5, 6, 8, 9, 10\ldots$ and every integer $x \geq 10$. The fact that for $x \geq 8$ the difference between successive depths upon which nodes appear is a constant $(= 1)$ is not unique to this pair of c_i; for all c_1, c_2 with $gcd(c_1, c_2) = 1$ there exists some integer N such that for every integer $n \geq N$, there exist $a_1, a_2 \geq 0$ with $a_1c_1 + a_2c_2 = n$. In this case $F(x)$ changes exactly at integer values and remains unchanged between them.

On the other hand if $(c_1, c_2) = (3, \pi)$ then successive levels of the form $a_1c_1+a_2c_2$ tend to be closer and closer together with the difference between successive levels tending to zero.

More generally if $(c_1, \ldots, c_r)$ are rationally related then the infinite tree can only have nodes on levels that are integer multiples of $d = gcd(c_1, \ldots, c_r)$; furthermore there is some X such that if $x \geq X$ is an integral multiple of d, then the tree contains nodes on level x. By contrast, if $(c_1, \ldots, c_r)$ are irrationally related we will now see that the distance between successive levels upon which nodes appear will decrease to zero. This is a consequence of the Kronecker-Weyl theorem on the uniform distribution of the multiples of irrational numbers [21]. ([24] gives a more sophisticated application of the Kronecker-Weyl theorem to Diophantine combinations of irrationally related pairs). Let $S = \alpha_1, \alpha_2, \alpha_3, \ldots$ be a sequence of reals in the unit interval $[0, 1]$. Set $n([x, y])$ to be the number of $\alpha_1, \alpha_2, \alpha_3, \ldots, \alpha_n$ that fall in the interval $[x, y]$. Then S is *uniformly distributed* in $[0, 1]$ if, $\forall 0 \leq x < y \leq 1$,

$$\lim_{n\to\infty} \frac{n([x,y])}{n} = y - x.$$

The Kronecker–Weyl theorem states that if θ is an irrational number then the sequence $\{\theta\}, \{2\theta\}, \{3\theta\}, \{4\theta\}, \ldots$ is uniformly distributed in $[0, 1]$[1].

Suppose that $(c_1, \ldots, c_r)$ is irrationally related. This implies that there are some c_i, c_j such that c_i/c_j is irrational. We will now show that even the levels in $\{a_ic_i + a_jc_j, :\ a_i, a_j \geq 0\}$, the set that can be reached using only c_i and c_j edges, have the property that the distance between successive levels upon which nodes appear will decrease to zero. Without loss of generality we may scale by dividing by c_1 and assume that $c_i = 1$ and $c_j = \theta$ is irrational (scaling maintains the property that the differences go to 0). Let $U = \{a + b\theta, :\ a, b \geq 0\}$ be the set of level depths. Fix m and $\forall n \geq 0$, $\forall 0 \leq t < m$, define

$$U_{n,t} = \left[n + \frac{t}{m}, n + \frac{t+1}{m}\right] \cap U$$

We will show that for all n large enough and all t, $U_{n,t} \neq \emptyset$. Since this will be true for every m it will prove that the distance between successive levels upon which nodes appear will decrease to zero.

[1] Our analysis doesn't require the full power of the theorem; it only uses the fact that $\{\theta\}, \{2\theta\}, \{3\theta\}, \{4\theta\}, \ldots$ is *dense* in $[0, 1]$.

Note that if $x = a + b\theta \in U$ then $x + 1 = (a+1) + b\theta \in U$. Thus, if for some n' and specific t, $U_{n',t} \neq \emptyset$ then, $\forall n > n'$, $U_{n,t} \neq \emptyset$. From the Kronecker-Weyl theorem we know that there is some N such that $\forall 0 \leq t < m$, there is some element of $\{\theta\}$, $\{2\theta\}$, $\{3\theta\}$, $\{4\theta\}$, ..., $\{N\theta\}$ in each of the subintervals $\left[\frac{t}{m}, \frac{t+1}{m}\right]$. This then, immediately implies that $\forall n > \lceil N\theta \rceil$, and $\forall 0 \leq t < m$, $U_{n,t} \neq \emptyset$ and we are done.

We encapsulate the above comments in a lemma for later use:

Lemma 3 *Let $(c_1, \ldots, c_r)$ be an r-tuple of non-negative reals.*

- *If $(c_1, \ldots, c_r)$ is rationally related with $d = gcd(c_1, \ldots, c_r)$ then $\exists J$ such that $\forall j > J$, $l_{j+1} - l_j = d$.*
- *If $(c_1, \ldots, c_r)$ is irrationally related then $\lim_{j\to\infty}(l_{j+1} - l_j) = 0$.*

We now state the exact asymptotics of $F(x)$[2]. Note that they reflect the qualitative difference described above. The proof of this theorem is given in Section 5.

Theorem 2 *Let $(c_1, \ldots, c_r)$ be an r-tuple of non-negative reals and define*

$$F(x) = \begin{cases} 1 + F(x - c_1) + F(x - c_2) + \cdots + F(x - c_r), & \text{if } x \geq c_1; \\ 1, & \text{if } 0 \leq x < c_1; \\ 0, & \text{if } x < 0; \end{cases} \tag{7}$$

Let α be the smallest real positive root of the equation $Q(z) = 1 - z^{c_1} - z^{c_2} - \ldots z^{c_r}$ and $\varphi = 1/\alpha$. Let $c = (\sum_{i=1}^{r} c_i \varphi^{-c_i})$. Then,

1. *If $(c_1, \ldots, c_r)$ is rationally related*

$$F(x) = D(x)\varphi^x + O(\rho^x)$$

   where $D(x) = \frac{d}{c(1-\varphi^{-d})}\varphi^{-d\{x/d\}}$ is a periodic function with period d and $0 \leq \rho < \varphi$.

2. *If $(c_1, \ldots, c_r)$ is irrationally related*

$$F(x) = \frac{1}{c \ln \varphi}\varphi^x + o(\varphi^x).$$

If the $(c_1, \ldots, c_r)$ are rationally related then $F(x)$ only changes at integral multiples of d; if the $(c_1, \ldots, c_r)$ are irrationally related then, as x increases, the difference between values at which $F(x)$ changes gets smaller and smaller so $F(x)$ behaves more and more as a continuous function.

We can plug the results of the theorem into equation (5) to find the behavior of $L(x)$. We summarize the results in the next theorem.

Theorem 3 *Let $(c_1, \ldots, c_r)$ be an R-tuple of non-negative reals and define*

$$L(x) = \begin{cases} L(x - c_1) + L(x - c_2) + \cdots + L(x - c_r), & \text{if } x \geq c_1; \\ 1, & \text{if } 0 \leq x < c_1; \\ 0, & \text{if } x < 0; \end{cases} \tag{8}$$

Let α be the smallest real positive root of the equation $Q(z) = 1 - z^{c_1} - z^{c_2} - \ldots z^{c_r}$ and $\varphi = 1/\alpha$. Let $c = (\sum_{i=1}^{r} c_i \varphi^{-c_i})$. Then,

[2]The asymptotics of $F(x)$ for $r = 2$ can be derived from [12] and [24] whom, in different ways both study the function $h(x) = F(\ln x)$.

1. *If* $(c_1, \ldots, c_r)$ *is rationally related*

$$L(x) = E(x)\varphi^x + O(\rho^x)$$

where $E(x) = \frac{d}{c(1-\varphi^{-d})}\varphi^{-d\{x/d\}}(1-\varphi^{-c_1})$ *is a periodic function with period* d *and* $\rho < \varphi$.

2. *If* $(c_1, \ldots, c_r)$ *is irrationally related*

$$L(x) = \frac{1-\varphi^{-c_1}}{c\ln\varphi}\varphi^x + o(\varphi^x).$$

Note: In the rational case we are using the fact that c_1 is an integral multiple of d to find that $D(x - c_1) = D(x)$.

3.3.1 Rational vs. Irrational

The theorems look rather strange and seem to raise more questions than they answer. For example, to what does this difference between rational and non-rational cases combinatorially correspond? Also, why is the expression for $L(x)$ in the rational case so different from that in the irrational one? In the paragraphs that follow we attempt to answer these questions and provide the reader with some intuition as to what is occurring. (This explanation is tangential to the remainder of the paper and may be skipped.)

First let us deal with the combinatorial meaning. Recall that we have previously defined $l_0, l_1, l_2, \ldots$ to be the sequence of values at which $F(x)$ changes. Combinatorially, if the $(c_1, \ldots, c_r)$ are rationally related with gcd $= d$ then for all j large enough, we find that $l_{j+1} - l_j = d$. For example, if $r = 3$ with $(c_1, c_2, c_3) = (1, \frac{4}{3}, \frac{3}{2}) = \frac{1}{6}(6, 8, 9)$ so $d = \frac{1}{6}$ and $l_0, l_1, l_2, \ldots$ is

$$1, \frac{4}{3}, \frac{3}{2}, \frac{12}{6}, \frac{14}{6}, \frac{15}{6}, \frac{16}{6}, \frac{17}{6}, \frac{18}{6}, \frac{20}{6}, \frac{21}{6}, \frac{22}{6}, \frac{23}{6}, \frac{24}{6}, \frac{25}{6}, \ldots$$

The periodic term $E(x)$ in the equation for $L(x)$ is actually a corrective term that permits us to write an equation valid for all x that still reflects the fact that $L(x)$ only changes at values of the form $x = md$.

If, though, the $(c_1, \ldots, c_r)$ are not rationally related then the spacing between successive l_j are not regular. For example, if $r = 2$ and $(c_1, c_2) = (1, \sqrt{2})$ then the sequence is

$$1,\ \sqrt{2},\ 2,\ 1+\sqrt{2},\ 2\sqrt{2},\ 3,\ 2+\sqrt{2},\ 1+2\sqrt{2},\ 4,\ 3\sqrt{2},\ 3+\sqrt{2},\ 2+2\sqrt{2},\ \ldots$$

with $\lim(l_{j+1} - l_j) = 0$. Thus for any $\epsilon > 0$ no matter how small, there exists X such that $\forall x > X$, $L_\lambda(x - \epsilon) < L_\lambda(x)$; as x increases, $L(x)$ behaves more and more like a smoothly growing function of x and less and less like a jump function so there is no periodic corrective term.

We can now discuss why the expressions for $L(x)$ are so different. In order to simplify our statements we will throw away the error terms and work with

$$\overline{L}(x) = \begin{cases} E(x)\varphi^x & \text{if } (c_1, \ldots, c_r) \text{ is rationally related,} \\ \frac{1-\varphi^{-c_1}}{c\ln\varphi}\varphi^x & \text{otherwise.} \end{cases}$$

We examine the *instantaneous rate of growth* of $\overline{L}(x)$, i.e., how the growth rate of $\overline{L}(x)$ changes with x. If the two types of expressions are really the 'same' we expect this value to be the same, irrespective of whether $(c_1, \ldots, c_r)$ is rationally related or not. Note first that if $(c_1, \ldots, c_r)$ is not rationally related then, since $\overline{L}(x)$ is a differentiable function, the instantaneous rate of growth is simply

$$\overline{L}'(x) = \frac{1-\varphi^{-c_1}}{c\ln\varphi}(\ln\varphi)\varphi^x = \frac{1-\varphi^{-c_1}}{c}\varphi^x.$$

Suppose, though, that $(c_1, \ldots, c_r)$ is rationally related. As discussed above, $L(x)$ will only change at values $x = md$ for integral d, i.e., for $(m-1)d < x < md$ we find $L(x) = L((m-1)d)$ and $L(x)$ has a jump of size $\Theta(\varphi^x)$ at values $x = md$. Thus the derivative of $\overline{L}(x)$ will not exist at these values of x and is 0 everywhere else. To capture the instantaneous rate of growth at $x = md$ we must calculate the *average* rate of growth over the interval $((m-1)d, md]$ which will be

$$\frac{1}{d}(\overline{L}(md) - \overline{L}((m-1)d)) = \frac{1}{d}\frac{d\,(1-\varphi^{-c_1})}{c(1-\varphi^{-d})}\left(\varphi^{md} - \varphi^{(m-1)d}\right) = \frac{1-\varphi^{-c_1}}{c}\varphi^x.$$

We therefore see that the two different expressions for $\overline{L(x)}$, the one for the rational case and the other for the non-rational one, are the same in the very strong sense in that they grow at the same rate. It is actually the requirement that they grow similarly that makes their equations appear different.

3.3.2 The Minimum Height of a Tree

In this subsection we derive the the minimum height of a tree with n leaves and discuss basic applications in the theory of distributed broadcasting protocols. [3]

Denote the minimum height of a tree with n leaves as $H(n)$; it is given by the recurrence

$$H(n) = \min_{\substack{n_1+n_2+\cdots+n_r=n\\ n_1,n_2,\ldots,n_r\geq 0}} \left[\max_{n_i\neq 0}(H(n_i)+c_i)\right] \tag{9}$$

where $H(1) = H(0) = 0$. Instead of attempting to solve this equation directly we will examine the structure of the minimum-height trees by using the results of the previous subsection.

First notice that $H(n)$ is certainly a nondecreasing function because if $H(n) = m$ then there is a tree T of height m containing n leaves. Peeling away leaves from the tree until there are $n-1$ leaves yields a tree with height at most m, so $H(n-1) \leq m$ (note that we might have to peel away many leaves because some might be the only child of their parent and discarding them does not decrease the number of leaves).

The tree containing all nodes of depth at most x has $L(x)$ leaves so $H(L(x)) \leq x$. Suppose x is a depth at which some nodes exist. It would be convenient if we could say that $H(L(x)) = x$. Unfortunately this is not always true. Consider, for example the case in which $r = 2$, $c_1 = 1$, and $c_2 = 2$. Let $x = n$ for some large integer n and examine the tree A_n containing all nodes of depth at most n. There is only one node at depth n; the node reached by traversing n left (c_1) edges. The sibling of this node is at depth $x + \pi - 1 > \pi$ and so is not in the tree. Removing the node at depth x therefore leaves a tree with $L(x)$ leaves and depth slightly less than x so $H(L(x)) < x$.

We use the following observation instead. Returning to the recurrence relation (4) defining $L(x)$ we see that the same recurrence relation also describes the *maximum* number of leaves that a tree of height x can have. This implies that if $n > L(x)$ then $H(n) > x$. Using this fact we can derive the asymptotic behavior of $H(n)$:

[3] These applications are gone into in more detail in a companion paper [14] which applies some of the techniques developed in parts of this article to analyze generalizations of lopsided trees that model various broadcasting protocols. This section is the jumping off point for [14]

Theorem 4 *Let $(c_1, \ldots, c_r)$ be an R-tuple of non-negative reals and $H(n)$ defined as above. Let α be the smallest real positive root of the equation $Q(z) = 1 - z^{c_1} - z^{c_2} - \ldots z^{c_r}$, $\varphi = 1/\alpha$ and $c = (\sum_{i=1}^{r} c_i \varphi^{-c_i})$.*

1. *If $(c_1, \ldots, c_r)$ is rationally related with $d = gcd(c_1, \ldots, c_r)$ then*
$$H(n) = d \left\lceil \frac{1}{d} \left(\log_\varphi n - \log_\varphi \left(\frac{d\,(1-\varphi^{-c_1})}{c\,(1-\varphi^{-d})} \right) \right) \right\rceil + O(1).$$
Furthermore there is some $N > 0$ such that if $n > N$ then the $O(1)$ term is of the form $+d$, 0, or $-d$.

2. *If $(c_1, \ldots, c_r)$ is irrationally related*
$$H(n) = \log_\varphi n - \log_\varphi \frac{1-\varphi^{-c_1}}{c \ln \varphi} + o(1).$$

Proof.

