Algorithms for Constructing Infinite Huffman Codes (Expanded version)*

Mordecai J. Golin
Dept. of Computer Science
Hong Kong U.S.T.
Clear Water Bay
Kowloon, Hong Kong
golin@cs.ust.hk

Kin Keung Ma
Dept. of Computer Science
Hong Kong U.S.T.
Clear Water Bay
Kowloon, Hong Kong
kkma@cs.ust.hk

Abstract

The structure of optimal (minimum-cost) prefix-free codes for geometrically distributed sources was first derived by Gallager & Van Voorhis. Their approach was later extended by Merhav, Seroussi and Weinberger to derive optimal codes for "two-sided" geometric distributions. The technique used combines (i) guessing optimal codes and then (ii) validating them using the Huffman encoding algorithms.

This paper presents the first *algorithmic* approach for constructing optimal prefix-free codes for (infinite) geometrically distributed sources. This approach will work for even more complicated generalizations of geometric sources where solutions cannot 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.

The basis of this new approach is a combinatorial derivation of the cyclic properties of optimal codes for infinite geometric (like) sources. This will permit designing efficient algorithms that restrict themselves to searching among codes possessing these properties.

1 Introduction

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

Let $\Sigma = \{0, 1, \ldots, r-1\}$ be an r-ary alphabet. A *Code* is simply a subset of Σ^*. For $w_1 = \sigma_1^1, \sigma_2^1, \ldots, \sigma_{\ell_1}^1 \in \Sigma^*$ and $w_2 = \sigma_1^2, \sigma_2^2, \ldots, \sigma_{\ell_2}^2 \in \Sigma^*$, w_1 is a *prefix* of w_2 if $\ell_1 \le \ell_2$ and, $\forall i \le \ell_1$, $\sigma_i^1 = \sigma_i^2$. A code $C \subseteq \Sigma^*$ is prefix-free (usually just called a *prefix-code*) if $\forall w, w' \in C$, w is not a prefix of w'. Prefix codes are

*This work was partially supported by Hong Kong CERG grants HKUST6162/00E, HKUST6082/01E and HKUST6206/02E. A preliminary version appeared in the *Proceedings of the Sixteenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'04).* This "Expanded Version" differs from the regular version of the same paper only by the addition of subsection 4.3 and having many more implementation results presented in the appendix. Last revised on August 10, 2004.

of special interest because a message written as the concatenation of words in a prefix-code is uniquely decipherable and in an efficient fashion.

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-code of n r-ary words $\{w_i\}_{i=0}^{n-1} \subset \Sigma^*$ 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 symbols in w_i, e.g., for $\Sigma = \{0, 1, 2\}$, $\text{length}(0212) = 4$.

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* [15].

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 nearly as well understood. It was proven[2] in [19] 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 [13], 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. **B**s occurring with probability p and **A**s with probability $1-p$. Now, for $i = 0, 1, 2, \ldots$ set $X_i = \underbrace{\mathbf{BB}\ldots\mathbf{BB}}_{i \text{ times}}\mathbf{A}$. *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 [14] and group testing [16] [21].

This special case of $\mathcal{P} = \mathcal{P}_p$ was studied by Gallager and Van Voorhis [8] who exhibited an optimal tree for every p. Their technique was to first define

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

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

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 of Huffman trees, and then showed that these codes "converge" to an infinite tree that is optimal for the infinite source.

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. Therefore, given an infinite sequence of recursively defined finite distributions along an associated infinite sequence of recursively defined trees, it might possible to recursively show that each tree possesses the sibling property and is therefore optimal. This is what Gallager and Van Voorhis did.

Using notation later introduced in this paper their result can be stated as:

Theorem 1.1 *(Gallager and Van Voorhis [8]) Given p, let m be the unique integer that satisfies*

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

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

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

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 $\forall p \in (q_{m-1}, q_m]$, M_m is an optimal tree for $\mathcal{P}_p$.

These optimal codes (trees) were later shown (in [10]) to be the unique[3] optimal codes for the given sources. Gallager and Van Voorhis' basic technique was later expanded upon and extended by [1] and [18] who showed that it could be applied to other families of infinite sources to prove the existence of the optimal trees (but not to actually find 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 technique was recently pushed even further by [20] who constructed optimal codes for two-sided geometric

[3]Technically, [10] showed that if $p \in (q_{m-1}, q_m)$ then M_m is the unique optimal tree for $\mathcal{P}_p$, while if $p = q_m$ then there are an uncountable number of optimal trees for $\mathcal{P}_p$, all of which are combinations of M_{m-1} and M_m.

