HKUST Theoretical Computer Science Center Research Report HKUST-TCSC-2003-11

Algorithms for Infinite Huffman-Codes*
(Extended Abstract)

Mordecai J. Golin
Dept. of Computer Science
Hong Kong U.S.T.
golin@cs.ust.hk

Kin Keung Ma
Dept. of Computer Science
Hong Kong U.S.T.
kkma@cs.ust.hk

Abstract

Optimal (minimum cost) binary prefix-free codes for infinite sources with geometrically distributed frequencies, e.g., $\mathcal{P} = \{p^i(1-p)\}_{i=0}^{\infty}$, $0 < p < 1$, were first (implicitly) suggested by Golomb over thirty years ago in the context of run-length encodings. Ten years later Gallager and Van Voorhis exhibited such optimal codes for *all* values of p. Just recently Merhav, Seroussi and Weinberger extended this further to find optimal binary prefix-free codes for *two-sided* geometric distributions.

These codes were derived by cleverly "guessing" optimal codes for *finite* sources, validating these guesses by using the sibling property of Huffman encoding, and then showing that the finite codes converge in a very specific sense to an optimal infinite one.

In this paper we describe the first *algorithmic* approach to constructing optimal prefix-free infinite codes. Our approach is to define an infinite weighted graph with the property that the least cost infinite path in the graph corresponds to the optimal code. We then show that even though the graph is infinite, the least-cost infinite path has a repetitive structure and that it is therefore possible to not only find this path but to find it relatively efficiently.

This approach will work for even more complicated generalizations of geometric sources where solutions can't be guessed as well as in extensions of Huffman-coding for which the Huffman algorithm no longer works, e.g., non-uniform cost encoding alphabet characters and/or other restrictions on the codewords. We illustrate our approach by deriving an algorithm for constructing optimal prefix free codes with a geometric source for the telegraph channel. We also implement our algorithm and show what the constructed codes look like in this case.

*This work was supported by Hong Kong CERG grants HKUST6162/00E, HKUST6082/01E and HKUST6206/02E

1 Introduction

In this paper we present the first *algorithmic* approach to constructing minimal-cost prefix codes for (generalized) geometric distributions.

Due to space considerations we do not develop the approach in its full generality. Instead, we illustrate the technique by developing an algorithm that works for one variant of Huffman coding and then sketch how this can be modified to work for many other variants. Some proofs are also omitted.

1.1 Background Consider a distribution[1]

$$\mathcal{P} = \{p_i\}_{i=0}^{n-1}, \quad p_0 \geq p_1 \geq p_2 \geq \cdots \geq p_{n-1}$$

on a set of n letters. The *Huffman Encoding problem* is to associate a prefix-free set of n binary words $\{w_i\}_{i=0}^{n} \subset \{0,1\}^*$ with $\mathcal{P}$ such that the expected word length $\sum_{i=0}^{n} p_i \cdot \text{length}(w_i)$ is minimized, where $\text{length}(w_i)$ is the number of bits in w_i, e.g., $\text{length}(0110) = 4$. A *prefix-free* set is one in which $\forall i \neq j$, w_i is not a prefix of w_j.

It is well known that finding such a code is equivalent to finding a tree with n leaves such that, when l_i is the length of the i-th highest leaf in the tree, then the expected external path length $\sum_{i=0}^{n} p_i l_i$ (achieved by placing the i-th letter (p_i) at the i-th highest leaf) is minimized. Such a tree may easily be found using the well-known *Huffman Encoding Algorithm* [13].

Suppose now that the situation is modified slightly to permit *infinite* sources, i.e.,

$$\mathcal{P} = \{p_i\}_{i=0}^{\infty}, \quad p_0 \geq p_1 \geq p_2 \geq \cdots.$$

In this case the problem of finding a prefix-free code, or equivalently, an infinite tree labelled with the p_i, with minimum weighted external path length, is not

[1] For this paper a *distribution* is a sequence such that, $\forall i, p_i > 0$ and $\sum_i p_i < \infty$. We do not require $\sum_i p_i = 1$.

nearly as well understood. It was proven[2] in [17] that optimal trees (codes) *exist* if and only if the entropy $-\sum_i p_i \log p_i$, of $\mathcal{P}$ is bounded but there is no algorithm for *constructing* optimal codes that works for all such $\mathcal{P}$ with bounded entropy. The reason that the basic Huffman algorithm can not be modified to work in such a case is that the Huffman algorithm starts by identifying the two smallest probabilities in the distribution and merging them; in the infinite source case, there is no smallest probability.

Certain restricted cases of the infinite source problem are better understood, though. The best known and earliest such case studied is that of the infinite binary codes (e.g., using only 0-s and 1-s) for the infinite geometric source. This is the source that fixes some p, $0 < p < 1$, and then defines $\mathcal{P}_p = \{(1-p)p^i\}_{i=0}^{\infty}$. As was noted by Golomb [11], such a source arises, for example, in the description of run-length encoding. Suppose we have a string of **A**s and **B**s in which each character occurs independently of every other one; **B**s occurring with probability p and **A**s with probability $1-p$. Now, for $i = 0, 1, 2, \ldots$ set $X_i = \underbrace{BB\ldots BB}_{i \text{ times}}A$. *Every* infinite string can be written uniquely as the concatenation of different X_is with the probability of X_i occurring being $(1-p)p^i$. We thus have a situation in which strings are composed of words from an infinite source with given distribution $\mathcal{P}_p$. Other problems that can be recast as finding a min-cost infinite tree with distribution $\mathcal{P}_p$ arise in operations research [12] and group testing [14] [19].