We first deal with the case that $(c_1, \ldots, c_r)$ are rationally related. Recall that nodes can only appear levels of the tree that are integral multiples of d. Substituting $x = md$ into Theorem 3 yields
$$L(md) = \frac{d\,(1-\varphi^{-c_1})}{c(1-\varphi^{-d})} \varphi^{md} + o(\varphi^{md}). \tag{10}$$
This implies that there is some integer M such that for all $m > M$, $L(md - d) < L(md)$. Therefore if n is large enough there must be an integer m such that $L(md - 1) < n \leq L(md)$ implying $H(n) = md$. Inverting equation (10) completes the proof of part 1.

In the irrational case, for fixed $\epsilon > 0$, Theorem 3 implies that there is some X such that $L(x-\epsilon) < L(x)$ for all $x > X$. This in turn implies that if n is large enough we can find x such that $L(x - \epsilon) < n \leq L(x)$ and therefore that $x - \epsilon < H(n) \leq x$. Inverting $L(x) = \frac{1-\varphi^{-c_1}}{c \ln \varphi} \varphi^x + o(\varphi^x)$ yields part 2 of the theorem.

□

3.3.3 Applications

In [2] Bar-Noy and Kipnis introduce the *Postal Model* of message passing for distributed systems. In this model, counting time from when a sender first starts sending a message, the sender requires one unit of time before completing the work of sending and being able to do something else but the recipient requires λ units of time to receive and process it: λ is a parameter representing the *latency* of the system. Bar-Noy and Kipnis demonstrated that, in t time units, the maximum number of recipients that can receive a broadcast message in a one-to-many broadcast scheme in this model satisfies:
$$F_\lambda(t) = \begin{cases} 1 & \text{if } 0 \leq t \leq \lambda \\ F_\lambda(t-1) + F_\lambda(t-\lambda) & \text{if } t \geq \lambda \end{cases} \tag{11}$$
The minimum time that it takes to broadcast to n recipients in the model therefore satisfies $f_\lambda(n) = \min\{t : F_\lambda(t) \geq n\}$.

Notice, though, that $F_\lambda(t)$ is exactly the function that we have labelled $L(t)$ with parameters $c_1 = 1$, $c_2 = \lambda$. The recurrence relations are exactly the same; to check that the initial conditions match it is enough to note that for $1 \leq t < \lambda$, $L(t) = L(t-1)$ so $L(t) = 1$ for $0 \leq t < \lambda$. Therefore $f_\lambda(x) = H(x)$. Applying Theorem 4 yields
$$f_\lambda(n) = \log_\varphi n + O(1)$$

where $\varphi = 1/\alpha$, where α is the smallest positive root of $1 - z - z^\lambda$.

This improves the upper bound of

$$f_\lambda(n) \leq 2\lambda + \frac{2\lambda \log n}{\log(\lceil\lambda\rceil + 1)}$$

given in [2]. The amount of improvement depends upon the particular value of λ but, as noted in [14], the ratio of the old bound over the new one goes to

$$\frac{2\lambda \log n}{\log(\lceil\lambda\rceil + 1)} \frac{1}{\log_\varphi n} = \frac{2\lambda}{\log_\varphi(\lceil\lambda\rceil + 1)} \to 2$$

as $\lambda \to \infty$.

3.4 The Cost of Varn codes

3.4.1 Exact Asymptotics of $C(T_n)$

Combining Theorems 2 and 1 we can derive the exact asymptotics of $C(T_n)$, the cost of the optimal lopsided tree with n leaves, as $n \to \infty$. As described previously this is equivalent to exactly analyzing the costs of Varn Codes, solving an open question posed in [16]. The proof of this theorem is given in Section 6.

Theorem 5 *Let $0 < c_1 \leq c_2 \leq \cdots \leq c_r$ and T_n be a tree with minimal external path length for those parameters. Let α be the smallest real positive root of the equation $Q(z) = 1 - z^{c_1} - z^{c_2} - \ldots z^{c_r}$ and $\varphi = 1/\alpha$. Let $c = \sum_{i=1}^r c_i\varphi^{-c_i}$. Let k and $h = x_k$ be as defined in Lemma 1.*

1. *If $(c_1, \ldots, c_r)$ is rationally related with gcd$(c_1, \ldots, c_r) = 1$ define*

$$K = \frac{1}{c(1-\varphi^{-1})}\left(\varphi^{\lfloor h\rfloor} - \sum_{i=1}^{k}\varphi^{\lfloor h\rfloor - c_i} + (k-1)\right)$$

$$A = \frac{(k-1)}{c(1-\varphi^{-1})K}, \qquad R = \log_\varphi((1-A)\varphi + A).$$

Then

$$\boxed{C(T_n) = n\log_\varphi n + B\left(\{\log_\varphi \frac{n}{K}\}\right)n + D\left(\{\log_\varphi \frac{n}{K}\}\right)n + o(n)}$$

where

$$B(\theta) = h + 1 - \log_\varphi K - \theta - \left(\frac{1}{\varphi - 1} + \{h\}(1-A)\right)\varphi^{1-\theta}$$

$$D(\theta) = \begin{cases} \{h\}(\varphi^{R-\theta} - 1) & h \leq c_r \text{ and } \theta \leq R \\ 0 & h \leq c_r \text{ and } \theta > R \\ 0 & h > c_r \end{cases}$$

If $(\overline{c_1}, \ldots, \overline{c_r}) = d(c_1, \ldots, c_r)$ with gcd$(c_1, \ldots, c_r) = 1$ then $C(\overline{T}n) = d\cdot C(T_n)$ where $C(\overline{T}_n)$ is the cost function defined by $(\overline{c_1}, \ldots, \overline{c_r})$.

2. *If $(c_1, \ldots, c_r)$ is irrationally related define $K = \frac{1}{c\ln\varphi}\left(\varphi^h - \sum_{i=1}^k \varphi^{h-c_i} + (k-1)\right)$. Then*

$$\boxed{C(T_n) = n\log_\varphi n + \left(h - \log_\varphi K - \frac{1}{\ln\varphi}\right)n + o(n).}$$

See Figures 18 and 19 for examples of how this asymptotic result compares with the real value of $C(T_n)$. These figures also rather nicely illustrate a surprising aspect of the analysis; in the rational case, there are sometimes not one but two periodic functions (in $\log_\varphi n$), $B(x)$ and $D(x)$, that arise.

Sometimes but not always. There are at least two cases in which $D(x) \equiv 0$. The first is when $h > c_r$. This always occurs, for example in the binary case ($r = 2$) when $k = 2$ so $h = c_1 + c_2 > c_2$. The second case is when $\{h\} = 0$, e.g., when h is an integer. Examples of both of these cases are given in Figures 18 and 19. Figure 19 also illustrates how a small change in one of the c_i can cause a switch from $D(x) \equiv 0$ to $D(x) \not\equiv 0$.

4 Derivation of the Structure of Optimal Trees

In this section we derive the proof of Theorem 1.

4.1 Background

We start by reviewing what happens in the binary case and describing the difficulty that occurs when trying to extend this analysis to the r-ary tree case.

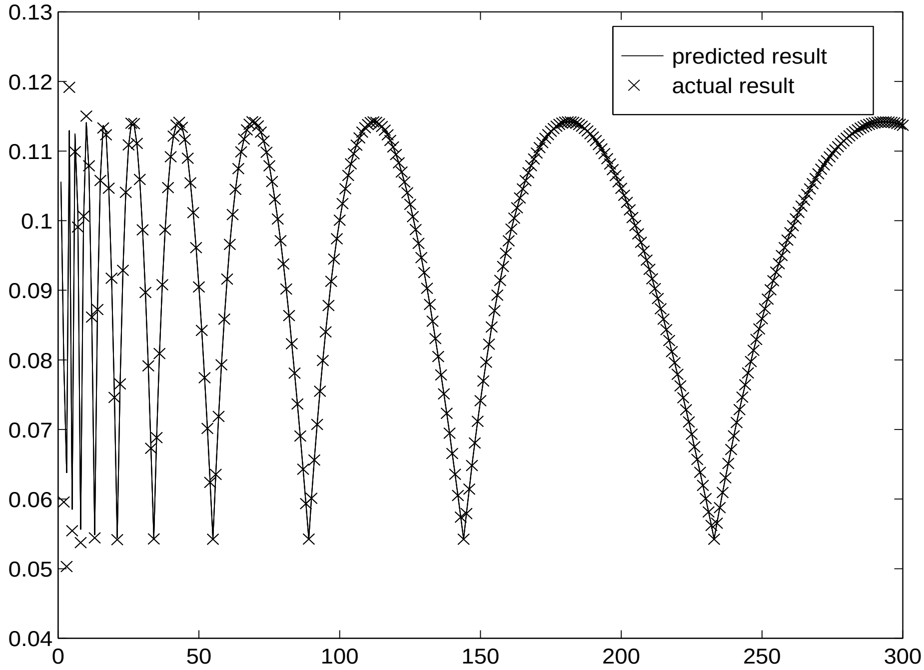
Recall that optimal trees must be proper. Thus *binary* ($r = 2$) optimal trees for n leaves have $n-1$ leaves and the procedure described by Theorem 1 creates a sequence of trees T_n^{n-1}, $n = 2, 3, 4, \ldots$. This sequence has the property that T_{n+1}^n is created from T_n^{n-1} by taking the smallest labelled leaf in T_n^{n-1}, making it internal and adding its two children. Another way of expressing this is that optimal T_{n+1}^n can be created by incrementally *branching* — taking the highest leaf in T_n^{n-1}, making it internal, and adding its two children. This is exactly the procedure described in [16]. See Figure 20 for an example. Note that the $2n-1$ *nodes* in the optimal tree are not necessarily the $2n-1$ highest nodes in the infinite tree. By contrast, as we shall soon see, the $n-1$ *internal nodes* in the optimal tree are the $n-1$ highest nodes in the infinite tree.

In the r-ary case, there are two ways to add a leaf to a tree. The first is again by a *branching* operation which transforms the minimum external node into an internal node, and adds its first two (smallest) children to the tree; the second is by an *adding* operation which adds a previously non-existing ith child ($2 < i \le r$) of some internal node to the tree. A natural extension of the above incremental algorithm is therefore to consider at each step, the cost of both *branching* and the minimum cost *adding* operations. If *adding* is cheaper, we perform the *adding*, otherwise we perform the *branching*. (This is the *"extension algorithm"* presented in [23]).

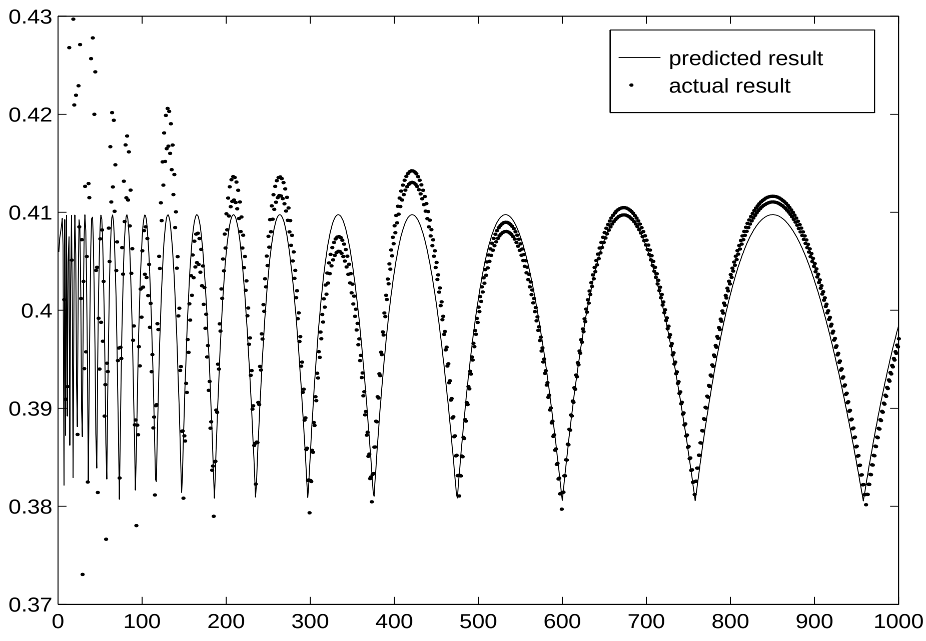
For example, if $r = 5$, $(c_1, c_2, c_3, c_4, c_5) = (3, 3, 3, 8, 8)$, the "extension algorithm" constructs the sequence of trees for $2, 3, 4, 5, 6$ leaves shown in $(a), (b), (c), (d), (e)$ of Figure 21 respectively.

However, it is not difficult to see that the tree constructed by the above algorithm is not always optimal. In the above example, the tree for 6 leaves with cost 34 in (e) is not optimal. The optimal tree T_6 should be (f) of Figure 21 which has cost 32. As pointed out in [23], the algorithm fails because it ignores the fact that it may be worth performing a new *branching* which is more expensive, if it enables a cheaper *adding* later. As in the above example, adding the 5^{th} child of node 1 is cheaper than branching at node 2 but the branching of node 2 enables the cheaper adding of another child (the 3rd child of node 2) later.

More precisely, the incremental algorithm fails because in the r-ary case, it is not always true that $T_n \subseteq T_{n+1}$, e.g., in the above example $T_5 \not\subseteq T_6$ because the 5^{th} child of node 1 is in T_5 but not in T_6. [23] uses a *"mending algorithm"* to change