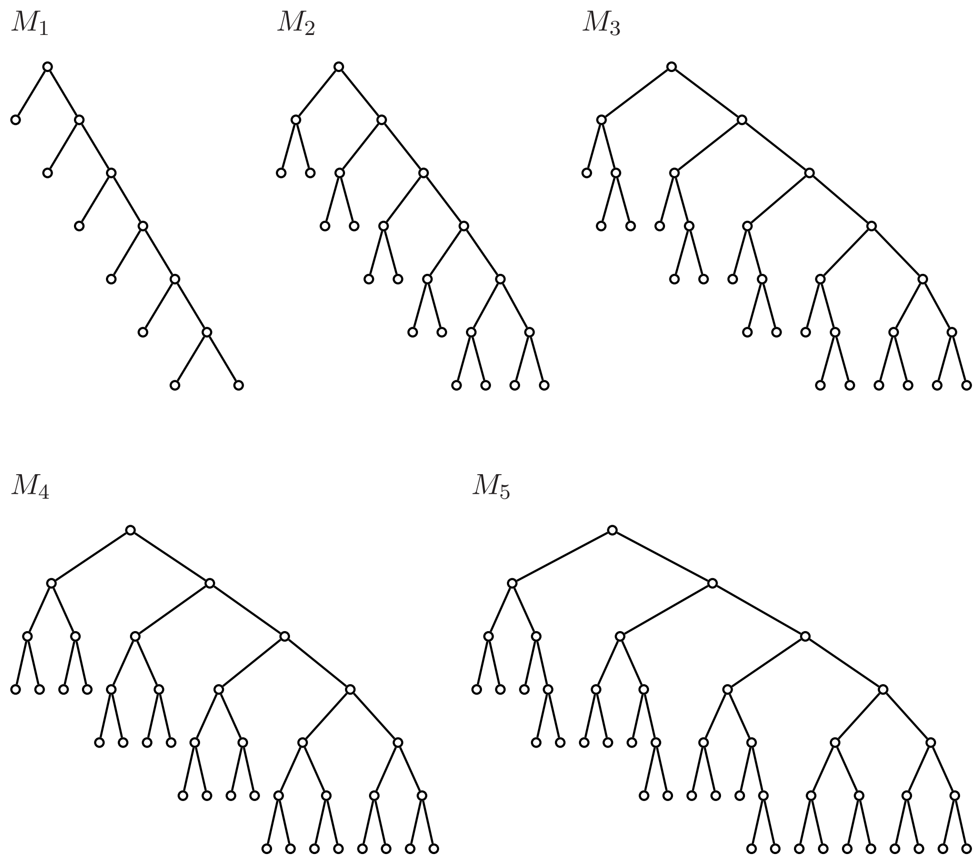


Figure 1: The tops of the infinite "optimal" trees M_1, M_2, M_3, M_4 and M_5.

Figure 2: The intervals $(q_m, q_{m+1}]_{m=1}^{\infty}$ partition $(0, 1)$ into a countable set of intervals and that $\forall p \in (q_{m-1}, q_m]$, M_m is an optimal tree for $\mathcal{P}_p$.

distributions (which have become useful in lossless image compression schemes). These are distributions $\mathcal{P}_{p,d}$ consisting of the infinite sequence

$$p_{x,d} = p^{|x+d|}, \quad x = 0, \pm1, \pm2, \pm3, \ldots \tag{2}$$

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 up until 2001 may be found in [2].

All of these results (except for [10] which proved 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 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.

1.1 Our new results

The primary advantage of the approach used in [8, 1, 20] 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 $\mathcal{P}_p$ or $\mathcal{P}_{p,d}$

A major disadvantage is that the approach only works if it is possible to *verify* 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 [17, 12] or has restrictions on the codewords, e.g., the one ended codes of [3] or the restricted-zero codes of [7] ([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 [20] seem to be as far as the technique can be pushed.

In this paper we develop the first *algorithmic* technique for constructing optimal infinite prefix-free codes for unequal cost encoding alphabets, e.g., those that 'charge' more to transmit a **0** than to transmit a **1**. The technique will also allow constructions of optimal prefix-free codes for many generalizations of geometric distributions. For example, it will allow distributions of the type

$$\begin{aligned}\mathcal{P}_{p,\alpha} \;=\; & p^{\alpha_0}, p^{\alpha_1}, p^{\alpha_2}, \ldots, p^{\alpha_{\mathcal{K}-1}}, \\ & p^{1+\alpha_0}, p^{1+\alpha_1}, p^{1+\alpha_2}, \ldots, p^{1+\alpha_{\mathcal{K}-1}}, \\ & p^{2+\alpha_0}, p^{2+\alpha_1}, p^{2+\alpha_2}, \ldots, p^{2+\alpha_{\mathcal{K}-1}}, \\ & p^{3+\alpha_0}, p^{3+\alpha_1}, p^{3+\alpha_2}, \ldots, p^{3+\alpha_{\mathcal{K}-1}}, \ldots\end{aligned}$$