This special case of $\mathcal{P} = \mathcal{P}_p$ was studied by Gallager and Van Voorhis [6] who exhibited an optimal tree for every p. Their technique was to first define a countable sequence of *finite* sources $\mathcal{P}_p^0, \mathcal{P}_p^1, \mathcal{P}_p^2, \mathcal{P}_p^3, \ldots$ that were better and better approximations to $\mathcal{P}_p$. They then "guessed" the structure of the optimum Huffman code for these sources, verified the correctness of their guess by using the "sibling" property[3] of Huffman trees, and then showed that these codes "converge" to an infinite tree that is optimal for the infinite source. Their result can be stated as:

THEOREM 1. *(Gallager and Van Voorhis [6]) Given p, let m be the unique integer that satisfies*

$$p^m + p^{m+1} \leq 1 < p^m + p^{m-1}.$$

Let a tree T be described as a sequence I_i, $i = 0, 1, 2, 3, \ldots$ where I_i is the number of internal nodes on level i. Then the tree described by

$$I_0, I_1, I_2, I_3, \ldots = 1, 2, 4, \ldots, 2^{\lfloor \log_2 m \rfloor}, m, m, m, \ldots \tag{1.1}$$

is optimal for $\mathcal{P}_p$.

If we use M_m to denote the mth such tree then Figure 1 contains the tops of M_1, M_2, M_3, M_4 and M_5. For later reference we point out that if q_m is the unique real root of $1 - p^m - p^{m-1} = 0$ in $(0, 1)$ then $\{q_m\}_{m=1}^{\infty}$ monotonically increases and converges to 1. Another way of viewing the theorem is that the intervals $(q_m, q_{m+1}]_{m=1}^{\infty}$ partition $(0, 1)$ into a countable set of intervals and that $\forall p \in (q_{m-1}, q_m)$, the optimal tree for $\mathcal{P}_p$ is M_m.

These optimal codes (trees) were later shown to be the unique optimal codes for the given sources in [8]. Gallager and Van Voorhis' basic technique was later expanded upon and extended by [1] and [16] who showed that it could be applied to other families of infinite sources to show the existence of the optimal trees. [1] also showed how to extend the theorem to cover the ternary case in which every parent can have three children rather than two. This work was recently pushed even further by [18] who constructed optimal codes for two-sided geometric distributions (which have become useful in lossless image compression schemes). These are distributions $\mathcal{P}_{p,d}$ consisting of the infinite sequence

$$p_{x,d} = p^{|x+d|}, \quad x = 0, \pm 1, \pm 2, \pm 3, \ldots$$

where $0 < p < 1$ and $d \in \Re$ is fixed[4]. They showed that the real (p, d) plane can be subdivided into regions such that, in each region, all of the $\mathcal{P}_{p,d}$ have the same optimal tree. A comprehensive survey of the latest results may be found in [2].

All of these results (except for [8] which was proving uniqueness) share the same basic approach in that they construct an optimal code/tree for the *infinite* source by (i) constructing a sequence of finite sources that are better and better approximations to the infinite source P, (ii) "guessing" the structure of the optimum Huffman code for these finite sources (iii) verifying

[2] since the set of codes is infinite, existence of an optimal code is not a-priori obvious.

[3] Let T be a binary tree with n leaves labelled with the n probabilities, $p_1, \ldots, p_n$. The weight of a leaf will be its assigned probability; the weight of an internal node is recursively defined to be the sum of the weights of its children. T has the *sibling property* if the weights of its nodes, read left to right, top to bottom, are nonincreasing. The essential observation is that a tree with the sibling property can be produced by the Huffman algorithm and therefore represents an optimal code. The subtle point is that the sibling property does not *construct* optimal codes; it *verifies* that guessed codes are optimal.

[4] Note that the p_x are not necessarily sorted in decreasing order.

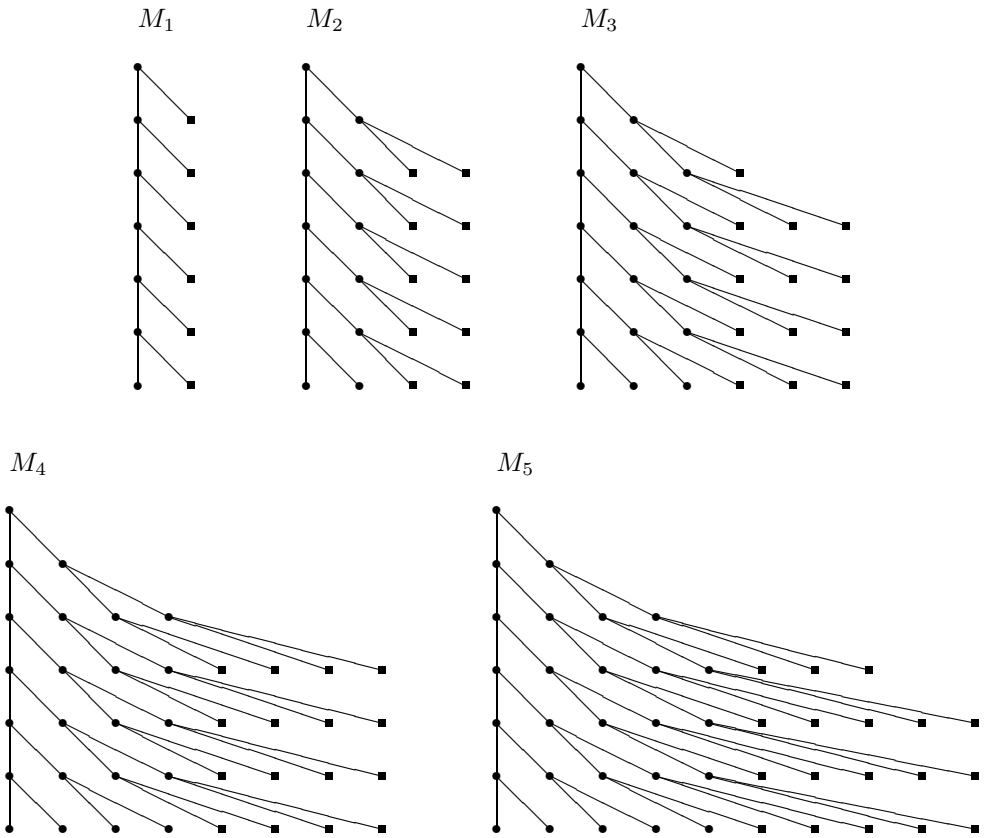


Figure 1: The tops of the infinite "optimal" trees M_1, M_2, M_3, M_4 and M_5. In this figure a filled-in circle represents an internal node while a square represents a leaf.

the correctness of the guess by using the "sibling" property of Huffman trees and (iv) showing that the finite codes/trees "converge" to an infinite tree that must be optimal for the infinite source.