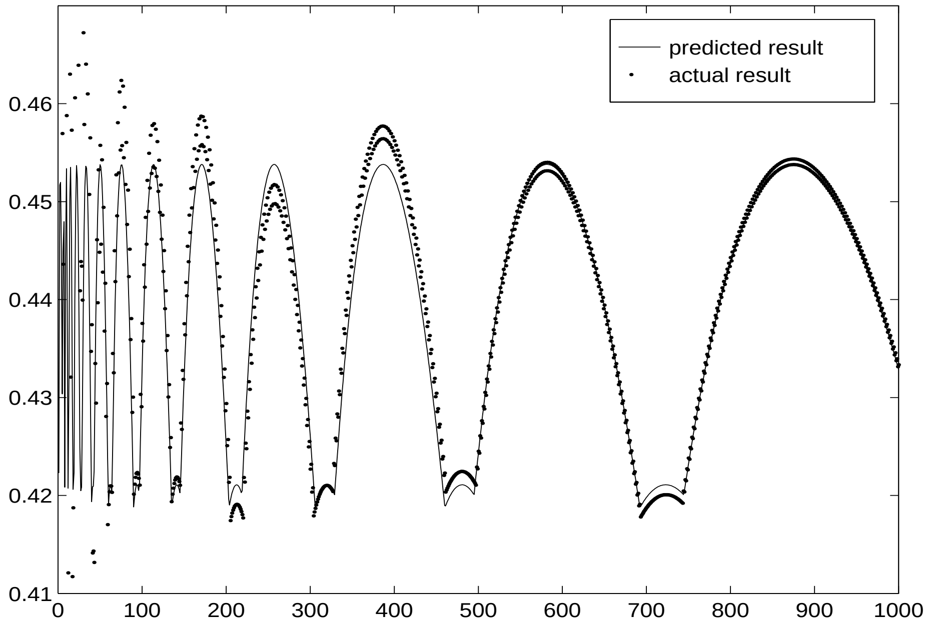


(a) $r = 2$ and $(c_1, c_2) = (1, 2)$

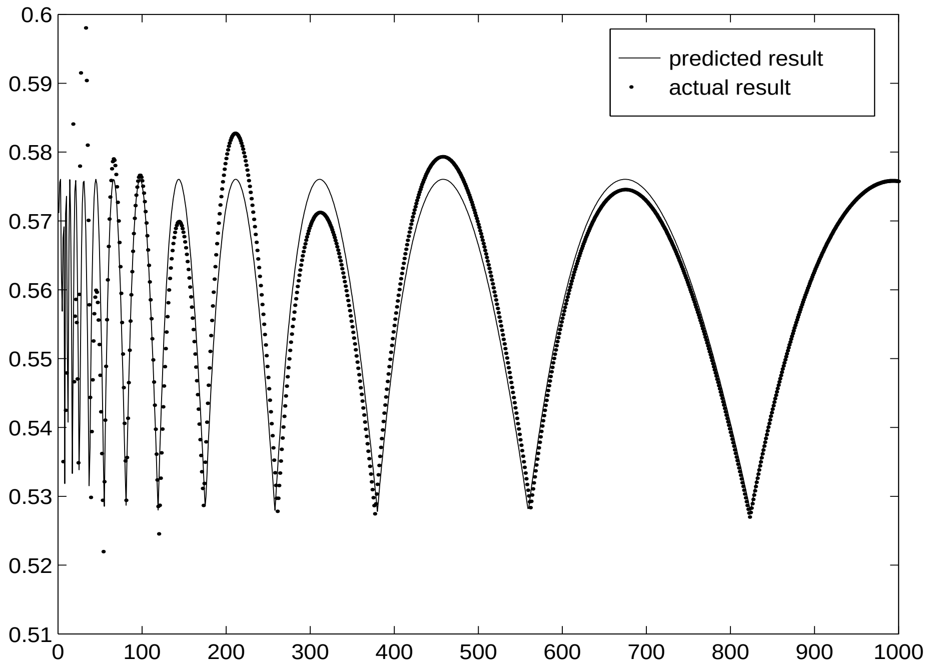


(b) $r = 3$ and $(c_1, c_2, c_3) = (3, 5, 7)$

Figure 18: The predicted cost is $B(\{\log_\varphi \frac{n}{K}\}) + D(\{\log_\varphi \frac{n}{K}\})$ as defined in Theorem 5, while the actual cost plots function $\frac{1}{n}(C(T_n) - n \log_\varphi n)$. (a) $r = 2$ and $(c_1, c_2) = (1, 2)$; $k = 2$, $h = 3$; (b) $r = 3$ and $(c_1, c_2, c_3) = (3, 5, 7)$; $k = 3$, $h = 7.5$. In both cases, since $h > c_r$, $D \equiv 0$.



(c) $r = 5$ and $(c_1, c_2, c_3, c_4, c_5) = (2, 3, 4, 7, 11)$



(d) $r = 5$ and $(c_1, c_2, c_3, c_4, c_5) = (2, 3, 5, 7, 11)$

Figure 19: The predicted cost is $B(\{\log_\varphi \frac{n}{K}\}) + D(\{\log_\varphi \frac{n}{K}\})$ as defined in Theorem 5, while the actual cost plots function $\frac{1}{n}(C(T_n) - n \log_\varphi n)$. (c) $r = 5$ and $(c_1, c_2, c_3, c_4, c_5) = (2, 3, 4, 7, 11)$; $k = 3$, $h = 4.5$; (d) $r = 5$ and $(c_1, c_2, c_3, c_4, c_5) = (2, 3, 5, 7, 11)$; $k = 3$, $h = 5$. In this case, since $\{h\} = 0$, $D \equiv 0$.

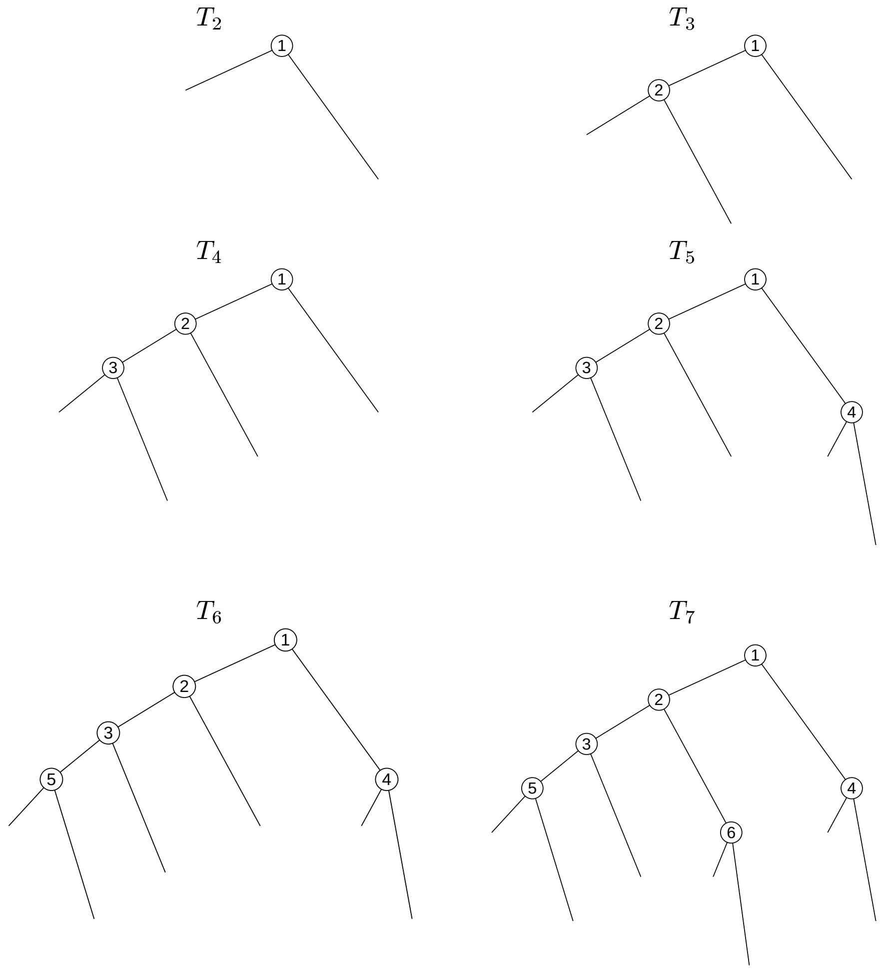


Figure 20: For $r = 2$, $(c_1, c_2) = (1, 3)$, T_{n+1} is constructed from T_n by making the highest leaf in the tree an internal node with two children.

the tree constructed by the *"extension algorithm"* into an optimal one in case it is not already optimal. This algorithm requires $O(nr^2)$ time.

Another approach to constructing optimal trees uses the fact that if m, the number of internal nodes in the optimal tree, is known, then these m nodes can be shown to be the m shallowest (i.e. least-depth) nodes of the infinite tree, while the leaves are the n shallowest available children of these nodes in the infinite tree. This type of tree is called a *shallow tree* in [15]. The trees T_n^m introduced in Definition 6 are, by definition, shallow trees. In the binary case, there is a one-one correspondence between the number of external nodes n and the number of internal nodes m, namely, $m = n - 1$. Therefore, in the binary case, an optimal tree T_n for n leaves is exactly the tree containing the highest $(n - 1)$ nodes in the infinite tree as internal nodes, each of which has both of its children in T_n.

However, for the r-ary trees, this kind of one-one correspondence between the number of internal nodes and the number of external nodes does not exist; trees with the same number of external nodes may have different numbers of internal nodes. We do know, though, that

Lemma 4 *[15] Fix n and set $m_{min} = \lceil \frac{n-1}{r-1} \rceil \leq m \leq n - 1$. Let T_n^m be as defined*

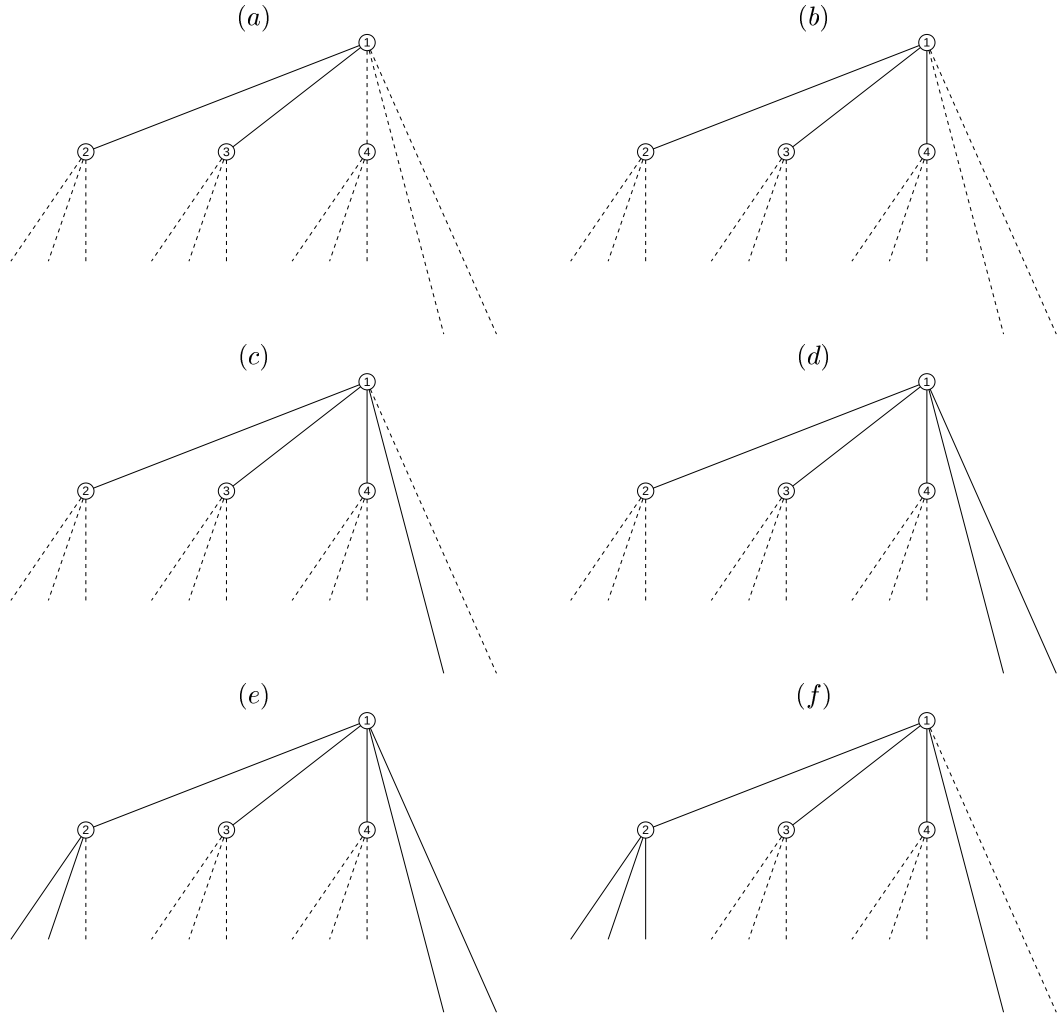


Figure 21: For $r = 5$, $(c_1, c_2, c_3, c_4, c_5) = (3, 3, 3, 8, 8)$, the extension algorithm constructs the sequence of trees for $2, 3, 4, 5, 6$ leaves shown in $(a), (b), (c), (d), (e)$ respectively. The tree in (e) is *not* optimal for 6 leaves. The tree in (f) is. Notice that T_5 in (d) is not a subtree of T_6 in (f). Dotted lines are infinite tree edges not in the tree.

in Definition 6. Then

1. *If T is any tree with n leaves and m internal nodes, then $C(T_n^m) \le C(T)$.*
2. *Let $m_{max} = min\{m : T_n^{m+1}$ is not proper$\}$. Then for all $m > m_{max}$, T_n^m is not proper.*
3. *There exists m_0, $m_{min} \le m_0 \le m_{max}$, such that $T_n^{m_0}$ is optimal.*

This lemma implies that one of the T_n^m must be optimal. To find an optimal tree, it therefore suffices to construct all the T_n^m, $m_{min} \le m \le m_{max}$, and return the one with the lowest cost. This, in fact, was the basis for the $O(n \log^2 r)$ time algorithm presented in [15]. See Figure 5 for an example.

In [15] the following observation due to R. Fleischer was reported:

Lemma 5 *The sequence of tree costs $C(T_n^m)$, $m_{min} \le m \le m_{max}$ is convex, i.e. for $m_{min} < m < m_{max}$,*

$$(C(T_n^{m+1}) - C(T_n^m)) \ge (C(T_n^m) - C(T_n^{m-1})).$$

In particular, this implies the existence of m_0 such that

$$C(T_n^{m_{min}}) > \cdots > C(T_n^{m_0-1}) > C(T_n^{m_0}) \le C(T_n^{m_0+1}) \le \cdots \le C(T_n^{m_{max}}).$$

This lemma, while quite beautiful, did not help at all with the analysis of the algorithm of [15]. In our paper here it will be of tremendous use, though, because it provides a *local* test of the optimality of any particular proper T_n^m. Simply compare T_n^m to its predecessor (if the predecessor exists, i.e., $m > m_{min}$) and its successor (if the successor is proper, i.e., $m < m_{max}$). We encapsulate this fact in a lemma:

Lemma 6 *T_n^m is optimal for n leaves if and only if T_n^m is proper and both of the following are true.*

- *$m = m_{min}$ or $m > m_{min}$ and $C(T_n^{m-1}) > C(T_n^m)$.*
- *$m = m_{max}$ or $m < m_{max}$ and $C(T_n^{m+1}) \ge C(T_n^m)$.*

4.2 Evolution of Shallow Trees

We now try to understand how the cost of T_n^m changes as m increases.

Let n be fixed and T_n^m and T_n^{m+1} be any two successive proper trees. (Figure 22 shows two successive proper trees.) Recall that $T_n^m = V_m \cup LEAF_n(V_m)$. Rewrite $LEAF_n(V_m) = \{u_1, u_2, \ldots, u_n\}$ where $u_1 < u_2 < \cdots < u_n$ (where we are now using the label of a node as its name).

By definition u_1, the smallest node in $LEAF_n(V_m)$ is the node labelled $m+1$ so $V_{m+1} = V_m \cup \{u_1\}$. Now let $child_i(w)$ denote the ith child of w. Then the set of n smallest children in $LEAF(V_{m+1})$ is the set of n smallest children in

$$\{u_2, \ldots, u_n\} \cup \{child_i(u_1) : i = 1, \ldots, r\}$$

so

$$LEAF_n(V_{m+1}) = \{u_2, \ldots, u_{n+1-d}\} \cup \{child_i(u_1) : i = 1, \ldots, d\}$$

where

$$\begin{aligned} d &= \max\{i : child_i(u_1) < u_{n+2-i}, i = 1, \ldots, r\} \\ &= \text{degree of } m+1 \text{ in } T_n^{m+1}. \end{aligned}$$

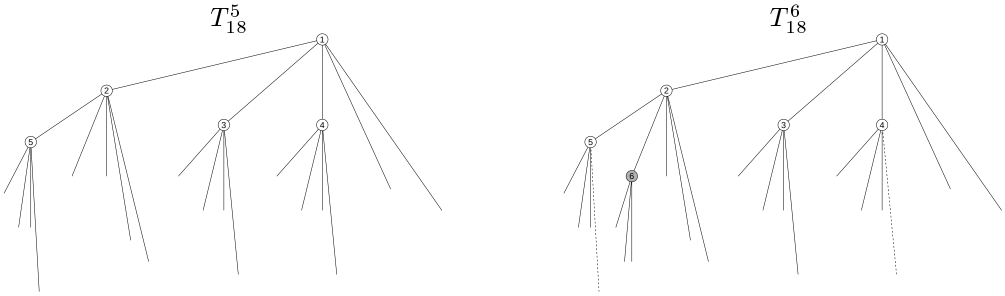


Figure 22: Two successive trees T_{18}^5 and T_{18}^6. u_1 is node 6 and $d = 3$. The dotted edges are not in the tree.