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

where $\alpha = (\alpha_0, \alpha_1, \ldots, \alpha_{\mathcal{K}})$, $0 \leq \alpha_0 \leq \alpha_1 \leq \alpha_2 \leq \cdots \leq \alpha_{\mathcal{K}-1} < 1$. This is a natural generalization of the two-sided distribution (2) of [20] (which is essentially the case $\mathcal{K} = 2$ with the extra constraint $\alpha_0 + \alpha_1 = 1$)

Given a description of the problem to be solved, e.g., to find the optimal prefix-free code for distribution $\mathcal{P}_p$ under the condition that a **0** has length 1 and a **1** has length 2, our algorithm will take as input a value of $p \in (0, 1)$ and output a minimal cost prefix free code.

There are two major ideas underlying our technique. The first is the fact – first proven for finite-source unequal-cost coding in [12] and later modified for one-ended 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 new problem this implies that we are searching for a *minimum cost infinite-link path in some infinite weighted graph.*

The second idea generalizes the observation [10] 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; in Figure 1 the tops of the first 5 optimal infinite trees illustrates this phenomenon.

In the graph world this last observation will be shown to imply that a *minimum cost infinite-link path* must have a particular finite cyclic structure, i.e., a "head" followed by an infinitely repeated cycle. We will be able to use this idea to modify the standard Dijkstra shortest path algorithm to look for a finite path that generates a minimum cost infinite cycle.

The remainder of the paper is organized as follows: In Section 2, we illustrate our approach by deriving properties of optimal $(1, 2)$-lopsided Huffman tree for the geometric source $\mathcal{P}_p$. We then use these properties to construct an algorithm that, given input $p \in (0, 1)$, outputs the optimal $(1, 2)$-lopsided Huffman tree for $\mathcal{P}_p$. Note that even though the tree is infinite it will have a cyclical structure and can therefore be described using finite space.We conclude the section by presenting results derived by running the algorithm. In Section 3 we show how to extend the algorithm developed in Section 2 so that it works to construct optimal trees for *any* integer letter costs. In Section 4 we then show how to extend our result to the generalized geometric distributions $\mathcal{P}_{p,\alpha}$ described in (3). Finally, we conclude in Section 5 by discussing the limitations of our techniques and posing some open questions.

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

This section is devoted to deriving properties of minimum cost $(1, 2)$-lopsided trees for geometric distributions $\mathcal{P}_p$ and then using these properties to develop an algorithm for constructing such trees. We concentrate on the details of this particular problem because it is the simplest lopsided tree case and therefore permits developing our combinatorial results without being encumbered by technical details needed by the general lopsided tree case. In Section 3 we will sketch how to modify the results of this section so that they work for the general $(\gamma_1, \gamma_2, \ldots, \gamma_r)$-*lopsided tree* case..

We start by introducing the problem. In the general unequal-cost or *lopsided tree* problem, we are given an encoding alphabet $\Sigma = \{\sigma_1, \sigma_2, \ldots, \sigma_r\}$ and weights $\gamma_1, \gamma_2, \ldots, \gamma_r$ such that $\mathrm{length}(\sigma_i) = \gamma_i$. The length of a word w is the sum of the lengths of its characters. For example, in the $(1, 2)$-case, $\Sigma = \{0, 1\}$, $\mathrm{length}(0) = 1$ and $\mathrm{length}(1) = 2$ (this is sometimes known as the *telegraph channel* [9]) so $\mathrm{length}(1000) = 5$.

Given the γ_i's the $(\gamma_1, \gamma_2, \ldots, \gamma_r)$-*lopsided tree* problem is to find a prefix-free set of codewords w_i such that $\sum_i p_i \mathrm{length}(w_i)$ is minimized. Equivalently, by setting the length of the i-th edge leaving a node to be γ_i 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. Figure 3 shows a $(2, 3, 5)$-lopsided tree with probabilities assigned to its leaves. Its expected external path length is 6.35.