1.2 Our new results The advantage of the approach used in the previous papers is that, by guessing the code structures, it at one stroke reports the optimal prefix-free codes for *all* of the distributions in the class.

A major disadvantage is that the approach only works if it is possible to *validate* the guessed code-structure. There are many variants of prefix-free coding for which the Huffman algorithm does not work, e.g., when the encoding alphabet has unequal letter costs[15, 10] or has restrictions on the codewords, e.g., the one ended codes of [3] or the restricted-zero codes of [5] ([2] has a nice survey of other variations); the Huffman algorithm does not work for any of these variations so it is impossible to verify guessed trees and the technique above can not be employed.

Another disadvantage is that, even for normal prefix-codes, it can be quite difficult to guess and verify the optimal codes for complicated sources. The two-sided geometric distributions of [18] seem to be as far as the technique can be pushed.

In this paper we develop the first *algorithmic* technique for constructing optimal prefix codes. It will permit unequal cost encoding alphabets that 'charge' more to transmit a '0' than to transmit a '1' and will also permit placing restrictions on the structures of the codewords, e.g., all codewords must end with a '1'; the technique will actually permit any restriction of the type, "all codewords must belong to given regular language $\mathcal{L}$".

The technique will also allow many generalizations of geometric distributions that use finite memory. For example, it will allow distributions of the type

$$\begin{array}{l} 1, p^{\alpha_1}, p^{\alpha_2}, \ldots, p^{\alpha_i}, \\ \quad p, p^{1+\alpha_1}, p^{1+\alpha_2}, \ldots, p^{1+\alpha_i}, \\ \quad p^2, p^{2+\alpha_1}, p^{2+\alpha_2}, \ldots, p^{2+\alpha_i}, \\ \quad p^3, p^{3+\alpha_1}, p^{3+\alpha_2}, \ldots, p^{3+\alpha_i}, \ldots \end{array} \tag{1.2}$$

where $0 \leq \alpha_1 \leq \alpha_2 \leq \cdots \leq \alpha_i < 1$, a natural generalization of the two sided distribution of [18].

The input to the algorithm will be the costs of the encoding letters, the restriction $\mathcal{L}$ on the codewords and

the distribution. The output will be a description of the optimal infinite coding tree.

There are two major ideas behind our algorithm. The first is the fact, developed for finite-source unequal-cost coding in [10] and later modified for restricted coding in [4], that prefix-code trees can be represented by paths in a weighted graph; a minimal-cost tree corresponds to a min-cost path. In our problem this means that we are looking for a *minimum cost infinite-link path in some infinite weighted graph.*

The second idea generalizes the observation [8] that optimum Huffman trees for the geometric source must be cyclic. That is, the structure of the optimum tree must be some finite head, followed by a finite part that infinitely cycles; Figure 1 illustrates the tops of the first 5 optimal infinite trees.

We can translate this into our graph description to show that a *minimum cost infinite-link path* must have a particular finite cyclic structure, i.e., a "head" followed by an infinitely repeated cycle, and then modify standard shortest path algorithms to look for a finite path that generates a minimum cost infinite cycle.

2 $(1,2)$-lopsided trees for geometric distributions.

As mentioned at the beginning of this abstract, due to space considerations we do not develop the approach in its full generality. Instead, we restrict ourselves to developing an algorithm that works for the special case of $(1,2)$ lopsided trees for geometric distributions. This example illustrates all of the basic ideas of the general technique.

In the general unequal-cost or *lopsided tree* problem, we are given weights α, β such that length$(0) = \alpha$ and length$(1) = \beta$. The length of a word w is the sum of the lengths of its characters. For example, if length$(0) = 1$ and length$(1) = 2$ (this is sometimes known as the *telegraph channel* [7]) then length$(1000) = 5$. Given α, β the (α, β)-*lopsided tree* problem is to find a prefix-free set of codewords w_i such that $\sum_i p_i \text{length}(w_i)$ is minimized. Equivalently, by setting the length of a left edge to be α and that of a right edge to be β (Figure 5) the problem is equivalent to finding a tree such that when ℓ_i is the length of the i-th highest leaf in the tree then the expected external path length, $\sum_i p_i \ell_i$ of the tree is minimized. For finite source distributions, even though the Huffman algorithm does not work, there are algorithms for solving this problem exactly e.g., [15, 10] and approximately, e.g., [9]; it is still unknown though, whether the problem is NP-Hard, polynomial time solvable, or lies somewhere else.

For the geometric source, there is no known algorithm for finding the tree. The previously developed techniques can not be applied because they require using the Huffman algorithm to verify guessed codes and the Huffman algorithm does not work for lopsided trees. In what follows we describe an algorithmic technique for finding the optimal $(1,2)$-lopsided tree for a given geometric distribution $\mathcal{P}_p$. The technique can easily be generalized to work for any (α, β)-lopsided tree when α/β is a rational number.

We first derive theorems describing the structure of the minimum cost $(1,2)$-lopsided trees for the geometric distribution $\mathcal{P}_p$. We then use these theorems to develop an algorithm which, given fixed p, will output the corresponding optimal tree for $\mathcal{P}_p$. Finally, the section concludes with the output of our program for various values of p.

2.1 Notation for describing optimal trees

In this section we generalize the notation introduced in [8] for infinite Huffman trees so that it can represent infinite $(1,2)$-lopsided trees. See Figure 2 for an example.