By assumption, T_n^{m+1} is proper, so $d \geq 2$. Then

$$T_n^{m+1} = V_{m+1} \cup \{u_2, \ldots, u_{n+1-d}\} \cup \{child_i(u_1) : i = 1, \ldots, d\}.$$

The (long) remainder of this section will be devoted to finding d. For later use we define

Definition 8 *Let T_n^m be any proper tree. Then $deg(T_n^m) \stackrel{\text{def}}{=}$ the number of children of internal node m in T_n^m.*

Note that when m, n are fixed then $d = deg(T_n^{m+1})$. Now introduce

Definition 9

$$\begin{aligned} S(l,t) &\stackrel{\text{def}}{=} \sum_{i=1}^{t} (l + c_i) - l \\ &= \textit{cost change in making an external node of depth } l \textit{ internal with } t \textit{ children} \\ L(l,t) &\stackrel{\text{def}}{=} t * l \\ &= \textit{the cost of } t \textit{ nodes of depth } l \end{aligned}$$

In this notation,

$$\begin{aligned} C(T_n^{m+1}) - C(T_n^m) &= \sum_{i=1}^{d} depth(child_i(u_1)) - depth(u_1) - \sum_{i=1}^{d-1} depth(u_{n+1-i}) \\ &= S(depth(u_1), d) - \sum_{i=1}^{d-1} depth(u_{n+1-i}). \end{aligned} \tag{12}$$

Note that since u_1 is the highest leaf in T_n^m and is also the deepest internal node in T_n^{m+1}

$$\min_{e \in EX(T_n^m)} depth(e) = depth(u_1) = \max_{v \in IN(T_n^{m+1})} depth(v);$$

since $d \geq 2$ and $u_{n-d+2} \in T_n^{m+1}$,

$$\max_{e \in EX(T_n^m)} depth(e) = depth(u_n) \geq depth(u_{n-d+2}) \geq \min_{u \in UN(T_n^{m+1})} depth(u).$$

Therefore, plugging back into (12),

$$\begin{aligned} &C(T_n^{m+1}) - C(T_n^m) \\ &\quad \leq S\left(\max_{v \in IN(T_n^{m+1})} depth(v), d\right) - L\left(\min_{u \in UN(T_n^{m+1})} depth(u), d-1\right) \end{aligned} \tag{13}$$

and

$$C(T_n^{m+1}) - C(T_n^m)$$
$$\geq S\left(\min_{e \in EX(T_n^m)} depth(e), d\right) - L\left(\max_{e \in EX(T_n^m)} depth(e), d-1\right) \qquad (14)$$

where $d = deg(T_n^{m+1})$.

The following lemma, previously stated without proof, will be crucial in allowing us to take advantage of the previous two equations.

Lemma 1 *Let* $x_m = \frac{\sum_{i=1}^{m} c_i}{m-1}$ *for* $m = 2, \ldots, r$. *There exists* $k \geq 2$ *such that*

$$x_2 \geq x_3 \geq \cdots \geq x_{k-1} \geq x_k < x_{k+1} < \cdots < x_r. \qquad (15)$$

(If $x_2 \leq x_3$ *set* $k = 2$. *If* $x_2 \geq x_3 \geq \cdots \geq x_{r-1} \geq x_r$ *set* $k = r$.*) Letting* k *be this value and setting* $h \stackrel{\text{def}}{=} x_k$ *we have, further, that, if* $k < r$ *then* $c_k \leq h < c_{k+1}$.

Proof. Starting with $x_{m-1} < x_m$,

$$\begin{aligned}
& \frac{\sum_{i=1}^{m-1} c_i}{m-2} < \frac{\sum_{i=1}^{m} c_i}{m-1} \\
\Longrightarrow \quad & (m-1)\sum_{i=1}^{m-1} c_i < (m-2)\sum_{i=1}^{m} c_i = (m-2)\sum_{i=1}^{m-1} c_i + (m-2)c_m \\
\Longrightarrow \quad & \sum_{i=1}^{m-1} c_i < (m-2)c_m \\
\Longrightarrow \quad & \sum_{i=1}^{m} c_i < (m-1)c_m \leq (m-1)c_{m+1} \\
\Longrightarrow \quad & m\sum_{i=1}^{m} c_i < (m-1)\sum_{i=1}^{m} c_i + (m-1)c_{m+1} = (m-1)\sum_{i=1}^{m+1} c_i \\
\Longrightarrow \quad & \frac{\sum_{i=1}^{m} c_i}{m-1} < \frac{\sum_{i=1}^{m+1} c_i}{m} \text{ (i.e., } x_m < x_{m+1})
\end{aligned}$$

Therefore, there exists $k \geq 2$ such that

$$x_2 \geq x_3 \geq \cdots \geq x_{k-1} \geq x_k < x_{k+1} < \cdots < x_r,$$

proving (2).

Now let $h \stackrel{\text{def}}{=} x_k$. To prove the second part of the lemma we first use the fact that $x_k \leq x_{k-1}$ so

$$\begin{aligned}
& \frac{\sum_{i=1}^{k} c_i}{k-1} \leq \frac{\sum_{i=1}^{k-1} c_i}{k-2} \\
\Longrightarrow \quad & (k-2)\sum_{i=1}^{k} c_i \leq (k-1)\sum_{i=1}^{k-1} c_i \\
\Longrightarrow \quad & (k-2)c_k \leq \sum_{i=1}^{k-1} c_i \\
\Longrightarrow \quad & (k-1)c_k \leq \sum_{i=1}^{k} c_i \\
\Longrightarrow \quad & c_k \leq \frac{\sum_{i=1}^{k} c_i}{k-1} = h
\end{aligned}$$

Similarly, since $x_k < x_{k+1}$,

$$\begin{aligned}
& \frac{\sum_{i=1}^{k} c_i}{k-1} < \frac{\sum_{i=1}^{k+1} c_i}{k} \\
\Longrightarrow\ & k \sum_{i=1}^{k} c_i < (k-1) \sum_{i=1}^{k+1} c_i \\
\Longrightarrow\ & \sum_{i=1}^{k} c_i < (k-1) c_{k+1} \\
\Longrightarrow\ & h = \frac{\sum_{i=1}^{k} c_i}{k-1} < c_{k+1}
\end{aligned}$$

Thus, $c_k \leq h < c_{k+1}$. □

For example, when $r = 5$ and $(c_1, c_2, c_3, c_4, c_5) = (3, 5, 5, 8.75, 10)$ (these are the parameters used in many of our examples, then

$$(x_2, x_3, x_4, x_5) = (8, 6.5, 7.25, 7.9375)$$

so $k = 3$ and $h = 6.5$. Notice that $5 = c_3 \leq 6.5 < 8.75 = c_4$.

This last lemma permits bounding the change in the cost of shallow trees as their number of internal nodes grow. We first note that replacing $(d-1)$ leaves of depth $l + h$ by making a leaf of depth l into an internal node with d children will not decrease the cost of the tree.

Lemma 7 *$S(l, d) \geq L(l + h, d - 1)$ for any real l, any integer $d \geq 2$.*

Proof.

$$\begin{aligned}
S(l,d) - L(l+h, d-1) &= \sum_{i=1}^{d} (l + c_i) - l - (d-1) * (l+h) \\
&= \sum_{i=1}^{d} c_i - (d-1) * h \\
&= (d-1) \left[\frac{\sum_{i=1}^{d} c_i}{d-1} - h \right] \geq 0 \text{ (since } h \leq x_d)
\end{aligned}$$

where the last inequality follows from Lemma 1. □

We also note that if an internal node of depth l has exactly k children, then modifying the tree by removing all k children of this node (so that itself becomes an external node) and adding $(k-1)$ new leaves with depth larger than $l + h$, will increase the cost of the tree.

Lemma 8 *For any real $\delta > 0$, $S(l, k) < L(l + h + \delta, k - 1)$. In particular, if $l = l_j$ and $\delta = l_{j+1} - l_j > 0$, we have $S(l_j, k) < L(l_{j+1} + h, k - 1)$.*

Proof. Recall that

$$h = \frac{\sum_{i=1}^{k} c_i}{k-1},$$

Now, for any $\delta > 0$,

$$\sum_{i=1}^{k} c_i = (k-1) * h < (k-1) * (h + \delta),$$

$$\Longrightarrow \sum_{i=1}^{k} c_i + (k-1)l < (k-1) * (l+h+\delta).$$

i.e. $S(l,k) < L(l+h+\delta, k-1)$. □

We now prove Theorem 1.

Proof [of Theorem 1]**.** We prove the theorem case by case. For each case, we prove the optimality of T_n^m by showing both

(a) if $m < m_{max}$ (i.e. T_n^{m+1} is proper), then $C(T_n^m) \le C(T_n^{m+1})$ and

(b) if $m > m_{min}$ (i.e. T_n^{m-1} exists), then $C(T_n^{m-1}) > C(T_n^m)$.

The optimality of T_n^m will follow from Lemma 5.

First recall that, if T_n^{m+1} is proper then, by (13),

$$C(T_n^{m+1}) - C(T_n^m) \ge S\left(\min_{e\in EX(T_n^m)} depth(e), d_1\right) - L\left(\max_{e\in EX(T_n^m)} depth(e), d_1 - 1\right) \quad (16)$$

where $d_1 = deg(T_n^{m+1}) \ge 2$.

If T_n^{m-1} exists, then by (14),

$$C(T_n^{m-1}) - C(T_n^m) \ge L\left(\min_{w\in UN(T_n^m)} depth(w), d_2 - 1\right) - S\left(\max_{v\in IN(T_n^m)} depth(v), d_2\right) \quad (17)$$

where $d_2 = deg(T_n^m)$.

1. $n = a_j$, $T_{a_j}^{m_j} = V_{m_j} \cup A_j$. (See Figure 8.) Clearly, $T_{a_j}^{m_j}$ is proper and

   $$\min_{e\in EX\left(T_{a_j}^{m_j}\right)} depth(e) = l_{j+1}, \qquad \max_{e\in EX\left(T_{a_j}^{m_j}\right)} depth(e) \le l_j + h, \qquad \max_{v\in IN\left(T_{a_j}^{m_j}\right)} depth(v) = l_j$$

   and $deg(T_{a_j}^{m_j}) = k$ since $c_k \le h < c_{k+1}$ (Lemma 1).

   Hence, if $T_{a_j}^{m_j+1}$ proper, by (16),

   $$\begin{aligned} C(T_{a_j}^{m_j+1}) - C(T_{a_j}^{m_j}) &\ge S(l_{j+1}, d_1) - L(l_j + h, d_1 - 1) \\ &\ge S(l_j, d_1) - L(l_j + h, d_1 - 1) \\ &\ge 0 \text{ (by Lemma 7)} \end{aligned}$$

   proving (a).

   If $T_{a_j}^{m_j-1}$ exists, then

   $$\min_{w\in UN\left(T_{a_j}^{m_j}\right)} depth(w) = l_j + h + \delta, \text{ for some } \delta > 0$$

   and, by (17),

   $$\begin{aligned} C(T_{a_j}^{m_j-1}) - C(T_{a_j}^{m_j}) &\ge L(l_j + h + \delta, k-1) - S(l_j, k) \\ &> 0 \text{ (by Lemma 8)} \end{aligned}$$

   proving (b).

   Therefore, $T_n = T_{a_j}^{m_j}$ is optimal for n leaves.

2. $a_j < n \le b_j$, $T_n^{m_j} = V_{m_j} \cup LEAF_n(V_{m_j})$. (See Figure 9.) Clearly, $T_n^{m_j}$ is proper and

   $$\min_{e\in EX\left(T_n^{m_j}\right)} depth(e) = l_{j+1}, \qquad \max_{e\in EX\left(T_n^{m_j}\right)} depth(e) \le l_{j+1} + h, \qquad \max_{v\in IN\left(T_n^{m_j}\right)} depth(v) = l_j$$

Hence, if $T_n^{m_j+1}$ is proper, by (16),

$$\begin{aligned} C(T_n^{m_j+1}) - C(T_n^{m_j}) &\geq S(l_{j+1}, d_1) - L(l_{j+1}+h, d_1-1) \\ &\geq 0 \text{ (by Lemma 7)} \end{aligned}$$

proving (a).

If $T_n^{m_j-1}$ exists, then

$$\min_{w \in UN\left(T_n^{m_j}\right)} depth(w) = l_j + h + \delta, \text{ for some } \delta > 0$$

and $d = deg(T_n^{m_j}) \geq k$ since $h \geq c_k$ (Lemma 1). By (17),

$$\begin{aligned} C(T_n^{m_j-1}) - C(T_n^{m_j}) &\geq L(l_j+h+\delta, d-1) - S(l_j, d) \\ &= L(l_j+h+\delta, k-1) + (d-k) * (l_j+h+\delta) \\ &\quad -S(l_j, k) - (l_j + c_{k+1}) - \cdots - (l_j + c_d) \end{aligned}$$

Since $l_j + c_i \leq \min_{w \in UN\left(T_n^{m_j}\right)} depth(w) = l_j + h + \delta$, for $i \leq d$,

$$\begin{aligned} C(T_n^{m_j-1}) - C(T_n^{m_j}) &\geq L(l_j+h+\delta, k-1) - S(l_j, k) \\ &> 0 \text{ (by Lemma 8)} \end{aligned}$$

proving (b).

Therefore, $T_n = T_n^{m_j}$ is optimal for n leaves.

3. $b_j < n \leq a_{j+1}$. We write $n = b_j + p(k-1) + q$, where $p = \lfloor (n-b_j)/(k-1) \rfloor$ and $q = (n - b_j) \bmod (k-1)$. Before starting, let L_j be the set of nodes on level l_j. From the definitions of a_j and b_j it is not hard to see that $a_{j+1} = b_j + (k-1)|L_{j+1}|$.

   Thus, for n in the range that we are examining, $p \leq |L_j|$ with $p = L_j$ if and only if $n = a_{j+1}$ (in which case $q = 0$). Note that if $n = a_{j+1}$ then the tree constructed by this part of the theorem is exactly $T_{a_{j+1}}^{m_{j+1}}$ (see the comments following the statement of Theorem 1 for a more detailed explanation) which, by the first part of the theorem, is already known to be optimal. For this reason we will restrict the remainder of the proof to $n < a_{j+1}$ which in turn implies that $p < |L_{j+1}|$.

   There are two cases:

   (A) $p > 0$ and $q = 0$: $T_n^{m_j+p} = V_{m_j+p} \cup LEAF_n(V_{m_j+p})$. (See Figure 10.) Clearly, $T_n^{m_j+p}$ is proper. Since $p < |L_{j+1}|$ at least one node in L_{j+1} remains in $EX\left(T_n^{m_j+p}\right)$ implying that

   $$\min_{e \in EX\left(T_n^{m_j+p}\right)} depth(e) = l_{j+1}.$$

   As in the previous parts of the proof,

   $$\max_{e \in EX\left(T_n^{m_j+p}\right)} depth(e) \leq l_{j+1} + h, \qquad \max_{v \in IN\left(T_n^{m_j+p}\right)} depth(v) = l_{j+1}$$

   and $deg(T_n^{m_j+p}) = k$ since $c_k \leq h < c_{k+1}$.