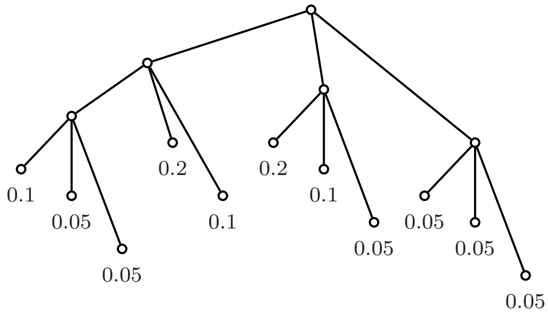
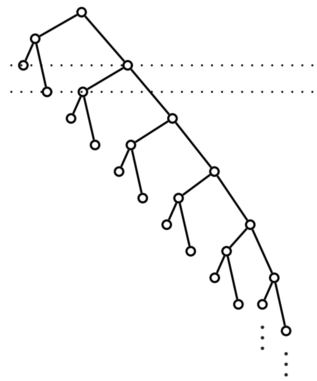


Figure 3: A $(2, 3, 5)$-lopsided tree.

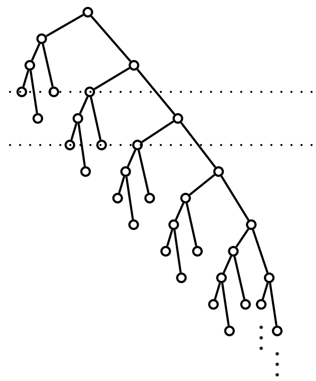
For finite source distributions, even though the Huffman algorithm does not work, when $(\gamma_1, \gamma_2, \ldots, \gamma_r)$ is fixed and integral there efficient algorithms for solving this problem exactly, e.g., [17, 12] and approximately, e.g., [11]; when the $(\gamma_1, \gamma_2, \ldots, \gamma_r)$ are given as inputs, 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 method, efficient or not, for finding the tree. The Gallager and Van Voorhis [8] techniques can not be applied because they require using the Huffman algorithm to verify a (countably) infinite number of 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 geometric distribution $\mathcal{P}_p = \{p^i\}_{i=0}^{\infty}$ defined for $p \in (0, 1)$.

The remainder of this section is split into three parts: Subsection 2.1 introduces the notation to be used. Subsection 2.2 introduces optimal forests and tools for manipulating them. Subsection 2.3 uses these tools for deriving the cyclic properties of optimal trees. Finally, subsection 2.4 uses these properties to develop an algorithm for constructing $(1, 2)$-lopsided trees. Actual results of the implemented algorithm are given in the appendix.



l	I_l	C_l	E_l	A_l
0	1	0	0	0
1	1	1	0	0
2	1	1	1	1
3	1	1	1	2
4	1	1	1	3
5	1	1	1	4
6	1	1	1	5
7	1	1	1	6
8	1	1	1	7
9	1	1	1	8
10	1	1	1	9
11	1	1	1	10
12	1	1	1	11



l	I_l	C_l	E_l	A_l
0	1	0	0	0
1	1	1	0	0
2	2	1	1	0
3	1	2	2	2
4	2	1	1	3
5	1	2	2	5
6	2	1	1	6
7	1	2	2	8
8	2	1	1	9
9	1	2	2	11
10	2	1	1	12
11	1	2	2	14
12	2	1	1	15

Figure 4: Two examples of infinite trees and their corresponding $\{(I_l, C_l, E_l)\}_{l=0}^{\infty}$ description. For easy reference we also provide $A_j = \sum_{l \le j} E_l$.

2.1 Notation for Describing Infinite Lopsided Trees

In this subsection we generalize the notation introduced in [10] for infinite Huffman trees so that it can represent infinite $(1,2)$-lopsided trees. See Figure 4 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*

$$\forall l \ge 0, \qquad \begin{array}{l} E_{l+2} + I_{l+2} = I_{l+1} + I_l \\ C_{l+1} = I_l. \end{array} \tag{3}$$

The I_l, C_l and E_l are, respectively, *internal* nodes, *cross* nodes and *external* (or *leaf*) nodes.. 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. We will use

$$L_l = (I_l, C_l, E_l)$$

to denote the l^{th} level of T. When T is not implicit we will also use the notation

$$I_l(T) = I_l, \quad C_l(T) = C_l, \quad E_l(T) = E_l, \quad L_l(T) = (I_l(T), C_l(T), E_l(T))$$

to denote the I_l C_l, E_l and L_l corresponding to T. We also define $A_l(T) = \sum_{j \le l} E_j(T)$ where $A_l(T)$ is the number of *leaves* on or above level 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 $d_i(T), i = 0, 1, 2, ...$ 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.,*

$$Cost(T,\mathcal{P}) = \sum_{0\le i} p_i d_i(T) = \sum_{0\le i} p_i \min\{l : i < A_l(T)\}. \tag{4}$$