DEFINITION 2.1. *A* tree $T = \{(I_l, C_l, E_l)\}_{l=0}^{\infty}$ *is an infinite sequence of tuples of non-negative integers satisfying* $(I_0, C_0, E_0) = (1, 0, 0)$ *and*

$$\forall l \geq 0, \quad \begin{array}{l} E_{l+2} + I_{l+2} = I_{l+1} + I_l \\ C_{l+1} = I_l. \end{array} \tag{2.3}$$

The I_l, C_l and E_l are called *internal* nodes, *cross* nodes and *external* (or *leaf*) nodes respectively. Intuitively I_l and E_l are the number of internal nodes and leaves at level l; C_l counts the number of tree edges (of length 2) that pass through level l without stopping at level l. Given a tree T we define

$$\forall l \geq 0, \quad I_l(T) = I_l, \quad C_l(T) = C_l, \quad E_l(T) = E_l.$$

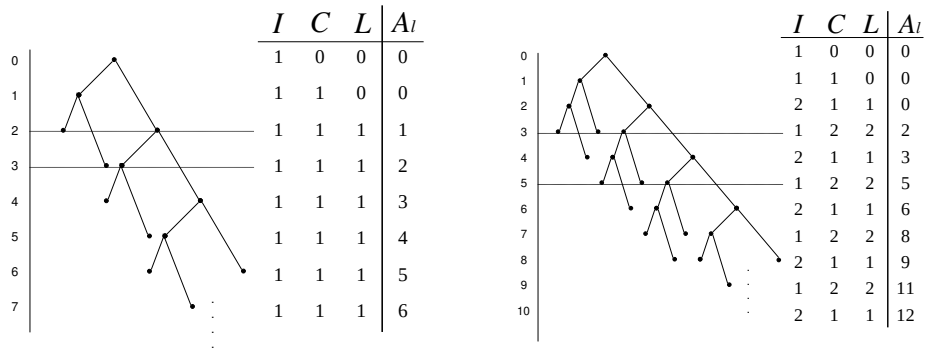
We also define $A_l(T) = \sum_{j \leq l} E_j(T)$ where A_l is the number of *leaves* on or above level l. When T is understood we will simply write A_l. We can now define Cost$(.,.)$ in a way that corresponds to the natural cost of a tree given a source.

DEFINITION 2.2. *Let T be a tree and $d_i(T), i = 0, 1, 2, \ldots$ be the depth of the ith leaf in T, breaking ties arbitrarily. Thus,*

$$d_i(T) = \min\{l : i < A_l(T)\}.$$

Let $\mathcal{P} = \{p_i\}_{i=0}^{\infty}$ be a nonincreasing sequence of nonnegative reals. The cost of T labelled by $\mathcal{P}$ *is the external path length of T when its leaves, sorted by increasing depth, are labelled with the p_i, i.e.,*

$$\mathit{Cost}(T, \mathcal{P}) = \sum_{0 \leq i} p_i d_i(T) = \sum_{0 \leq i} p_i \min\{l : i < A_l(T)\}.$$



I	C	L	A_l
1	0	0	0
1	1	0	0
1	1	1	1
1	1	1	2
1	1	1	3
1	1	1	4
1	1	1	5
1	1	1	6

I	C	L	A_l
1	0	0	0
1	1	0	0
2	1	1	0
1	2	2	2
2	1	1	3
1	2	2	5
2	1	1	6
1	2	2	8
2	1	1	9
1	2	2	11
2	1	1	12

Figure 2: Two examples of infinite trees and their corresponding $\{(I_l, C_l, E_l)\}_{l=0}^{\infty}$ description. For easy reference we also provide $A_j = \sum_{l \le j} E_l$

As examples note that the first tree in Figure 2 has cost

$$2p_0 + 3p_1 + 4p_2 + \ldots = \sum_{i=0}^{\infty}(i+2)p_i$$

while the second tree has cost

$$\begin{aligned} &3p_0 + 3p_1 + 4p_2 + 5p_3 + 5p_4 + 6p_5 + 7p_6 + \ldots \\ &= \sum_{i=0}^{\infty}[(2i+3)(p_{3i} + p_{3i+1}) + (2i+2)p_{3i+2}] \end{aligned}$$

Note that for geometric distribution $\mathcal{P}_p$ the cost has a particularly simple form:

LEMMA 2.1. *If p is fixed then*

$$\begin{aligned} Cost(T, \mathcal{P}_p) &= \sum_{0 \le i} p^i \min\{l : i < A_l(T)\} \\ &= \sum_{0 \le l} \sum_{A_l \le j} p^j \\ &= \frac{1}{1-p} \sum_{0 \le l} p^{A_l(T)}. \end{aligned} \tag{2.4}$$

Returning to the same examples the first tree in Figure 2 has cost

$$\frac{1}{1-p}\left(p^0 + p^0 + p^1 + p^2 + p^3 + \ldots\right) = \frac{1}{1-p}\left(1 + \frac{1}{1-p}\right).$$

Similarly, the second tree has cost

$$\begin{aligned} &\frac{1}{1-p}\left(p^0 + p^0 + p^0 + p^2 + p^3 + p^5 + p^6 + p^8 + \ldots\right) \\ &= \frac{1}{1-p}\left(2 + \sum_{i=0}^{\infty} p^i - \sum_{i=0}^{\infty} p^{1+3i}\right) \\ &= \frac{1}{1-p}\left(2 + \frac{1}{1-p} - \frac{p}{1-p^3}\right) \end{aligned}$$

DEFINITION 2.3. *A tree T is* optimal *for $\mathcal{P}_p$ if, for all trees T',*

$$Cost(T, \mathcal{P}_p) \le Cost(T', \mathcal{P}_p).$$

Our problem is, given p, to find an optimal tree for $\mathcal{P}_p$.

2.2 The Structure of Optimal Trees In [8] it is proven that optimal $(1,1)$ trees for geometric distributions have bounded width. The same idea can be applied to $(1,2)$ trees as well. We sketch the proofs here:

LEMMA 2.2. *(Optimal Trees have bounded width). Let p be fixed and $B(p) = \min\{k : p^k < \frac{1-p}{2}\}$. If T is any optimal tree for $\mathcal{P}_p$, then $\forall l, I_l(T) \le B(p)$. Furthermore, $\forall l, E_l(T) \le 2B(p)$ and $C_l(T) \le B(p)$.*

Proof. (Sketch) If there is some optimal T and l such that $I_l(T) > B(p)$ simply replace $I_l(T) - B(p)$ of the internal nodes on level l with leaves. It is a relatively straightforward calculation to show that the resulting tree has lower cost than T, contradicting optimality. The second part of the lemma follows from the fact that $\forall l > 1, E_l(T) \le I_{l-1}(T) + I_{l-2}(T)$ and $C_l(T) = I_{l-1}(T)$ □