   Hence, if $T_n^{m_j+p+1}$ is proper, by (16),

   $$\begin{aligned} C(T_n^{m_j+p+1}) - C(T_n^{m_j+p}) &\geq S(l_{j+1}, d_1) - L(l_{j+1}+h, d_1-1) \\ &\geq 0 \text{ (by Lemma 7)} \end{aligned}$$

proving (a).
If $T_n^{m_j+p-1}$ exists, then

$$\min_{w \in UN\left(T_n^{m_j+p}\right)} depth(w) = l_{j+1} + h + \delta, \text{ for some } \delta > 0$$

and by (17),

$$\begin{aligned} C(T_n^{m_j+p-1}) - C(T_n^{m_j+p}) &\geq L(l_{j+1} + h + \delta, k-1) - S(l_{j+1}, k) \\ &> 0 \text{ (by Lemma 8)} \end{aligned}$$

proving (b).
Therefore, $T_n = T_n^{m_j+p}$ is optimal for n leaves.

(B) $p \geq 0$ and $k - 1 > q > 0$:
Note that if $m_j + p > m_{min} = \lceil \frac{n-1}{r-1} \rceil$, then $T_n^{m_j+p}$ exists. (See Figure 11.)

case(i) Both $T_n^{m_j+p}$ exists and $C(T_n^{m_j+p+1}) \geq C(T_n^{m_j+p})$:
In this case, we prove $T_n = T_n^{m_j+p}$ is optimal. To do so we must show that $C(T_n^{m_j+p-1}) > C(T_n^{m_j+p})$ if $T_n^{m_j+p-1}$ exists.
Suppose that $T_n^{m_j+p-1}$ exists. Then

$$\min_{w \in UN\left(T_n^{m_j+p}\right)} depth(w) = l_{j+1} + h + \delta, \text{ for some } \delta > 0$$

If $p = 0$, then

$$\max_{v \in IN\left(T_n^{m_j+p}\right)} depth(v) = l_j$$

and $d = deg(T_n^{m_j}) \geq k$ since $h \geq c_k$ (Lemma 1). By (17),

$$\begin{aligned} C(T_n^{m_j+p-1}) - C(T_n^{m_j+p}) &\geq L(l_{j+1} + h + \delta, d-1) - S(l_j, d) \\ &= L(l_{j+1} + h + \delta, k-1) + (d-k) * (l_{j+1} + h + \delta) \\ &\quad -S(l_j, k) - (l_j + c_{k+1}) - \cdots - (l_j + c_d) \end{aligned}$$

Since $l_j + c_i \leq \min_{w \in UN\left(T_n^{m_j+p}\right)} depth(w) = l_{j+1} + h + \delta$, for $i \leq d$,

$$\begin{aligned} C(T_n^{m_j+p-1}) - C(T_n^{m_j+p}) &\geq L(l_{j+1} + h + \delta, k-1) - S(l_j, k) \\ &> L(l_j + h + \delta, k-1) - S(l_j, k) \\ &> 0 \text{ (by Lemma 8)} \end{aligned}$$

proving (b).
Otherwise $p > 0$,

$$\max_{v \in IN\left(T_n^{m_j+p}\right)} depth(v) = l_{j+1}$$

and $deg(T_n^{m_j+p}) = k$. By (17),

$$\begin{aligned} C(T_n^{m_j+p-1}) - C(T_n^{m_j+p}) &\geq L(l_{j+1} + h + \delta, k-1) - S(l_{j+1}, k) \\ &> 0 \text{ (by Lemma 8)} \end{aligned}$$

proving (b).
Therefore, $T_n = T_n^{m_j+p}$ is optimal for n leaves. g

case(ii) Either $T_n^{m_j+p}$ does not exist or $C(T_n^{m_j+p+1}) < C(T_n^{m_j+p})$:

In this case, we prove that $T_n^{m_j+p+1}$ is optimal. To do so we must show that $C(T_n^{m_j+p+2}) \geq C(T_n^{m_j+p+1})$ if $T_n^{m_j+p+2}$ is proper.

$$\min_{e \in EX\left(T_n^{m_j+p+1}\right)} depth(e) \geq l_{j+1}, \qquad \max_{e \in EX\left(T_n^{m_j+p+1}\right)} depth(e) \leq l_{j+1}+h.$$

Hence, if $T_n^{m_j+p+2}$ is proper, by (16),

$$\begin{aligned} C(T_n^{m_j+p+2}) - C(T_n^{m_j+p+1}) &\geq S(l_{j+1}, d_1) - L(l_{j+1}+h, d_1-1) \\ &\geq 0 \text{ (by Lemma 7)} \end{aligned}$$

proving (a).

Therefore, $T_n = T_n^{m_j+p+1}$ is optimal for n leaves.

□

5 Analysis of $F(x)$

In this section we prove Theorem 2. Recall its statement: *Let $(c_1, \ldots, c_r)$ be an r-tuple of non-negative reals and define*

$$F(x) = \begin{cases} 1 + F(x-c_1) + F(x-c_2) + \cdots + F(x-c_r), & \textit{if } x \geq c_1; \\ 1, & \textit{if } 0 \leq x < c_1; \\ 0, & \textit{if } x < 0; \end{cases} \tag{18}$$

Let α be the smallest real positive root of the equation $Q(z) = 1 - z^{c_1} - z^{c_2} - \ldots z^{c_r}$ and $\varphi = 1/\alpha$. Let $c = (\sum_{i=1}^{r} c_i \varphi^{-c_i})$. Then,

1. *If $(c_1, \ldots, c_r)$ is rationally related*

$$F(x) = D(x)\varphi^x + o(\rho^x)$$

where $D(x) = \frac{d}{c(1-\varphi^{-d})}\varphi^{-d\{x/d\}}$ is a periodic function with period d and $0 \leq \rho < \varphi$.

2. *If $(c_1, \ldots, c_r)$ is irrationally related*

$$F(x) = \frac{1}{c \ln \varphi}\varphi^x + o(\varphi^x).$$

Proof. Our proof will proceed in stages. We first restrict the analysis to the case in which the $(c_1, \ldots, c_r)$ are integers such that $gcd(c_1, \ldots, c_r) = 1$ and prove the correctness of the theorem using generating functions. We then show how to scale this result to prove the theorem for all rationally related cases. We conclude by analyzing the irrationally related case via Mellin Transform-like techniques. To start we will need the following simple lemma[4]

Lemma 9 *Let $Q(z) = 1 - z^{c_1} - z^{c_2} - \ldots z^{c_r}$ where either the $(c_1, \ldots, c_r)$ are positive integers such that $gcd(c_1, \ldots, c_r) = 1$ or the $(c_1, \ldots, c_r)$ are irrationally related. Let α be the smallest positive root of $Q(z)$. Then*

1. *α is a simple root with $0 < \alpha < 1$.*
2. *If $Q(z) = 0$ and $z \neq \alpha$, then $|z| > \alpha$.*

[4]The authors thank Xavier Gourdon for suggesting this approach.

Proof. Notice first that $\alpha \in (0,1)$ because $1 = Q(0) > 0 > Q(1) = 1 - r$. Furthermore, $Q'(\alpha) \neq 0$, so α must be a simple root.

To prove (2.) suppose that $z = \beta e^{i\theta}$ is another root with $0 \leq \beta \leq \alpha$, $0 \leq \theta < 2\pi$. If $\beta \leq \alpha$ then

$$\begin{aligned} \Re(Q(z)) &= 1 - \beta^{c_1}\Re(e^{ic_1\theta}) - \beta^{c_2}\Re(e^{ic_2\theta}) - \ldots - \beta^{c_r}\Re(e^{ic_r\theta}) \\ &\geq 1 - \alpha^{c_1} - \alpha^{c_2} - \ldots - \alpha^{c_r} = 0 \end{aligned}$$

where $\Re(z)$ is the real part of z. Equality holds in the equation if and only if $\beta = \alpha$ and $\Re(e^{ic_j\theta}) = 1$ for all $1 \leq j \leq r$. In other words there must exist positive integers k_j such that $c_j\theta = 2\pi k_j$, $1 \leq j \leq r$. This in turn implies $c_j = \frac{2\pi}{\theta}k_j$ contradicting either the fact that $gcd(c_1, \ldots, c_r) = 1$ in the rational case or that the $(c_1, \ldots, c_r)$ are irrationally related. Therefore $Q(z) = 0$ and $|z| \leq \alpha$ imply $z = \alpha$. $\square$

We can now prove the theorem for the case that the $(c_1, \ldots, c_r)$ are integers with $gcd(c_1, \ldots, c_r) = 1$ through the use of straightforward generating-function techniques. Referring back to (3) we see that

$$\begin{aligned} G(z) &= \sum_{n=0}^{\infty} F(n)z^n \\ &= \sum_{n=0}^{c_1-1} z^n + \sum_{n=c_1}^{\infty} \left[1 + F(n-c_1) + F(n-c_2) + \cdots + F(n-c_r)\right] z^n \\ &= \sum_{n=0}^{c_1-1} z^n + \sum_{n=c_1}^{\infty} z^n + \sum_{n=0}^{\infty} F(n-c_1)z^n + \ldots + \sum_{n=0}^{\infty} F(n-c_r)z^n \\ &= \frac{1}{1-z} + z^{c_1}\sum_{n=0}^{\infty} F(n)z^n + \ldots + z^{c_r}\sum_{n=0}^{\infty} F(n)z^n \\ &= \frac{1}{1-z} + (z^{c_1} + \ldots + z^{c_r})G(z) \end{aligned}$$

so

$$G(z) = \frac{1}{(1-z)(1 - z^{c_1} - \ldots - z^{c_r})} = \frac{1}{(1-z)Q(z)}.$$

Now, $F(n)$ is the coefficient of z^n in $G(z)$ so by Lemma 9 and standard generating function techniques [11], $F(n) = k\varphi^n + O(\rho^n)$, where α is the smallest root of $Q(z)$, $\varphi = 1/\alpha$,

$$k = -\frac{1}{\alpha(1-\alpha)\frac{dQ}{d\alpha}(\alpha)} = \frac{1}{(1-\varphi^{-1})}\frac{1}{\sum_i c_i\varphi^{-c_i}} = \frac{1}{c\,(1-\varphi^{-1})},$$

and $0 < \rho < \varphi$ (actually ρ can be taken to be any value $\frac{1}{\alpha'} - \epsilon$ where α' is the modulus of the second smallest modulus root of $Q(z)$ and $\epsilon \geq 0$ is any arbitrary value). Since $F(x)$ only changes at integral values of x this requires

$$F(x) = F(\lfloor x \rfloor) = \frac{c}{1-\varphi^{-1}}\varphi^{-\{x\}}\varphi^n + O(\rho^n)$$

proving the theorem in the rational case when $gcd(c_1, \ldots, c_r) = 1$.

To prove the theorem for the rational case in which $d = gcd(c_1, \ldots, c_r) \neq 1$ let $({c_1}', \ldots, {c_r}') = \frac{1}{d}(c_1, \ldots, c_r)$. Then $gcd({c_1}', \ldots, {c_r}') = 1$. Define

$$F'(x) = \begin{cases} 1 + F'(x - {c_1}') + F'(x - {c_2}') + \cdots + F'(x - {c_r}'), & \text{if } x \geq {c_1}'; \\ 1, & \text{if } 0 \leq x < {c_1}'; \\ 0, & \text{if } x < 0; \end{cases} \tag{19}$$

set $Q'(z) = 1 - z^{c'_1} - z^{c'_2} - \ldots z^{c'_r}$, let α' be its smallest positive root and $\varphi' = 1/\alpha'$.

From equation (6) it is easy to see that for all $x \geq 0$ $F(x) = F'(x/d)$. Furthermore $Q(x) = Q'(x^d)$ so $\varphi' = \varphi^d$. Finally

$$c' = \sum_i c_i' \left(\varphi'\right)^{-c_i'} = \frac{1}{d} \sum_i c_i \varphi^{-c_i} = \frac{c}{d}$$

Combining all of these facts yields

$$\begin{aligned} F(x) &= F'(x/d) = \frac{1'}{c'\left(1-\varphi'^{-1}\right)} \varphi'^{-\{x/d\}} \varphi'^{x/d} + O\left(\left(\rho'\right)^{x/d}\right) \\ &= \frac{d}{c\left(1-\varphi^{-d}\right)} \varphi^{-d\{x/d\}} \varphi^{x} + O\left(\rho^{x}\right). \end{aligned}$$

where $\rho' < \varphi'$ and $\rho = (\rho')^{1/d} < (\varphi')^{1/d} = \varphi$. This proves the theorem for all rational cases. To prove it for the irrational case we will calculate the Mellin transform of $F(\ln x)$ and find the asymptotics of $F(\ln x)$ by taking the inverse Mellin transform. To perform this last step we will use the following lemma due to Fredman and Knuth [12] which is in turn a modification of an earlier result due to Landau:

Lemma 10 *[[12] Lemma 4.3] Let $f(t)$ be a nondecreasing function of the real variable t, with $f(t) \geq 0$. Assume that $G(s) = \int_1^\infty f(t)dt/t^{s+1}$ is an analytic function of the complex variable s when $\Re(s) \geq \gamma > 0$, except for a first-order pole at $s = \gamma$ with positive residue C. Then $f(t) \sim Ct^\gamma$.*

Note: If, as will occur in the functions $f(t)$ that we examine, $f(t) = 0$ for $t \leq, 1$ then $G(-s)$ is the Mellin transform of $f(t)$ and the lemma is revealed to be a special case of the inversion theorem for Mellin-transforms. Note too that $f(x) \sim g(x)$ means that $f(x) = g(x) + o(g(x))$.

Define the function $f(t)$ by $f(t) = 0$ for $t \leq 1$ and $f(t) = F(\ln t)$ for $t \geq 1$. Setting $d_j = e_j^c$ we find that for $t \geq 1$

$$\begin{aligned} f(t) &= 1 + F(\ln t - c_1) + F(\ln t - c_2) + \ldots F(\ln t - c_r) \\ &= 1 + f(t/d_1) + f(t/d_2) + \ldots f(t/d_r) \end{aligned}$$

We now show that $G(s) = \int_1^\infty f(t)dt/t^{s+1}$ satisfies the conditions of the Lemma; we can therefore apply it to find the the asymptotics of $f(t)$ and ultimately $F(x)$.

Set α and φ as in the theorem statement and define $\gamma = \ln \varphi$. Let $(\overline{c_1}, \overline{c_2}, \ldots, \overline{c_r})$ be any tuple such that $\overline{c_i} \leq c_i$ for all $1 \leq i \leq r$, and let $\overline{F}(x)$ be the "number of nodes" function associated with $(\overline{c_1}, \overline{c_2}, \ldots, \overline{c_r})$. The characterization of $F(x)$ given by equation (6) shows that $F(x) \leq \overline{F}(x)$.