As examples note that the first tree in Figure 4 has cost

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

while the second tree has cost

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

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

Lemma 2.1 *If* p *is fixed then*

$$\begin{aligned} Cost(T,\mathcal{P}_p) &= \sum_{0\le i} p^i \min\{l : i < A_l(T)\} \\ &= \sum_{0\le l}\sum_{A_l\le j} p^j = \frac{1}{1-p}\sum_{0\le l} p^{A_l(T)}. \end{aligned} \tag{5}$$

Returning to the our examples the first tree in Figure 4 has cost

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

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',

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

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

2.2 Combinatorial Operations on Trees and Forests

Our approach will be to prove structural properties of optimal trees and then use these properties to design an algorithm. More specifically, our ultimate aim is to prove the *Cycle Theorem* (Theorem 2.4) which states that, if T is any optimal tree, then it is possible to chop off some known finite top of T, and infinitely cycle the tail of the top section to create a cyclic tree which is also optimal.

It will be actually be easier to prove theorems on the more general class of *forests* where the only difference between a tree and a forest is that, while the top level of a tree has only *one internal* node, the top level of a forest can have many internal nodes, leaves and cross nodes.

More formally, a $(1,2)$-lopsided forest is structurally the same as a $(1,2)$-lopsided tree as defined in (3), without the condition that $(I_0, C_0, E_0) = (1,0,0)$.

Definition 2.4 *A* forest $F = \{(I_l, C_l, E_l)\}_{l=0}^{\infty}$ *is an infinite sequence of tuples of non-negative integers satisfying*

$$\forall l \geq 0, \qquad \begin{array}{l} E_{l+2} + I_{l+2} = I_{l+1} + I_l \\ C_{l+1} = I_l. \end{array} \tag{6}$$

The cost of a forest given source $\mathcal{P}_p$ is then naturally given by $\mathrm{Cost}(F, \mathcal{P}_p) = \sum_{0 \leq i} p_i d_i(F)$. To define optimality we need the notion of *compatibility.*

Definition 2.5 *Two levels $L = (I, C, E)$ and $L' = (I', C', E')$ are said to be compatible if $I + E = I' + E'$ and $C = C'$.*

Optimality for forests is defined as:

Definition 2.6 *A forest F is optimal for $\mathcal{P}_p$ if, for all forests F' with $L_0(F')$ compatible with $L_0(F)$,*

$$\mathit{Cost}(F, \mathcal{P}_p) \leq \mathit{Cost}(F', \mathcal{P}_p).$$

When $L_0(F) = L_0(F') = (1,0,0)$, i.e., are trees, this reduces to Definition 2.3.

We can now define some useful combinatorial operations on $(1,2)$-lopsided forests. In what follows we assume that

$$F = \{L_0, L_1, L_2, L_3, ...\} \text{ and } F' = \{L'_0, L'_1, L'_2, L'_3, ...\}.$$

Definition 2.7 *See Figures 5, 6 and 7 for illustrations of the below.*
Let $i, j \geq 0$ be integers.

$$\begin{aligned} F^{<i>} &= \{(I_0, C_0, i), L_1, L_2, L_3, ...\} \\ \mathit{Truncate}(F, j) &= \{L_j, L_{j+1}, L_{j+2}, L_{j+3}, ...\} \\ H_i &= \{(1, i, 0), (1, 1, i), (1,1,1), (1,1,1), (1,1,1), (1,1,1), ...\} \end{aligned}$$

Suppose L'_0 and L_j are compatible, then

$$\mathit{Replace}(F, F', j) = \{L_0, L_1, L_2, ..., L_{j-1}, L'_0, L'_1, L'_2, L'_3, ...\}$$

Suppose, for $i < j$, L_i and L_j are compatible, then

$$\begin{aligned} \textit{Repeat}(F,i,j) &= \{L_0, L_1, ..., L_i, ..., L_{j-1}, L_i, ..., L_{j-1}, L_j, L_{j+1}, ...\} \\ \textit{Cycle}(F,i,j) &= \{L_0, L_1, ..., L_i, ..., L_{j-1}, L_i, ..., L_{j-1}, L_i, ..., L_{j-1}, ...\} \end{aligned}$$

Intuitively, $F^{<i>}$ is F with i leaf nodes on its top level and Truncate(F, j) is F starting at its j^{th} level. Replace(F, F', j) has as its head (top) the first $j-1$ levels of F followed by a tail (bottom) which is F'. Cycle(F, i, j) has as its head the first $i-1$ levels of F and as its tail an infinite cycle of levels $L_i, \ldots, L_{j-1}$.