An immediate consequence of this lemma is that there are at most $2B^3(p)$ different possible $\{(I_l, C_l, E_l)\}$ triples that can appear in any optimal tree for $\mathcal{P}_p$. This means that somewhere in the first $2B^3(p)$ levels of an optimal tree T, some level *must repeat.*

THEOREM 2. *(Cycle Theorem) Let p be fixed. Let T be an optimal tree for. Let t be the smallest index such that there exists an i' such that*

$$\begin{aligned} &(I_t(T), C_t(T), E_t(T)) = \\ &\quad (I_{t+i'}(T), C_{t+i'}(T), E_{t+i'}(T)). \end{aligned} \tag{2.5}$$

For this t let i be the smallest i' such that (2.5) is satisfied. Now define tree T' to be the tree that has the

same first t levels as T and then infinitely repeats the next i levels. That is

$$(I_l(T'), C_l(T'), E_l(T'))$$

$$= \begin{cases} (I_l(T), C_l(T), E_l(T)) & \text{if } l < t. \\ (I_{t+(l-t) \bmod i}(T), \\ C_{t+(l-t) \bmod i}(T), & \text{if } l \geq t. \\ E_{t+(l-t) \bmod i}(T)) \end{cases}$$

Then T' is also an optimal tree for $\mathcal{P}_p$.

Proof. (Sketch) Split T into three parts; levels $0, \ldots, t-1$ are $Head(T)$, levels $t, \ldots, t+i-1$ are $Mid(T)$ and levels $t+i, \ldots, \infty$ are $Tail(T)$. Let T^j be the infinite tree composed of $Head(T)$ followed by j copies of $Mid(T)$ followed by $Tail(T)$ (it is not difficult to see that T^j satisfies (2.3) and is therefore really a tree). The optimality of T can be shown to imply that each T^j is also optimal; otherwise T could be modified to build a lower cost tree. The trees T^j are all optimal and, *as a sequence,* converge levelwise to tree T'. Standard convergence theorems [17, 8] then imply that T' is optimal. □

The theorem and Lemma 2.1 together tell us, at least, theoretically, that our problem is solvable. Given any p we can simply construct the exponentially huge but bounded number of cyclic trees restricted to $I_l \leq B(p)$ and compare all of their costs for that value of p, returning the minimum one. In the next section we will see a much more efficient procedure.

Before doing so we first remark on one almost immediate result on the structure of the solution space of optimal trees.

COROLLARY 2.3. *The cost of any optimal tree T for p is a rational function in which the degrees of the numerators and denominators are upper-bounded by a function of $B(p)$.*

Proof. Let $M = A_{t+si} - A_{t+(s-1)i} = A_{t+i} - A_t$. From equation (2.4) and theorem 2,

$$\begin{aligned} & \text{Cost}(T, \mathcal{P}_p) \\ (2.6) \quad = \quad & \frac{1}{1-p} \sum_{0 \leq l} p^{A_l} \\ = \quad & \frac{1}{1-p} \left[\sum_{0 \leq l < t} p^{A_l} + \frac{1}{1-p^M} \sum_{t \leq l < t+i} p^{A_l} \right] \\ = \quad & \frac{\sum_{0 \leq l < t+i} p^{A_l} - \sum_{0 \leq l < t} p^{A_l+M}}{1-p-p^M+p^{M+1}}. \end{aligned}$$

How large can M be? As seen before $t+i \leq 2B^3(p)$ and $E(l) \leq 2B(p)$. Therefore $M \leq 4B^4(p)$. By definition $A_l \leq 2B(p)l$. Putting this all together we see that both the degrees of the numerators and denominators are upper-bounded by $8B^4(p)$. □

Now, note that given p, the number of cyclic trees that could possibly be optimal trees for any $p' \in (0, p]$ is bounded by a function of p. The cost of the optimal tree is therefore the lower envelope of a finite number of rational functions in p'. The numerators and denominators of the rational functions are also bounded by some universal constant in p so any pair of such functions only intersects a constant number of times. This implies that the lower envelope of these functions has complexity bounded by a function in p. Putting all of this together we see that the interval $(0, p]$ can be subdivided into a finite number of subintervals, each with an associated tree which is optimal for every p' in the subinterval. This implies

THEOREM 3. *(Partition theorem) There exists a monotonically increasing sequence of real numbers $\{q_i\}_{i=0}^{\infty}$ such that $q_0 = 0$ and $\lim_{i \to \infty} q_i = 1$ and a countable sequence of trees $\{T_i\}_{i=0}^{\infty}$ such that tree T_i is optimal for $\forall p \in (q_i, q_{i+1}]$.*

As mentioned earlier, Gallager and Voorhis' result (Theorem 1) implies this result for the standard Huffman case. Their proof was constructive. We have just seen that the result is not specific to the Huffman case but is also correct for the $(1, 2)$-lopsided case (and as discussed, can be extended to all of the other generalizations mentioned).

2.3 Algorithm In this section we will present an algorithm that, given p, finds an optimal $(1, 2)$-lopsided tree for $\mathcal{P}_p$. The idea behind our algorithm will be to modify the top-down algorithm developed in [10] that found a min-cost lopsided tree for a finite source by finding a min-cost path in a directed graph. That algorithm worked by starting from a node corresponding to the tree "root" and building trees top down. An edge in the graph corresponded to adding a level to the bottom of a tree; the cost of the edge corresponded to the "contribution" of that level to the tree's cost. Thus every path in the graph starting at the "root node" corresponded to some tree (a length k path corresponding to a depth k tree) with the cost of a path equaling the cost of the corresponding tree. After defining an appropriate destination node the algorithm for finding a min-cost lopsided tree was simply to find a min-cost root-destination path in the graph.

Although the trees, and associated paths, that we are now constructing, are infinite, we know from the previous subsection that optimal trees have the pattern of a head followed by cycles. Therefore our search

space is actually finite and we will be able to modify the algorithm of [10] to find a min-cost infinite path. Before discussing the algorithm we need to introduce some definitions.