Now suppose that $(\overline{c_1}, \overline{c_2}, \ldots, \overline{c_r})$ are rationally related and let $\overline{\varphi}$ be the reciprocal of the smallest positive root of $1 - z^{\overline{c_1}} - z^{\overline{c_2}} - \ldots - z^{\overline{c_r}}$. Then $F(x) \leq \overline{F}(x) = \theta\left(\overline{\varphi}^x\right)$. As $(\overline{c_1}, \overline{c_2}, \ldots, \overline{c_r})$ approaches closer and closer to $(c_1, \ldots, c_r)$ continuity implies that $\overline{\varphi} \to \varphi$, implying that $F(x) = O\left((\varphi + \epsilon)^x\right)$ for every $\epsilon > 0$. Thus, for every fixed $\epsilon' > 0$,

$$f(t) = F(\ln t) = O\left((\varphi + \epsilon)^{\ln t}\right) = O\left(t^{\gamma + \epsilon'}\right)$$

(where the constant in the $O()$ might depend upon ϵ'). This in turn proves that $G(s)$ converges uniformly and is analytic in in the halfplane $\{s | \Re(s) > \gamma + \epsilon'\}$. We can therefore solve for $G(s)$ in that halfplane as follows.

$$G(s) = \int_1^\infty f(t)dt/t^{s+1}$$

$$
\begin{aligned}
&= \int_1^\infty dt/t^{s+1} + \int_1^\infty f(t/d_1)dt/t^{s+1} + \ldots + \int_1^\infty f(t/d_r)dt/t^{s+1} \\
&= \frac{1}{s} + \left[d_1^{-s} + d_2^{-s} + \ldots + d_r^{-s}\right] G(s)
\end{aligned}
$$

from which we derive

$$
G(s) = \frac{1}{s(1 - d_1^{-s} + d_2^{-s} + \ldots + d_r^{-s})} \tag{20}
$$

This equation is valid in the halfplane $\{s|\Re(s) > \gamma\}$. Now notice that

$$
1 - d_1^{-s} + d_2^{-s} + \ldots + d_r^{-s} = 1 - e^{-c_1 s} - e^{-c_2 s} + \ldots + e^{-c_r s} = Q(e^{-s}).
$$

Therefore $\gamma = \ln\varphi = -\ln\alpha$ is a pole of $G(s)$ and from Lemma 9 all poles of $G(s)$ must be on or to the left of the line $\{s|\Re(s) = \gamma\}$. Furthermore, φ is the only pole on that line. This is because for $s = \varphi + iy$, $y \neq 0$ we have

$$
\begin{aligned}
1 - e^{-c_1 s} - \ldots - e^{-c_r s} &= 1 - \alpha^{c_1}\Re\left(e^{-c_1 iy}\right) - \ldots - \alpha^{c_r}\Re\left(e^{-c_r yi}\right) \\
&\geq 1 - \alpha^{c_1} - \ldots \alpha^{c_r} = 0
\end{aligned}
$$

with equality if and only if $\Re\left(e^{-c_j yi}\right) = 1$ for all j, i.e., there exist positive integers k_j such that $c_j y = 2\pi k_j$ or $c_j = 2\pi k_j / y$ contradicting the fact that $(c_1, \ldots, c_r)$ are irrationally related.

We can therefore analytically continue $G(s)$ over and to the left of the line $\{s|\Re(s) = \gamma\}$ and the analytic continuation has a first order pole at $s = \gamma$ but no other singularity on that line. The residue of $G(s)$ at $s = \gamma$ is

$$
\frac{1}{\gamma \sum_i \ln d^i \, d_i^{-\gamma}} = \frac{1}{\ln\varphi \sum_i c_i \varphi^{-c_i}}
$$

Applying Lemma 10 we find

$$
F(x) = f(e^x) = \frac{1}{\gamma \sum_i \varphi^{-c_i} c_i} e^{\gamma x} + o\left(e^{\gamma x}\right) = \frac{1}{\ln\varphi\, c}\varphi^x + o\left(\varphi^x\right)
$$

proving the theorem.

Note. We mention here that it is actually possible to use Mellin-transform techniques to analyze the rational case as well. The reason we do not do so is that in the rational case the line $\{s|\Re(s) = \gamma\}$ will contain an infinite number of poles, all of whose residues must be added together (yielding a Fourier series representation of the periodic function); the generating function technique yields a simpler representation of the answer.

□

6 The Cost of Optimal Trees

In this section we shall combine our knowledge of the combinatorial structure of optimal trees T_n (Theorem 1) with our analysis of $F(x)$ (Theorem 2) to derive the proof of Theorem 5 describing how the cost of Varn codes grow as n increases.

We divide the proof into three parts, each of which will have its own subsection. In the first part we derive some general lemmas, true for *all* choices of $(c_1, \ldots, c_r)$, describing the growth of costs of optimal trees. In the second part we specialize this lemma to the irrational $(c_1, \ldots, c_r)$. We conclude in the third part by specializing the lemmas to rational $(c_1, \ldots, c_r)$. The analyses of the rational and irrational parts, taken together, will prove the theorem.

As we will soon see, the analysis of the rational case will be much more technically complicated than that of the irrational one, The intuitive reason for the difference in difficulties is that our approach will be to calculate $C(T_n)$ by first finding j such that $a_j \le n < a_{j+1}$, calculating $C(T_{a_j})$ and $C(T_n) - C(T_{a_j})$, and then combining them to get $C(T_n)$. In the irrational case, as j increases, the optimal trees grow smoothly (this is the content of Lemma 13), i.e., the difference between successive depths in the tree tend to zero and the number of nodes per any *individual* level will be relatively small (since the nodes are distributed among many levels). This relative paucity of nodes on the bottom level will imply that $C(T_n) - C(T_{a_j}) = o(C(T(n)))$, and we will therefore be able to approximate $C(T_n)$ by $C(T_{a_j})$, which will be much easier to calculate. In the rational case the levels in the tree are equally spaced and, more importantly, the number of nodes on successive levels grow geometrically. This implies that the number of nodes on the bottom level will always be a constant fraction of the total nodes in the tree. This fact will require that our analysis of the contribution of nodes on the bottom level must be delicate since $C(T_n) - C(T_{a_j})$ could be a substantial fraction of $C(T_n)$. It is this which leads to much of the complications.

6.1 General Cost Lemmas

Recall that T_{a_j} is the optimal tree which has as internal nodes the set V_{m_j} consisting of all nodes in the infinite tree at depth l_j or above. In Section 3.1 we also defined

$$A_j = \{v \in LEAF(V_{m_j}) : depth(v) \le l_j + h\},$$

$$B_j = A_j \cup \{v \in LEAF(V_{m_j}) : l_j + h < depth(v) \le l_{j+1} + h\},$$

$a_j = |A_j|$, and $b_j = |B_j|$. A_j is the set of leaves in T_{a_j}. The highest nodes in A_j are the $m_{j+1} - m_j$ nodes at depth l_{j+1} in the tree. Since node labelling is consistent with the depth ordering the nodes on depth l_{j+1} are labelled

$$m_j + 1,\, m_j + 2,\, m_j + 3,\, \ldots,\, m_j + l_{j+1}.$$

Finally let

$$u_1 < u_2 < u_3 < \cdots < u_{b_j - a_j}$$

be the labels of the nodes in $B_j \setminus A_j$.

With these definitions we can now prove:

Lemma 11 *Given n let j be such that $a_j \le n < a_{j+1}$. There are three possible cases:*

- *If $a_j \le n \le b_j$ then*

$$C(T_n) = C(T_{a_j}) + \sum_{i=1}^{n-a_j} depth(u_i). \tag{21}$$

In particular

$$C(T_{b_n}) = C(T_{a_j}) + \sum_{i=1}^{b_j-a_j} depth(u_i). \tag{22}$$

- *If $n = b_j + p(k-1)$ then*

$$C(T_n) = C(T_{b_j}) + p(k-1)(l_{j+1} + h). \tag{23}$$

- *If* $n = b_j + p(k-1) + q$ *with* $0 < q < k$ *then*

$$C(T_n) = C(T_{b_j}) + (p(k-1)+q)(l_{j+1}+h) + O(1). \tag{24}$$

Proof. The first part follows directly from Theorem 1, part (2).

To prove the second part note that if $n = b_j + p(k-1)$ then from Theorem 1 (3), T_n is created by starting with T_{b_j}, taking p of its leaves from depth l_{j+1} and making each of them internal with k leaves. The change in cost resulting from making one such leaf internal is

$$-l_{j+1} + \sum_{i=1}^{k}(l_{j+1}+c_i) = (k-1)(l_{j+1}+h)$$

where we are using the definition $h = \frac{\sum_{i=1}^{k} c_i}{k-1}$. The proof of the second part follows.

To prove the third part we recall from the discussion following the statement of Theorem 1 that $T_{b_j+p(k-1)+q}$ is created either by starting with $T^{m_j+p}_{b_j+p(k-1)}$ and adding the q smallest unused leaves in V_{m_j+p} to the tree or starting with $T^{m_j+p+1}_{b_j+(p+1)(k-1)}$ and erasing the $k-1-q$ deepest leaves in that tree. In both cases the nodes added or subtracted must have depth at least l_{j+1} and at most $l_{j+1}+c_r$ (since all of their parents have depth most l_{j+1}). Thus, in both cases the third part follows from (23) $\square$

We now observe that $F(x)$ is a step function which jumps $(F(l_i) - F(l_{i-1}))$ at l_i. For $\delta > 0$, we may thus express $F(l_j+\delta) - F(l_j)$ by the Riemann-Stieltjes integral,

$$F(l_j+\delta) - F(l_j) = \int_{l_j}^{l_j+\delta} dF(x).$$

This notation will make our analysis somewhat easier.

We start by deriving an expression for $Cost(T_{a_j})$ that we will afterwards combine with Lemma 11 to yield an expression for general T_n.

Lemma 12

$$\begin{aligned}
a_j &= \int_{l_j}^{l_j+h} dF(x) - \sum_{i=1}^{k}\int_{l_j}^{l_j+h-c_i} dF(x) && (25)\\
b_j &= \int_{l_j}^{l_{j+1}+h} dF(x) - \sum_{i=1}^{k}\int_{l_j}^{l_{j+1}+h-c_i} dF(x) && (26)\\
C(T_{a_j}) &= (l_j+h)a_j - \int_{l_j}^{l_j+h} F(x)dx + \sum_{i=1}^{k}\int_{l_j}^{l_j+h-c_i} F(x)dx && (27)
\end{aligned}$$

Proof. By definition

$$\begin{aligned}
A_j &= \{v \in LEAF(V_{m_j}) : depth(v) \le l_j + h\}\\
&= \bigcup_{i=1}^{r}\{child_i(u) : u \in V_{m_j} \text{ and } l_j < depth(child_i(u)) \le l_j + h\}\\
&= \{v : l_j < depth(v) \le l_j + h\} \setminus \bigcup_{i=1}^{r}\{child_i(u) : l_j < depth(u) \le l_j + h - c_i\}\\
&= \{v : l_j < depth(v) \le l_j + h\} \setminus \bigcup_{i=1}^{k}\{child_i(u) : l_j < depth(u) \le l_j + h - c_i\}
\end{aligned}$$

Therefore,

$$\begin{aligned}
a_j &= |\{v : l_j < depth(v) \le l_j + h\}| - \sum_{i=1}^{k} |\{child_i(u) : l_j < depth(u) \le l_j + h - c_i\}| \\
&= |\{v : l_j < depth(v) \le l_j + h\}| - \sum_{i=1}^{k} |\{u : l_j < depth(u) \le l_j + h - c_i\}| \\
&= \int_{l_j}^{l_j+h} dF(x) - \sum_{i=1}^{k} \int_{l_j}^{l_j+h-c_i} dF(x)
\end{aligned}$$

Similarly,

$$b_j = \int_{l_j}^{l_{j+1}+h} dF(x) - \sum_{i=1}^{k} \int_{l_j}^{l_{j+1}+h-c_i} dF(x)$$

and

$$\begin{aligned}
C(T_{a_j}) &= \sum_{v \in A_j} depth(v) \\
&= \sum_{l_j < depth(v) \le l_j + h} depth(v) - \sum_{i=1}^{k} \sum_{l_j < depth(u) \le l_j + h - c_i} depth(child_i(u)) \\
&= \sum_{l_j < depth(v) \le l_j + h} depth(v) - \sum_{i=1}^{k} \sum_{l_j + c_i < depth(u) \le l_j + h} depth(u) \\
&= \int_{l_j}^{l_j+h} x dF(x) - \sum_{i=1}^{k} \int_{l_j}^{l_j+h-c_i} (x + c_i) dF(x).
\end{aligned}$$

Integrating the last equation by parts gives

$$\begin{aligned}
C(T_{a_j}) &= (l_j + h)F(l_j + h) - l_j F(l_j) - \sum_{i=1}^{k} (l_j + h) F(l_j + h - c_i) + \\
&\quad \sum_{i=1}^{k} (l_j + c_i) F(l_j) - \int_{l_j}^{l_j+h} F(x)dx + \sum_{i=1}^{k} \int_{l_j}^{l_j+h-c_i} F(x)dx
\end{aligned}$$

Therefore,

$$\begin{aligned}
C(T_{a_j}) &= (l_j + h) \left[F(l_j + h) - \sum_{i=1}^{k} F(l_j + h - c_i) \right] + (k-1)(l_j + h)F(l_j) + \\
&\quad \int_{l_j}^{l_j+h} F(x)dx + \sum_{i=1}^{k} \int_{l_j}^{l_j+h-c_i} F(x)dx \\
&= (l_j + h)a_j - \int_{l_j}^{l_j+h} F(x)dx + \sum_{i=1}^{k} \int_{l_j}^{l_j+h-c_i} F(x)dx. \qquad (28)
\end{aligned}$$

□

6.2 The Irrational Case

We now prove Theorem 5 under the assumption that $(c_1, \ldots, c_r)$ is irrationally related. Recall from Theorem 2 that in this case

$$F(x) = \frac{1}{c \ln \varphi} \varphi^x + o(\varphi^x). \qquad (29)$$

and from Lemma 3

$$l_{j+1} - l_j = o(1). \tag{30}$$

Now set $K = \frac{1}{c \ln \varphi}\left(\varphi^h - \sum_{i=1}^{k} \varphi^{h-c_i} + (k-1)\right)$. Then

Lemma 13 *If* $(c_1, \ldots, c_r)$ *is irrationally related and* j *is such that* $a_j \le n < a_{j+1}$ *:*

$$\begin{aligned}
a_j &= \varphi^{l_j} K + o\left(\varphi^{l_j}\right) && (31)\\
a_{j+1} - a_j &= o\left(\varphi^{l_j}\right) && (32)\\
l_j &= \log_\varphi n - \log_\varphi K + o(1) && (33)\\
n &= \Theta(a_j) = \Theta(a_{j+1}) && (34)\\
C(T_{a_j}) &= (l_j + h)a_j - \frac{a_j}{\ln \varphi} - o\left(\varphi^{l_j}\right) && (35)
\end{aligned}$$

Proof. From (29)

$$\begin{aligned}
a_j &= \int_{l_j}^{l_j+h} dF(x) - \sum_{i=1}^{k} \int_{l_j}^{l_j+h-c_i} dF(x)\\
&= \frac{1}{c \ln \varphi}\varphi^{l_j}\left(\varphi^h - 1 - \sum_{i=1}^{k}\left(\varphi^{h-c_i} - 1\right) + o\left(\varphi^{l_j}\right)\right)\\
&= \varphi^{l_j} K + o\left(\varphi^{l_j}\right)
\end{aligned}$$

Combining this with the fact that $l_{j+1} - l_j = o(1)$ yields

$$\begin{aligned}
a_{j+1} - a_j &= \varphi^{l_j} K\left(\varphi^{l_{j+1}-l_j} - 1\right) + o\left(\varphi^{l_j}\left(1 + \varphi^{l_{j+1}-l_j}\right)\right)\\
&= o\left(\varphi^{l_j}\right)
\end{aligned}$$

Since $a_j \le n < a_{j+1}$ we have that

$$\varphi^{l_j} K + o\left(\varphi^{l_j}\right) \le n < \varphi^{l_{j+1}} K + o\left(\varphi^{l_{j+1}}\right). \tag{36}$$

Taking $\log_\varphi$ and again using the fact that $l_{j+1} - l_j = o(1)$ yields

$$l_j = \log_\varphi n - \log_\varphi K + o(1).$$

From (31) and (32) we know that $\Theta(a_j) = \Theta(a_{j+1})$. Since $a_j \le n < a_{j+1}$ this implies that

$$n = \Theta(a_j) = \Theta(a_{j+1}).$$

Finally notice that

$$\begin{aligned}
\int_{l_j}^{l_j+h} F(x)dx - \sum_{i=1}^{k} \int_{l_j}^{l_j+h-c_i} F(x)dx &= \frac{1}{\ln \varphi}\varphi^{l_j}\left(\varphi^h - 1 - \sum_{i=1}^{k}\left(\varphi^{h-c_i} - 1\right) + o\left(\varphi^{l_j}\right)\right)\\
&= \frac{a_j}{\ln \varphi} + o\left(\varphi^{l_j}\right)
\end{aligned}$$

so

$$\begin{aligned}
C(T_{a_j}) &= (l_j + h)a_j - \int_{l_j}^{l_j+h} F(x)dx + \sum_{i=1}^{k} \int_{l_j}^{l_j+h-c_i} F(x)dx\\
&= (l_j + h)a_j - \frac{a_j}{\ln \varphi} - o\left(\varphi^{l_j}\right)
\end{aligned}$$

$\square$

The last lemma we will need before proving the result is

Lemma 14 *Let* $(c_1, \ldots, c_r)$ *be irrationally related and* j *be such that* $a_j \le n < a_{j+1}$. *Then*

$$C(T_n) = (l_j + h)n - \frac{a_j}{\ln \varphi} + o(n).$$

Proof. The proof proceeds by splitting into the three cases treated by Theorem 1.