The following lemma states that many of these operations preserve optimality of forests.

Lemma 2.2 *If F and F' are optimal for p then*

1. *$\forall i \le E_0$, $F^{<i>}$ is optimal.*
2. *$\forall j \ge 0$, Truncate(F, j) is optimal.*
3. *If L_0' and L_j are compatible, then Replace(F, F', j) is optimal.*
4. *If L_i and L_j are compatible, then Repeat(F, i, j) is optimal.*
5. *For $i < j$, Cost$(H_i) >$ Cost(H_j).*

Proof. First we prove that (1.) is true. Suppose that for some $i \le E_0$, there exists an optimal forest F_* such that $L_0(F_*)$ and $L_0(F^{<i>})$ are compatible and $\text{Cost}(F_*) < \text{Cost}(F^{<i>})$. Consider $F_*^{<E_0'+E_0-i>}$ where $L_0(F_*^{<E_0'+E_0-i>})$ and $L_0(F)$ are compatible. Then,

$$\text{Cost}(F_*^{<E_0'+E_0-i>}) = p^{E_0-i}\text{Cost}(F') < p^{E_0-i}\text{Cost}(F^{<i>}) = \text{Cost}(F)$$

contradicting the optimality of F. Thus $F^{<i>}$ is optimal and (1.) follows.

To prove that (2.), (3.) and (4.) are true, first notice that F can be divided into two parts: a top part consisting of levels $0, 1, ..., j-1$, and a bottom part which is Truncate(F, j). Replace(F, F') is constructed by keeping the top part of F but replacing the bottom part with F'. Thus,

$$\text{Cost}(F) - \text{Cost}(\text{Replace}(F, F', j)) = p^{A_{j-1}}(\text{Cost}(\text{Truncate}(F, j) - \text{Cost}(F')). \tag{7}$$

Suppose that for some j, there exists an optimal forest F_* such that $L_0(F_*)$ and $L_j(F)$ are compatible and $\text{Cost}(F_*) < \text{Cost}(\text{Truncate}(F, j))$. Then equation 7 implies $\text{Cost}(\text{Replace}(F, F_*, j)) < \text{Cost}(F)$ which contradicts the optimality of F. This proves (2.).

To prove (3.), note that if F' is optimal with $L_0(F')$ and $L_j(F)$ being compatible then (2.) implies that $\text{Cost}(F') = \text{Cost}(\text{Truncate}(F, j))$ so equation 7 implies $\text{Cost}(\text{Replace}(F, F', j)) = \text{Cost}(F)$ and the optimality of Replace(F, F', j).

(4.) follows from the fact that Repeat$(F, i, j) = $ Replace$(F, \text{Truncate}(F, i), j)$.

Lastly, (5.) holds since for all k, the depth of the leaf assigned to p_k in H_i is larger than or equal to the one in H_j. □

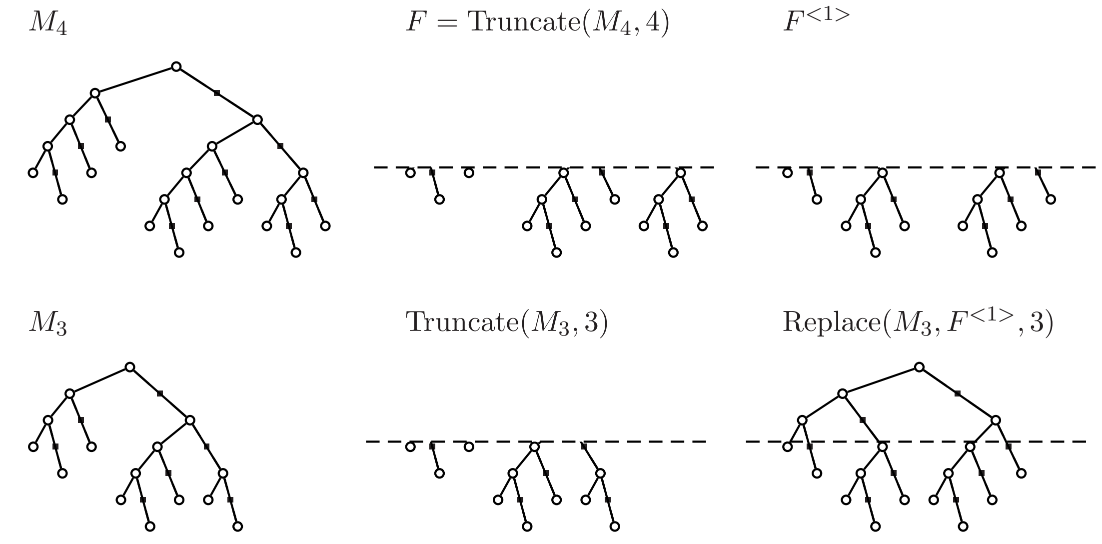


Figure 5: M_3 and M_4 are two infinite trees (with only their tops shown). The other forests are examples of the operations introduced in Definition 2.7. For purposes of illustration, forests defined using the Truncate() operator are left on the same level at which they appeared in their original trees.