We construct an infinite directed weighted graph $G = (V, E)$ where $V = \{(m; e, c) : m, e, c \text{ integers} \geq 0\}$; Each vertex $(m; e, c)$ has $e+1$ directed edges leaving it. For $0 \leq q \leq e$ the edges in E leaving the vertex are

$$((m; e, c), (m + e - q; c + q, q)). \tag{2.7}$$

The weight of all of the edges leaving $(m; e, c)$ will be $\frac{p^{m+1}}{1-p}$.

The reason for introducing this definition is that if we consider the vertices as corresponding to levels in a tree and edges as corresponding to the possibility of one level following another we can create a $1-1$ correspondence between trees and paths in G starting from $(0; 1, 1)$; in this correspondence a length k path will correspond to a depth k tree and an infinite path to an infinite tree.

Intuitively, vertex $(m; e, c)$ will correspond to the level of a tree that has m leaves *above it*, c cross edges *passing through it*, and e leaves plus internal nodes *on it*. More formally, recall our definition of an infinite tree as a sequence $T = \{(I_l, C_l, E_l)\}_{l=0}^{\infty}$ with $A_l(T) = \sum_{j \leq l} E_j(T)$ being the number of leaves on or above level l. Working through the details (omitted in this extended abstract) we can find that given tree T we can associate a path $\{(m_l; e_l, c_l)\}_{l=1}^{\infty}$ in G with $(m_1; e_1, c_1) = (0; 1, 1)$ and, for $i > 1$, $(m_l; e_l, c_l) = (A_{l-1}, I_l + E_l, C_l)$. In the other direction, given a path $\{(m_l; e_l, c_l)\}_{l=1}^{\infty}$ in G with $(m_1; e_1, c_1) = (0; 1, 1)$ we can associate a tree such that, $(I_0, C_0, E_0) = (1, 0, 0)$ and for $i \geq 1$ $(I_l, C_l, E_l) = (c_{l+1}, c_l, m_{l+1} - m_l)$. Furthermore, using Lemma 2.1 we can work out that the cost of a tree for $\mathcal{P}_p$ will be exactly equal to the cost of the associated infinite path. Thus, our problem of trying to find a minimum-cost tree is equivalent to finding a minimum-cost infinite path in G that starts at node $(0; 1, 1)$.

Of course, since the graph is infinite, we can not explicitly run a shortest path algorithm. Instead we use the structural properties from section 2.2 to restrict our search. The first thing to note is that Lemma 2.2 permits us to restrict $e, c \leq B(p)$. After doing this we run Dijkstra's shortest path algorithm starting at node $(0; 1, 1)$, building a shortest path tree in the graph.

The main observation is that the paths that we are building in the shortest-path-tree can considered as *prefixes* of an infinite path (these prefixes correspond to the top levels of an infinite tree). From Theorem 2 (the cycle theorem) we know that if some path $\{(m_l; e_l, c_l)\}_{l=1}^{n}$ is a finite prefix of an optimal path and, for some $i < n$, $(e_i, c_i) = (e_n, c_n)$, then the path must cycle starting from $n+1$ and we can figure out exactly what the corresponding cyclic tree is; applying Lemma 2.1 will give its cost.

This motivates us to modify Dijkstra's algorithm so that, at the time we add node $(m; e, c)$ into the tree, we walk backwards in the tree from $(m; e, c)$ until we either reach some other node (m', e, c) if such a node exists, or the root. If we find the root we just continue normally with Dijkstra's algorithm. If we find a node (m', e, c) then we have found a cycle corresponding to a candidate cyclic tree. We calculate the cost of this tree; if it has the minimum cyclic tree cost found so far we keep it as the new minimum, otherwise, we throw it away. In neither case do we continue processing $(m; e, c)$ by updating its neighbors' costs since we know that it can not contribute to any other candidate trees. Note that the algorithm must terminate since any path that gets longer than $2B^2(p)$ edges will contain some repeated (e, c) value. Furthermore, given any tree path $\{(m_l; e_l, c_l)\}_{l=1}^{n}$ we have $m_n \leq \sum_{l=1}^{n} e_l \leq n2B(p)$. Since $n \leq 2B^2(p)$ we have that $m_n \leq 4B^3(p)$ so we can use this restriction to make our graph finite. Figure 3 gives the pseudocode for the algorithm.

To prove that the algorithm gives the correct answer we must show that it finds an optimal cyclic tree. Let p be fixed and $\{(m_l; e_l, c_l)\}_{l=1}^{\infty}$ correspond to some optimal cyclic tree T. Let (t, i) be the minimum pair (lexicographically) such that $(e_t, c_t) = (e_{t+i}, c_{t+i})$. Since the prefix of a shortest path is a shortest path we know that $\{(m_l; e_l, c_l)\}_{l=1}^{t+i}$ is a shortest path. If this is the path to $(m_{t+i}, e_{t+i}, c_{t+i})$ found by the algorithm then the algorithm will, by definition, find T. A difficulty might arise if there are two possible shortest paths to $(m_{t+i}, e_{t+i}, c_{t+i})$ and the algorithm did not find $\{(m_l; e_l, c_l)\}_{l=1}^{t+i}$. It is possible to prove by contradiction, though, that if this happens, then the algorithm will find another cyclic tree that has the same cost as T (proof omitted in this extended abstract) and therefore it will always find some optimal cyclic tree.

2.4 Results As an illustration of the technique we implemented the algorithm in the previous section for finding optimal $(1, 2)$ lopsided trees and ran it for p ranging from 0.001 to 0.960 with 0.001 increments. Figure 6 shows the ranges such that all p sampled in the same range share the same optimal tree. Note that this seems to correspond to the statement of the Partition theorem, Theorem 3 (but see the comment at the end of the next section).