- $\underline{a_j \le n \le b_j}$: In this case the optimal tree T_n is formed by starting with T_{a_j} and adding the $n - a_j$ new leaves $u_1, u_2, u_3, \ldots, u_{n-a_j}$. By definition, $l_j + h < depth(u_i) \le l_{j+1} + h$ so, from Lemma 11,

$$\begin{aligned} C(T_n) &= C(T_{a_j}) + (n - a_j)(l_j + h) + O(n - a_j)(l_{j+1} - l_j) \\ &= (l_j + h)a_j - \frac{a_j}{\ln \varphi} - o\left(\varphi^{l_j}\right) + (n - a_j)(l_j + h) + O(n - a_j)(l_{j+1} - l_j) \\ &= (l_j + h)n - \frac{a_j}{\ln \varphi} + o(n). \end{aligned}$$

  In particular

$$C(T_{b_j}) = (l_j + h)b_j - \frac{a_j}{\ln \varphi} + o(n).$$

- $\underline{n = b_j + p(k-1) < a_{j+1}}$: In this case, using the second part of Lemma 11:

$$\begin{aligned} C(T_n) &= C(T_{b_j}) + p(k-1)(l_{j+1} + h) \\ &= (l_j + h)b_j - \frac{a_j}{\ln \varphi} + o(n) + p(k-1)(l_{j+1} + h) \\ &= (l_j + h)n - \frac{a_j}{\ln \varphi} + o(n). \end{aligned}$$

- $\underline{n = b_j + p(k-1) + q < a_{j+1},\ q \neq 0.}$: From the third part of Lemma 11

$$\begin{aligned} C(T_n) &= C(T_{b_j}) + (p(k-1) + q)(l_{j+1} + h) + o(1) \\ &= (l_j + h)b_j - \frac{a_j}{\ln \varphi} + o(n) + (p(k-1) + 1)(l_{j+1} + h) \\ &= (l_j + h)n - \frac{a_j}{\ln \varphi} + o(n). \end{aligned}$$

□

We have just seen that

$$C(T_n) = (l_j + h)n - \frac{a_j}{\ln \varphi} + o(n).$$

To transform this into the form given in the statement of Theorem 5 we note from Lemma 13 that $l_j = \log_\varphi n - \log_\varphi K$. From the same lemma we also have that $a_{j+1} - a_j = o\left(\varphi^{l_j}\right)$ so if $a_j \le n < a_{j+1}$ then $n - a_j = o\left(\varphi^{l_j}\right) = o(n)$. Combining these facts proves that, for all n,

$$\boxed{T(n) = n \log_\varphi n + \left(h - \log_\varphi K - \frac{1}{\ln \varphi}\right) n + o(n).}$$

and we have completed the proof for the irrational case.

6.3 The Rational Case

We now analyze the rational case. We will assume that the c_i's are all positive integers with $gcd(c_1, \ldots, c_r) = 1$. At the end of this section we will quickly discuss what happens when this is not the case.

We note that, by Lemma 3, the assumption of $gcd(c_1, \ldots, c_r) = 1$ implies that $\exists J$ such that $\forall j > J$, $l_{j+1} - l_j = 1$. We start the analysis by using this fact to specialize Lemma 11 for the rational case:

Lemma 15 *For* $a_j \le n \le a_{j+1}$,

$$C(T_n) = C(T_{a_j}) + (n - a_j)(l_j + 1 + \lfloor h \rfloor) + \mathcal{X}(n - b_j)\{h\} + O(1)$$

where

$$\mathcal{X}(\theta) = \begin{cases} \theta, & \textit{if } \theta > 0; \\ 0, & \textit{otherwise.} \end{cases}$$

Proof. We assume that j is large enough so that $l_{j+1} = l_j + 1$. Otherwise, the $O(1)$ term in the expression will absorb the cost of the tree. There are three cases:

1. $a_j \le n \le b_j$:
   From Lemma 11, equation (21),
   $$C(T_n) = C(T_{a_j}) + \sum_{i=1}^{n-a_j} depth(u_i),$$
   where the u_i are in the set
   $$B_j \setminus A_j = \{v \in LEAF(V_{m_j}) : l_j + h < depth(v) \le l_{j+1} + h\}.$$
   Using the fact that $l_{j+1} = l_j + 1$ we have that $\forall i$, $depth(u_i) = l_j + 1 + \lfloor h \rfloor$. Thus
   $$C(T_n) = C(T_{a_j}) + (n - a_j)(l_j + 1 + \lfloor h \rfloor) \tag{37}$$
   In particular
   $$C(T_{b_j}) = C(T_{a_j}) + (b_j - a_j) * (l_j + 1 + \lfloor h \rfloor) \tag{38}$$

2. $b_j < n < a_{j+1}$ with $n = b_j + p(k-1) + q$, $0 \le q < (k-1)$:
   From Lemma 11, equations (23) and (24),
   $$C(T_n) = C(T_{b_j}) + (p(k-1) + q)(l_{j+1} + h) + O(1).$$
   Combining this with (38)
   $$\begin{aligned} C(T_n) &= C(T_{b_j}) + (p(k-1) + q)(l_{j+1} + h) + O(1) \\ &= C(T_{a_j}) + (b_j - a_j) * (l_j + 1 + \lfloor h \rfloor) + (p(k-1) + q)(l_{j+1} + h) + O(1) \\ &= C(T_{a_j}) + (n - a_j) * (l_j + 1 + \lfloor h \rfloor) + (n - b_j)\lfloor h \rfloor + O(1) \\ &= C(T_{a_j}) + (n - a_j) * (l_j + 1 + \lfloor h \rfloor) + \mathcal{X}(n - b_j)\{h\} + O(1) \end{aligned}$$

3. $n = a_{j+1}$:
   Recall from Theorem 1 that $T_{a_{j+1}}$ is obtained from T_{b_j} by making all leaves of depth l_{j+1} into internal nodes with k children. Since $h = \frac{1}{(k-1)} \sum_{i=1}^{k} c_i$, we have
   $$\begin{aligned} C(T_{a_{j+1}}) &= C(T_{b_j}) + \frac{a_{j+1} - b_j}{k-1} \left(\sum_{i=1}^{k} c_i + (k-1) l_{j+1} \right) \\ &= C(T_{a_j}) + (b_j - a_j)(l_j + 1 + \lfloor h \rfloor) + (a_{j+1} - b_j)(l_{j+1} + h) \\ &= C(T_{a_j}) + (a_{j+1} - a_j)(l_j + 1 + \lfloor h \rfloor) + (a_{j+1} - b_j)\{h\} \quad (39) \\ &= C(T_{a_j}) + (a_{j+1} - a_j)(l_j + 1 + \lfloor h \rfloor) + \mathcal{X}(a_{j+1} - b_j)\{h\} \quad (40) \end{aligned}$$

□

To successfully employ the previous lemma we will need better expressions for a_j, b_j, and $C(T_{a_j})$.

Lemma 16 *Let $j \geq 0$ be an integer. Then*

$$\begin{aligned}
a_j &= K\varphi^{l_j} + O(\rho^{l_j}) \\
b_j &= K\varphi^{l_j+R} + O(\rho^{l_j}) \\
C(T_{a_j}) &= (l_j + 1 + \lfloor h \rfloor)a_j - \frac{1}{\varphi - 1}K\varphi^{l_j+1} + \{h\}\frac{k-1}{c(1-\varphi^{-1})}\varphi^{l_j} + O(\rho^{l_j})
\end{aligned}$$

where $\rho < \varphi$ and K, R, and A are as defined in Theorem 5:

$$K = \frac{1}{c(1-\varphi^{-1})}(\varphi^{\lfloor h \rfloor} - \sum_{i=1}^{k} \varphi^{\lfloor h \rfloor - c_i} + (k-1)),$$

$$A = \frac{(k-1)}{c(1-\varphi^{-1})K}, \qquad R = \log_\varphi((1-A)\varphi + A).$$

Proof. Recall from Theorem 2 that

$$F(l_j) = \frac{1}{c(1-\varphi^{-1})}\varphi^{l_j} + O(\rho^{l_j}). \tag{41}$$

for some $\rho < \varphi$. Plugging this into (25) in Lemma 12 we find that

$$\begin{aligned}
a_j &= \int_{l_j}^{l_j+h} dF(x) - \sum_{i=1}^{k} \int_{l_j}^{l_j+h-c_i} dF(x) \\
&= F(l_j+h) - F(l_j) - \sum_{i=1}^{k} \Big(F(l_j+h-c_i) - F(l_j)\Big) \\
&= F(l_j+h) + (k-1)F(l_j) - \sum_{i=1}^{k} F(l_j+h-c_i) \\
&= K\varphi^{l_j} + O(\rho^{l_j}).
\end{aligned}$$

Again from Theorem 2

$$\begin{aligned}
b_j &= \int_{l_j}^{l_{j+1}+h} dF(x) - \sum_{i=1}^{k} \int_{l_j}^{l_{j+1}+h-c_i} dF(x) \\
&= F(l_j+1+h) - F(l_j) - \sum_{i=1}^{k} \Big(F(l_j+1+h-c_i) - F(l_j)\Big) \\
&= F(l_j+1+h) + (k-1)F(l_j) - \sum_{i=1}^{k} F(l_j+1+h-c_i) \\
&= K\varphi^{l_j+1} - \frac{1}{c(1-\varphi^{-1})}(\varphi^{j+1} - \varphi^j) + O(\rho^{l_j}) \\
&= K\varphi^{l_j}\left(\left(1 - \frac{(k-1)}{c(1-\varphi^{-1})K}\right)\varphi + \frac{c(k-1)}{c(1-\varphi^{-1})K}\right) + O(\rho^j) \\
&= K\varphi^{j+R} + O(\rho^j).
\end{aligned}$$

Finally, returning yet again to Theorem 2, recall that

$$C(T_{a_j}) = (l_j + h)a_j - \int_{l_j}^{l_j+h} F(x)dx + \sum_{i=1}^{k} \int_{l_j}^{l_j+h-c_i} F(x)dx \tag{42}$$

To proceed we need the fact that $F(x)$ only changes at integral values $x = l_j$. Thus $\forall x$, $F(x) = F(\lfloor x \rfloor)$ and, for any $\delta > 1$, we have

$$\int_{l_j}^{l_j+\delta} F(x)dx = \sum_{t=l_j}^{l_j+\lfloor\delta\rfloor-1} F(t) + \{\delta\}F(l_j+\delta).$$

Continuing yields that

$$\begin{aligned}
-\int_{l_j}^{l_j+h} F(x)dx + \sum_{i=1}^{k}\int_{l_j}^{l_j+h-c_i} F(x)dx &= -\sum_{t=l_j}^{l_j+\lfloor h\rfloor-1} F(t) + \sum_{i=1}^{k}\sum_{t=l_j}^{l_j+\lfloor h\rfloor-c_i-1} F(t) \\
&\quad -\{h\}\left[F(l_j+h) - \sum_{i=1}^{k} F(l_j+h-c_i)\right] \\
&= (1-\{h\})a_j - \sum_{t=l_j+1}^{l_j+\lfloor h\rfloor} F(t) \\
&\quad + \sum_{i=1}^{k}\sum_{t=l_j+1}^{l_j+\lfloor h\rfloor-c_i} F(t) + \{h\}(k-1)F(l_j)
\end{aligned}$$

Substituting this back into (42) gives

$$C(T_{a_j}) = (l_j+1+\lfloor h\rfloor)a_j - \sum_{t=l_j+1}^{l_j+\lfloor h\rfloor} F(t) + \sum_{i=1}^{k}\sum_{t=l_j+1}^{l_j+\lfloor h\rfloor-c_i} F(t) + \{h\}(k-1)F(l_j) \quad (43)$$

Now, using the expansion for $F(x)$ in (41),

$$\begin{aligned}
C(T_{a_j}) &= (l_j+1+\lfloor h\rfloor)a_j - \sum_{t=l_jj+1}^{l_j+\lfloor h\rfloor} F(t) + \sum_{i=1}^{k}\sum_{t=l_jj+1}^{l_j+\lfloor h\rfloor-c_i} F(t) + \{h\}(k-1)F(j) \\
&= (l_j+1+\lfloor h\rfloor)a_j - \frac{1}{c(1-\varphi^{-1})}\sum_{t=l_j+1}^{l_j+\lfloor h\rfloor}\varphi^t \\
&\quad + \frac{1}{c(1-\varphi^{-1})}\sum_{i=1}^{k}\sum_{t=l_j+1}^{l_j+\lfloor h\rfloor-c_i}\varphi^t + \{h\}\frac{k-1}{c(1-\varphi^{-1})}\varphi^{l_j} + O(\rho^{l_j}) \\
&= (l_j+1+\lfloor h\rfloor)a_j - \frac{1}{c(1-\varphi^{-1})}\frac{\varphi^{l_j+\lfloor h\rfloor+1} - \varphi^{l_j+1}}{\varphi-1} \\
&\quad + \frac{1}{c(1-\varphi^{-1})}\sum_{i=1}^{k}\frac{\varphi^{l_j+\lfloor h\rfloor-c_i+1} - \varphi^{l_j+1}}{\varphi-1} \\
&\quad + \{h\}\frac{k-1}{c(1-\varphi^{-1})}\varphi^{l_j} + O(\rho^{l_j}) \\
&= (l_j+1+\lfloor h\rfloor)a_j - \frac{1}{\varphi-1}K\varphi^{l_j+1} + \{h\}\frac{k-1}{c(1-\varphi^{-1})}\varphi^{l_j} + O(\rho^{l_j})
\end{aligned}$$

proving the lemma. □

The last piece we will need is a a crude bound on the difference in costs between trees. Essentially it says that adding a node at the l_jth level will contribute a cost of $O(l_j)$.