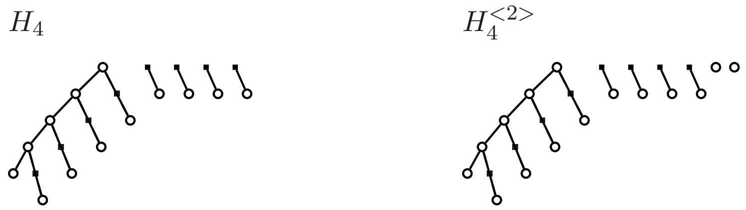


Figure 6: The forests H_4 and $H_4^{<2>}$. With the extra leaves on level 0 in $H_4^{<2>}$, it is obvious that $\text{Cost}(H_4) > \text{Cost}(H_4^{<2>})$.

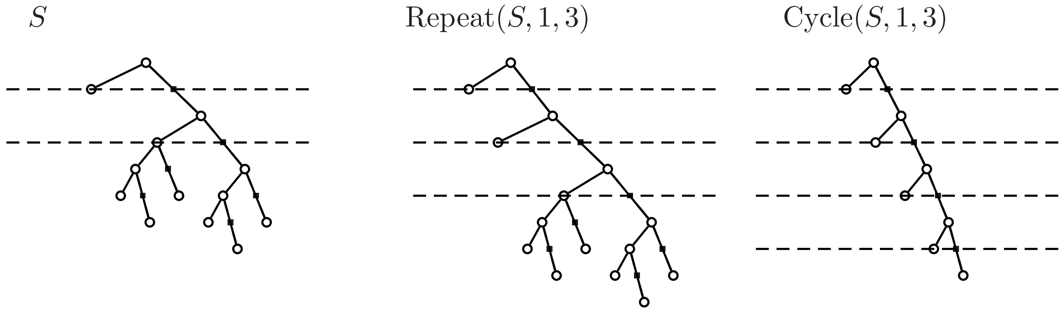


Figure 7: Demonstration of Repeat() and Cycle() operations on tree S.

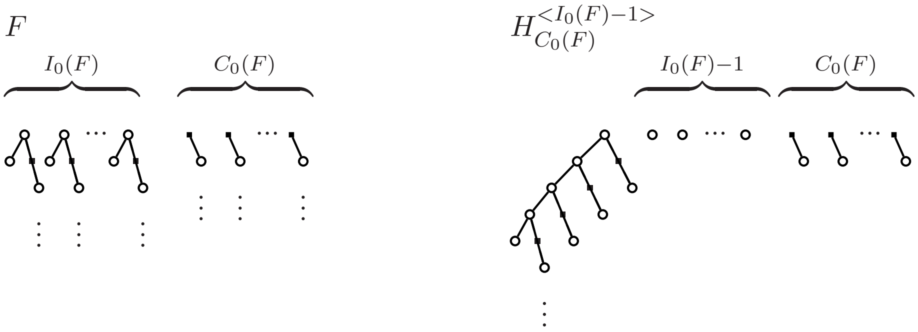


Figure 8: Illustration of the proof of Lemma 2.3.

2.3 Structure of Optimal Forests

We are now in a position to derive structural properties of optimal forests. Since trees are special cases of forests, these properties will obviously also hold for trees.

In [10] it is proven that optimal $(1,1)$-forests (that is, forests with *equal-cost* coding letters) for geometric distributions have bounded width. The same can be proven for $(1,2)$-forests as well.

Lemma 2.3 *(Optimal Forests have bounded width).*
Let p be fixed and $B(p) = \min\left\{k : p^k < \frac{1-p}{2}\right\}$. If F is any optimal forest for $\mathcal{P}_p$, then $\forall l, I_l(F) \leq B(p)$. Furthermore, $\forall l, E_l(F) \leq 2B(p)$ and $C_l(F) \leq B(p)$.