Figure 4 shows the path expansions of the optimal trees found and Figure 5 gives realizations of the actual trees (when we write $\mathcal{P}_{\sim 0.755}$ the $\sim$ denotes that our sampling indicates the boundary of interval to be close

Input: p

1. **Initialization**
   Set $\texttt{COST}[m; e, c] := \infty$ for all entries
   with $m < 4B^3(p)$ and $e, c < B(p)$.
   Set $\texttt{COST}[0; 1, 1] := 0$. $\texttt{PUSH}(Q, (0; 1, 1))$.
   /* The current minimum cost of infinite path. */
   Set $\texttt{min_cost} := \infty$.
2. **while** $Q \neq \emptyset$ **do**
2a. $(m; e, c) \leftarrow \texttt{EXTRACT_MIN}(Q)$.
   /* Check for repetition of (e, c) */
   for each vertex $(m'; e', c')$ on the
   path from the root to $(m; e, c)$ **do**
   if $(e', c') = (e, c)$ **then**
   /* Calculate cyclic tree cost */
2b. $\texttt{cost} := \texttt{COST}[m'; e', c']$
   $+ \frac{\texttt{COST}[m;e,c] - \texttt{COST}[m';e',c']}{1-p^{m-m'}}$.
   if $\texttt{min_cost} > \texttt{cost}$ **then**
   $\texttt{min_cost} := \texttt{cost}$.
   minpath $\leftarrow$ a list of vertices on the path.
   Goto 2.
   end for
   /* Relaxation */
2c. $\texttt{new_cost} := \texttt{COST}[m; e, c] + p^{m+1}/(1 - p)$.
2d. **for** $q := 0$ to e **do**
2e. Let $(m'; e', c') := (m + e - q; c + q, q)$.
   if $\texttt{COST}[m'; e', c'] > \texttt{new_cost}$ **then**
   $\texttt{COST}[m'; e', c'] := \texttt{new_cost}$.
   if $(m'; e', c')$ is not in Q
   AND $\texttt{COST}[m'; e', c'] < \texttt{min_cost}$ **then**
   $\texttt{PUSH}(Q, (m'; e', c'))$.
   end for
   end while
3. Extract tree from minpath.

Figure 3: Modified Dijkstra algorithm for finding optimal infinite tree

to 0.755 but we do not know the exact boundary). Note that there are many (uncountable) ways to realize the trees given the corresponding path expansions. We choose the presented ones because of their easily observable recursive structures.

3 Extensions, conclusions and open problems

Previous work on constructing minimum-cost prefix-free codes for infinite source alphabets were non-algorithmic and very restricted in their applicability. In this extended abstract we developed the first *algorithmic* technique for constructing minimum-cost prefix-free codes. This new technique is applicable to many generalizations of Huffman coding, e.g., unequal cost letters and restrictions on codewords, that the previously known technique could not work on. This new technique also works for a larger set of generalizations of geometric sources than could be solved for previously.

In this abstract we only described an algorithm for constructing an optimum $(1, 2)$-lopsided tree for a standard geometric source. Modifying the algorithm to work for other (α, β)-lopsided trees only requires changing the tuple definition of a tree given in Definition 2.3 to reflect the (α, β), i.e., instead of only saying how many edges are *crossing* a level the tuple has to encode how many edges are crossing the level at 1 unit into their length, how many at 2 units into their length, etc.. Similarly, the technique can be modified to deal with regular language restrictions on the codewords (for regular languages described by Deterministic Finite Automatons) by modifying the tuple definition to indicate how many nodes in each DFA state there are per level. Once these definition modifications are made all of the other lemmas and theorems can be modified to work as well. Similarly, for modifications of the geometric source such as (1.2) we can have the tuple encode at which α_t the labellings of the leaves on a particular level start and modify everything else accordingly.

The big open question in this area is to design a technique that would work for non-geometric-like sources. Unfortunately, this problem has been open for a long time and nothing seems to be known about how to attack it [2].

Another, more technical problem, would be to improve the *Partition Theorem* (Theorem 3). This theorem states that, for all the generalizations of Huffman coding discussed, the unit interval can be decomposed into a countable union of subintervals such that if p, p' are in the same subinterval then the same tree is optimal for both p and p'. This was a generalization of a theorem due to Gallager and Voorhis (Theorem 1) for the standard Huffman case. Gallager and Voorhis's result actually implies something much stronger for the standard Huffman case which is that *if tree T is optimal for both p and p' with $p < p'$ then T is also optimal for all $q \in [p, p']$*. The question is whether this statement holds for the general cases addressed by the partition theorem. If it did then our algorithm's results would be much stronger. For example, in Section 2.4 we presented the optimal trees for *sampled* values of p and found that all sampled p in the given intervals had the same optimal tree. If the italicized statement above held in the general case then we would know that those trees were optimal for *all* p in the given intervals, not just the sampled ones.

Interval	Expansion
$\mathcal{P}_{0.001} - \mathcal{P}_{0.500}$	$\underline{(0;1,1)} \to (1;1,0) \to \underline{(1;1,1)}$
$\mathcal{P}_{0.500} - \mathcal{P}_{\sim 0.683}$	$\underline{(0;1,1)} \to \underline{(0;2,1)} \to \underline{(1;2,1)}$
$\mathcal{P}_{\sim 0.683} - \mathcal{P}_{\sim 0.755}$	$(0;1,1) \to \underline{(0;2,1)} \to \underline{(0;3,2)} \to (2;3,1) \to \underline{(3;3,2)}$
$\mathcal{P}_{\sim 0.755} - \mathcal{P}_{\sim 0.809}$	$(0;1,1) \to (0;2,1) \to \underline{(0;3,2)} \to \underline{(1;4,2)} \to \underline{(3;4,2)}$
$\mathcal{P}_{\sim 0.809} - \mathcal{P}_{\sim 0.838}$	$(0;1,1) \to (0;2,1) \to (0;3,2) \to \underline{(0;5,3)} \to \underline{(3;5,2)} \to \underline{(5;5,3)}$
$\mathcal{P}_{\sim 0.838} - \mathcal{P}_{\sim 0.864}$	$(0;1,1) \to (0;2,1) \to (0;3,2) \to (0;5,3) \to \underline{(2;6,3)} \to \underline{(5;6,3)}$

Figure 4: Signature expansions for the first 6 intervals. Underlined signatures are where level repetitions occur.