Lemma 17 *Let j be such that $a_j < n_2 \leq a_{j+1}$ and suppose $n_1 < n_2$. Then*

$$C(T_{n_2}) - C(T_{n_1}) = O(l_j(n_2 - n_1)).$$

Proof. We first assume that $n_1 = n$, $n_2 = n+1$. This implies that $a_j \leq n < n+1 \leq a_{j+1}$ so, by Lemma 15, we have

$$C(T_{n+1}) - C(T_n) = (j + 1 + \lfloor h \rfloor) + \mathcal{X}(n+1-b_j) - \mathcal{X}(n-b_j) + O(1)$$

Since $\mathcal{X}(n+1-b_j) - \mathcal{X}(n-b_j) \leq 1$,

$$C(T_{n+1}) - C(T_n) = O(l_j).$$

Therefore, for $a_j < n_2 \leq a_{j+1}$ and $n_1 < n_2$,

$$\begin{aligned} C(T_{n_2}) - C(T_{n_1}) &= \sum_{n=n_1}^{n_2-1} (C(T_{n+1}) - C(T_n)) \\ &= (n_2 - n_1) O(l_j) \\ &= O(l_j(n_2 - n_1)) \end{aligned}$$

□

Combining the previous facts will permit us to prove the the correctness of the theorem for rationally related $(c_1, \ldots, c_r)$ such that the c_i's are all positive integers with $gcd(c_1, \ldots, c_r) = 1$. To do this recall that, from Lemma 16, we know that $a_j = K\varphi^{l_j} + O(\rho^j)$. Set

$$a'_j = K\varphi^{l_j}. \tag{44}$$

Then $|a_j - a'_j| = O(\rho^j)$. Now suppose $a'_j \leq n < a'_{j+1}$. Then

$$l_j = \log_\varphi \frac{a'_j}{K} = \left\lfloor \log_\varphi \frac{n}{K} \right\rfloor, \qquad \varphi^{l_j} = \frac{n}{K} \varphi^{-\{\log_\varphi \frac{n}{K}\}}. \tag{45}$$

Again, from Lemma 16, recall that $b_j = K\varphi^{l_j + R} + O(\rho^j)$. Set

$$b'_j = K\varphi^{l_j + R}. \tag{46}$$

Then

$$b'_j = n\varphi^{R - \{\log_\varphi \frac{n}{K}\}}, \quad b_j - b'_j = O(\rho^j). \tag{47}$$

We are assuming that $a'_j \leq n < a'_{j+1}$. To prove the theorem we will need to treat the three cases

$$a_j \leq n \leq a_{j+1}, \qquad n < a_j, \qquad a_{j+1} < n,$$

separately.

1. $a_j \leq n \leq a_{j+1}$: From Lemma 15 we have that

$$\begin{aligned} C(T_n) &= C(T_{a_j}) + (n - a_j)(l_j + 1 + \lfloor h \rfloor) + \mathcal{X}(n - b_j)\{h\} + O(1) \\ &= C(T_{a_j}) + (n - a'_j)(l_j + 1 + \lfloor h \rfloor) + \mathcal{X}(n - b'_j)\{h\} + o(\varphi^{l_j}) \end{aligned}$$

where we are using the fact that

$$n - b_j = n - b'_j + (b_j - b'_j) = n - b'_j + O(\rho^{l_j}).$$

2. $n < a_j$: In this case $a'_j \le n < a_j$, implying that $n - a'_j < a'_j - a_j = O(\rho^{l_j})$. Thus, using Lemma 17,

$$\begin{aligned}
C(T_n) &= C(T_{a_j}) + \left(C(T_n) - C(T_{a_j})\right) \\
&= C(T_{a_j}) + O(l_j(n - a'_j)) \\
&= C(T_{a_j}) + O(l_j \rho^{l_j}) \\
&= C(T_{a'_j}) + (n - a_j)(l_j + 1 + \lfloor h \rfloor) + \mathcal{X}(n - b'_j)\{h\} + o(\varphi^{l_j}).
\end{aligned}$$

3. $a_{j+1} < n$: In this case $a_{j+1} \le n < a'_{j+1}$, implying that $n - a_{j+1} < a'_{j+1} - a_{j+1} = O(\rho^{l_j})$. We attack this in two steps. The first one is quite similar to the previous case:

$$\begin{aligned}
C(T_n) &= C(T_{a_{j+1}}) + C(T_n) - C(T_{a_{j+1}}) \\
&= C(T_{a_{j+1}}) + O(l_{j+1}(n - a_{j+1})) \\
&= C(T_{a_{j+1}}) + O(l_j \rho^{l_j}).
\end{aligned}$$

Now note that, from Lemma 15,

$$C(T_{a_{j+1}}) = C(T_{a_j}) + (a_{j+1} - a_j)(l_j + 1 + \lfloor h \rfloor) + \mathcal{X}(a_{j+1} - b_j)\{h\} + O(1)$$

Since

$$a_{j+1} - a_j = n - a_j + (a_{j+1} - n) = n - a_j + o(\varphi^{l_j})$$

and

$$a_{j+1} - b_j = n - b'_j + (a_{j+1} - n) + (b_j - b'_j) = n - b'_j + o(\varphi^{l_j})$$

We have

$$C(T_n) = C(T_{a_j}) + (n - a_j)(l_j + 1 + \lfloor h \rfloor) + \mathcal{X}(n - b'_j)\{h\} + o(\varphi^{l_j}).$$

Combining parts 1, 2 and 3 above we find that we have proven, *for all* n, satisfying $a'_j \le n < a'_{j+1}$, that

$$C(T_n) = C(T_{a_j}) + (n - a_j)(l_j + 1 + \lfloor h \rfloor) + \mathcal{X}(n - b'_j)\{h\} + o(\varphi^{l_j}).$$

To complete the Theorem we substitute the values for $C(T_{a_j})$, a_j and b_j found in Lemma 16 and use (17) to find

$$\begin{aligned}
C(T_n) &= (l_j + 1 + \lfloor h \rfloor)a_j - \frac{1}{\varphi - 1} K \varphi^{l_j + 1} + \{h\} \frac{k - 1}{c(1 - \varphi^{-1})} \varphi^{l_j} \\
&\quad + (n - a_j)(l_j + 1 + \lfloor h \rfloor) + \mathcal{X}(n - b'_j)\{h\} + o(\varphi^{l_j}) \\
&= (l_j + 1 + \lfloor h \rfloor)n - \frac{1}{\varphi - 1} \varphi^{1 - \{\log_\varphi \frac{n}{K}\}} n \\
&\quad + \{h\} A K \varphi^{l_j} + \{h\} \mathcal{X}(n - b'_j) + o(\varphi^{l_j}).
\end{aligned}$$

To simplify this equation note that $\mathcal{X}(\theta) = \mathcal{X}(-\theta) + \theta$ and that $\mathcal{X}(n\theta) = n\mathcal{X}(\theta)$. Thus

$$\{h\}\mathcal{X}(n - b'_j) = \{h\} \left(n - b'_j + n\mathcal{X}\left(1 - \frac{b'_j}{n}\right)\right) = \{h\} \left(n - b'_j + n\mathcal{X}\left(1 - \varphi^{R - \{\log_\varphi \frac{n}{K}\}}\right)\right).$$

Also

$$\begin{aligned}
AK\varphi^{l_j} - b'_j &= AK\varphi^{l_j} - K\varphi^{l_j + R} \\
&= K\varphi^{l_j}\left(A - \varphi^R\right) \\
&= K\varphi^{l_j}\left[A - ((1 - A)\varphi + A)\right] \\
&= K\varphi^{l_j + 1}[1 - A].
\end{aligned}$$

Finally

$$
\begin{aligned}
l_j = \quad &= \left\lfloor \log_\varphi \frac{n}{K} \right\rfloor \\
&= \log_\varphi n - \log_\varphi K - \left\{ \log_\varphi \frac{n}{K} \right\}.
\end{aligned}
$$

Combining everything we find that

$$
\begin{aligned}
C(T_n) &= n \log_\varphi n + \left[h + 1 - \log_\varphi K - \left\{ \log_\varphi \frac{n}{K} \right\} - \frac{1}{\varphi - 1} \varphi^{1 - \{\log_\varphi \frac{n}{K}\}} \right. \\
&\qquad \left. + \{h\} K \varphi^{l_j + 1} [1 - A] + + \{h\} \mathcal{X} \left(1 - \varphi^{R - \{\log_\varphi \frac{n}{K}\}} \right) n \right. \\
&= n \log_\varphi n + \left[h + 1 - \log_\varphi K - \left\{ \log_\varphi \frac{n}{K} \right\} - \frac{1}{\varphi - 1} \varphi^{1 - \{\log_\varphi \frac{n}{K}\}} \right] \\
&\qquad + \{h\} (1 - A) \varphi^{1 - \{\log_\varphi \frac{n}{K}\}} + \{h\} \mathcal{X} \left(1 - \varphi^{R - \{\log_\varphi \frac{n}{K}\}} \right) n
\end{aligned}
$$

Thus

$$
\boxed{C(T_n) = n \log_\varphi n + B\left(\{\log_\varphi \frac{n}{K}\} \right) n + D\left(\{\log_\varphi \frac{n}{K}\} \right) n + o(n)}
$$

where $B(\theta)$ and $D(\theta)$ are periodic functions with period 1 as defined in the theorem statement.

Recall that we have all along been assuming that $gcd(c_1, \ldots, c_r) = 1$. We now quickly discuss what happens if this is not the case. Suppose that $(\overline{c_1}, \ldots, \overline{c_r}) = d(c_1, \ldots, c_r)$ where $gcd(c_1, \ldots, c_r) = 1$, and $d \neq 1$. Then for every tree T with n leaves for $(c_1, \ldots, c_r)$ there is a corresponding tree $\overline{T}$ for $(\overline{c_1}, \ldots, \overline{c_r})$ with $dC(T) = C(\overline{T})$ and vice-versa. The correspondence is the natural one that maps i^{th} edges to i^{th} edges. In particular, if T_n and $\overline{T}_n$ are respective optimal trees then $C(\overline{T}_n) = d \cdot C(T_n)$. This is the statement of the theorem so we are done.

Acknowledgments: The authors would like to thank Dr. Jacob Ecco for introducing them to the Morse Code puzzle which sparked this investigation. They would also like to thank Assaf Schuster for pointing them towards the Postal-Model, Rudolf Fleischer for conversations concerning the construction algorithm, and Julia Abrahams for pointers to references. The work of the authors was partially supported by HK RGC Grants HKUST 181/93E, HKUST652/95E, 6082/97E and 6137/98E

References

[1] `Doris Altenkamp and Kurt Mehlhorn`, "Codes: Unequal Probabilities, Unequal Letter Costs," *Journal of the Association for Computing Machinery*, **27**(3) (July 1980) 412-427.

[2] `A. Bar-Noy and S. Kipnis`, "Designing Broadcasting Algorithms in the Postal Model for Message-Passing Systems," *Mathematical Systems Theory,* **27**(5) (1994), pp.431–452.

[3] `Phil Bradford, Mordecai J. Golin, Lawrence L. Larmore, and Wojciech Rytter`, "Optimal Prefix-Free Codes for Unequal Letter Costs and Dynamic Programming with the Monge Property," Proceedings of the *Sixth European Symposium on Algorithms (ESA '98),* (1998),

[4] `V. S.-N. Choi`, Lopsided Trees: Analyses and Algorithms. M.Phil. Thesis, Hong Kong University of Science and Technology. Department of Computer Science. Technical Report HKUST-CS95-36. (1995).

[5] D.M. Choy and C.K. Wong, "Construction of Optimal Alpha-Beta Leaf Trees with Applications to Prefix Codes and Information Retrieval," *SIAM Journal on Computing,* **12**(3) (August 1983) pp. 426-446.

[6] I. Csiszár, "Simple Proofs of Some Theorems on Noiseless Channels," *Inform. Contr.,* bf 14 (1969) pp. 285-298

[7] I. Csiszár, G. Katona and G. Tsunády, "Information Sources with Different Cost Scales and the Principle of Conservation of Energy," *Z. Wahrscheinlichkeitstheorie verw,* **12**, (1969) pp. 185-222

[8] N. Cot, "A linear-time ordering procedure with applications to variable length encoding," *Proceedings of the 8th Annual Princeton Conference on Information Sciences and Systems,* (1974) 460-463.

[9] N. Cot, "Complexity of the Variable-length Encoding Problem," *Proceedings of the 6th Southeast Conference on Combinatorics, Graph Theory and Computing,* (1975) 211-224.

[10] Thomas M. Cover and Joy A. Thomas, *Elements of Information Theory,* John Wiley and Sons, (1991).

[11] Philippe Flajolet and Robert Sedgewick, *An Introduction to the Analysis of Algorithms,* Addison-Wesley, (1996).

[12] M. L. Fredman and D. E. Knuth, "Recurrence Relations Based on Minimization," *Journal of Mathematical Analysis and Applications,* **48** (1974) 534-559.

[13] M. Golin and G. Rote, "A Dynamic Programming Algorithm for Constructing Optimal Prefix-Free Codes for Unequal Letter Costs," *IEEE Transactions on Information Theory,* **44**(5) (September 1998) 1770-1781.

[14] M. Golin and Assaf Schuster "Optimal Point-to-Point Broadcast Algorithms via Lopsided Trees," *Discrete Applied Mathematics,* **93** (1999) 233-263.

[15] M. Golin and N. Young, "Prefix Codes: Equiprobable Words, Unequal Letter Costs," *SIAM Journal on Computing,* **25**(6) (December 1996) 1281-1292.

[16] Sanjiv Kapoor and Edward Reingold, "Optimum Lopsided Binary Trees," *Journal of the Association for Computing Machinery,* **36** (3) (July 1989) 573-590.

[17] R. Karp, "Minimum-Redundancy Coding for the Discrete Noiseless Channel," *IRE Transactions Information Theory,* **7** (1961) 27-39.

[18] Donald E. Knuth, *The Art of Computer Programming: Vol I (2nd ed.) Fundamental Algorithms,* Addison-Wesley, Reading, Mass.. (1973).

[19] Donald E. Knuth, *The Art of Computer Programming: Vol III, Sorting and Searching,* Addison-Wesley, Reading, Mass.. (1973).

[20] Abraham Lempel, Shimon Even, and Martin Cohen, "An Algorithm for Optimal Prefix Parsing of a Noiseless and Memoryless Channel," *IEEE Transactions on Information Theory,* **IT-19**(2) (March 1973) 208-214.

[21] Ivan Niven, "Irrational Numbers," *Carus Mathematical Monographs,* **11**. The Mathematical Association of America, Washington D.C., 1956.

[22] R. M. Krause, "Channels Which Transmit Letters of Unequal Duration," *Inform. Contr.,* **5** (1962) pp. 13-24,

[23] Y. Perl, M. R. Garey, and S. Even. "Efficient Generation of Optimal Prefix Code: Equiprobable Words Using Unequal Cost Letters," *Journal of the Association for Computing Machinery*, 22(2):202–214, April 1975.

[24] Nicholas Pippenger, "An Elementary Approach to Some Analytic Asymptotics," *SIAM Journal of Mathematical Analysis,* **24**(5) (September 1993) 1361-1377.

[25] Serap A. Savari, "Some Notes on Varn Coding," *IEEE Transactions on Information Theory,* **40**(1) (Jan. 1994) pp. 181-186.

[26] Robert Sedgewick, *Algorithms, 2nd edition,* Addison-Wesley, (1988).

[27] C.E. Shannon "A Mathematical Theory of Communication," *Bell System Technical Journal* **27** (1948) 379-423, 623-656.

[28] L. E. Stanfel, "Tree Structures for Optimal Searching," *Journal of the Association for Computing Machinery,* **17**(3) (July 1970) 508-517.

[29] B.F. Varn "Optimal Variable Length Codes (Arbitrary Symbol Costs and Equal Code Word Probabilities)," *Informat. Contr.,* **19** (1971) pp. 289-301