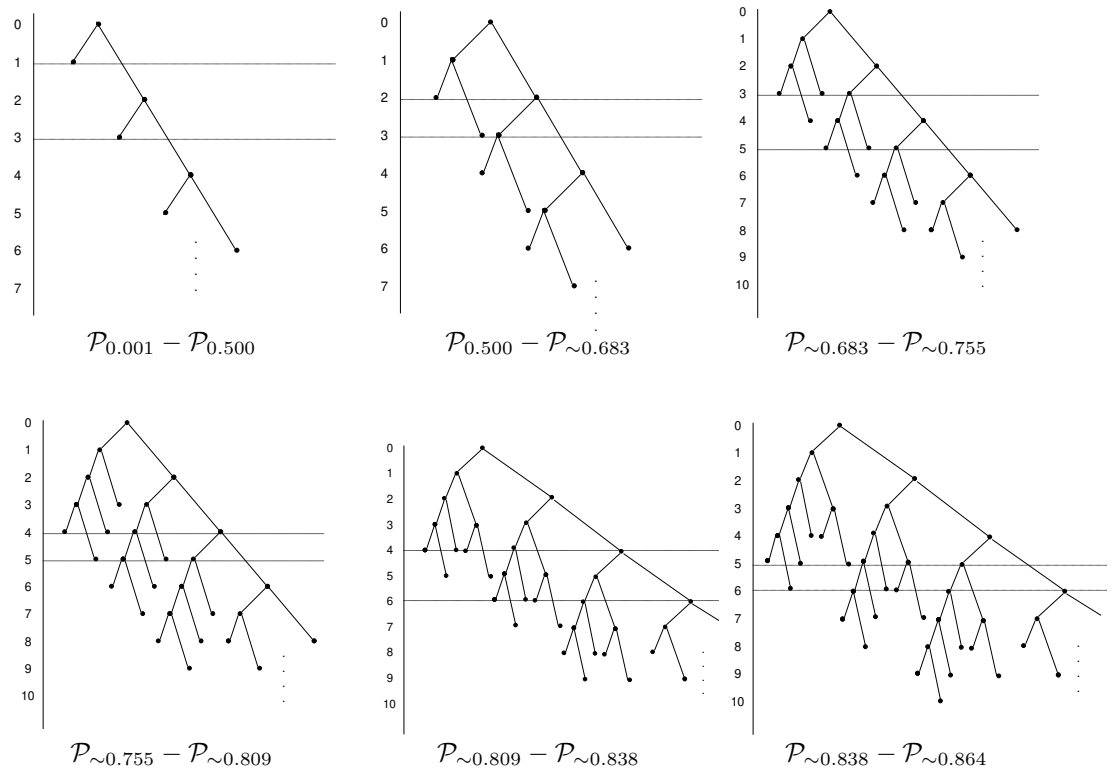


Figure 5: Optimal trees for the first 6 intervals. The two horizontal lines show the two levels that form the repetition.

Figure 6: Partition of $(0, 1)$ so that all sampled ps in the same interval share the same tree.

References

[1] Julia Abrahams, "Huffman-Type Codes for Infinite Source Distributions," *Journal of the Franklin Institute,* **331B**(3) (1994) 265-271.

[2] Julia Abrahams, "Code and Parse Trees for Lossless Source Encoding," *Communications in Information and Systems,* **1**(2)(April 2001) 113-146.

[3] Toby Berger and Raymond W. Yeung, "Optimum "1"-Ended Binary Prefix Codes," *IEEE Transactions on Information Theory* , **36**(6) (November, 1990) 1435-1441.

[4] S.L. Chan and M. Golin, "A Dynamic Programming Algorithm for Constructing Optimal "1"-ended Binary Prefix-Free Codes," *IEEE Transactions on Information Theory,* **46** (4) (July 2000) 1637-44.

[5] S. Dolev, E. Korach and D. Yukelson, "The Sound of Silence: Guessing Games for Saving Energy in Mobile Environment," *Eighteenth Annual Joint Conference of IEEE Computer and Communications Societies (IEEE INFOCOM'99),* (1999) 768-775.

[6] Robert G. Gallager and David C. Van Voorhis, "Optimal Source Codes for Geometrically Distributed Integer Alphabets," *IEEE Transactions on Information Theory,* (March 1975) 228-230.

[7] E.N. Gilbert, "Coding with Digits of Unequal Cost," *IEEE Transactions on Information Theory,* **41**(2) (March 1995) 596-600

[8] Mordecai Golin, "A Combinatorial Approach to Golomb Forests," *Theoretical Computer Science,* **263**(1-2) (July 2001) 283-304.

[9] Mordecai Golin, Claire Kenyon, and Neal Young, "Huffman Coding with Unequal Letter Costs," *Proceedings of the 34th ACM Symposium on Theory of Computing (STOC2002),* (May 2002) 785-791.

[10] 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.

[11] S.W. Golomb, "Run Length Encodings," *IEEE Transactions on Information Theory,* **IT-12** (July 1966) 399-401.

[12] R. Hassan, "A Dichotomous Search for A Geometric Random Variable," *Operations Research,* **32**(2) (1984) 423-439.

[13] D. A. Huffman, "A Method for the Construction of Minimum-Redundancy Codes," *Proceedings of the IRE,* **40** (September 1952) 1098-1101.

[14] F. K. Hwang, "On Finding a Single Defective in Binomial Group Testing," *Journal of the American Statistical Association,* **69**(345) (1974) 146-150.

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

[16] A. Kato, Te Sun Han and H. Nagaoka, "Huffman coding with an infinite alphabet.," *IEEE Transactions on Information Theory,* **42** (3) (May 1996) 977-84.

[17] Tamas Linder, Vahid Tarokh and Kenneth Zeger, "Existence of Optimal Prefix Codes for Infinite Source Alphabets," *IEEE Transactions on Information Theory,* **43**(6) (November 1997) 2026-2028.

[18] N. Merhav, G. Seroussi and M. J. Weinberger, "Optimal Prefix Codes for Sources with Two-Sided Geometric Distributions," *IEEE Transactions on Information Theory,* **46**(1) (Jan 2000) 229-236.

[19] Y. C. Yao and F. K. Hwang, "On Optimal Nested Group testing Algorithms," *Journal of Statistical Planning and Inference,* **24** (1990) 167-175