Proof. (See Figure 8) Suppose by contradiction that F is optimal and $I_l(F) > B(p)$ for some l. Without loss of generality we may assume that $l = 0$ and $E_l(F) = 0$. Otherwise, by Lemma 2.2, we may replace F by the optimal forest $\text{Truncate}(F, l)^{<0>}$.

Since all leaves of F are below level 0, $\frac{1}{1-p} < \text{Cost}(F)$. But then

$$\begin{aligned}\text{Cost}(H_{C_0(F)}^{<I_0(F)-1>}) &\leq \text{Cost}(H_0^{<I_0(F)-1>}) \\ &= \frac{p^{I_0(F)-1}(2-p)}{(1-p)^2} \\ &< \frac{2p^B}{(1-p)^2} < \frac{1}{1-p} < \text{Cost}(F)\end{aligned}$$

contradicting the optimality of F. The second part of the lemma follows from the fact that $\forall l > 1, E_l(F) \leq I_{l-1}(F) + I_{l-2}(F)$ and $C_l(F) = I_{l-1}(F)$ $\square$

An immediate consequence of this lemma is that there are at most $3B^2(p)$ different possible $\{(I_l + E_l, C_l)\}$ pairs that can appear as a level in any optimal forest for $\mathcal{P}_p$. This implies that somewhere in the first $3B^2(p)$ levels of an optimal forest F, some two levels are compatible to each other. We are now able to prove that there always exists some optimal tree which is cyclic, i.e., is composed of some finite head followed by some finite sequence of levels that are infinitely repeated.

Theorem 2.4 *(Cycle Theorem) Let p be fixed and F be an optimal forest. Let (t, i) be a pair with $t+i \leq 3B^2(p)$ such that $L_t(F)$ and $L_{t+i}(F)$ are compatible. Then $F' = Cycle(F, t, t+i)$ is also an optimal forest for $\mathcal{P}_p$.*

Proof. Let $C = \mathrm{Cost}(F)$ which is optimal. Let F_j, $j = 0, 1, 2, ...$, be a sequence of forests such that $F_0 = F$ and $F_j = \mathrm{Repeat}(F_{j-1}, t, t+i)$. By Lemma 2.2, Repeat(.) preserves optimality. Since F_0 is optimal, by induction $\mathrm{Cost}(F_j) = C$ for all j. Let $\mathrm{Cost}_j(F) = \sum_{d=1}^{j} \left(\sum_{i=A_{d-1}(F)}^{A_d(F)-1} dp^i \right)$ be the cost contributed by the leaves for the *first* j levels of F. Then, $\forall j$,

$$\mathrm{Cost}_j(F') = \mathrm{Cost}_j(F_j) < C.$$

But, by the definition of cost,

$$\mathrm{Cost}(F') = \lim_j \mathrm{Cost}_j(F') = \lim_j \mathrm{Cost}_j(F_j) \leq C.$$

Since $\mathrm{Cost}(F') \geq C$ this implies $\mathrm{Cost}(F') = C$ and the optimality of F'. $\square$

This theorem and Lemma 2.1 together tell us that since we have a finite search space, our problem, at least theoretically, is solvable. Given any p we can construct all of the possible tops of trees that could be the top of an optimal tree, i.e., sequences of at most $2B^2(p)$ triples with each level having $I_l \leq B(p)$. For each of these tops we construct the cyclic tree defined by Theorem 2.4 and calculate its cost, finishing by returning the cyclic tree with minimum cost. This procedure is not practical though, since in the worst case we would have to examine on the order of $\Theta\left((B(p))^{2B^2(p)}\right)$ trees; in the next section we will see a much more efficient procedure.

Before continuing we first note one almost immediate result on the structure of the solution space of optimal trees which is of independent interest.

Corollary 2.5 *The cost of any optimal tree T for p is a rational function*[5] *of p in which the degrees of the numerators and denominators are upper-bounded by $12B^3(p)$.*

Proof. Let $M = A_{t+si} - A_{t+(s-1)i} = A_{t+i} - A_t$. From equation (5) and Theorem 2.4,

$$\begin{aligned}
\mathrm{Cost}(T, \mathcal{P}_p) &= \frac{1}{1-p} \sum_{0 \leq l} p^{A_l} \qquad (8)\\
&= \frac{1}{1-p} \left[\sum_{0 \leq l < t} p^{A_l} + \frac{1}{1-p^M} \sum_{t \leq l < t+i} p^{A_l} \right]\\
&= \frac{\sum_{0 \leq l < t+i} p^{A_l} - \sum_{0 \leq l < t} p^{A_l + M}}{1 - p - p^M + p^{M+1}}.
\end{aligned}$$

How large can M be? As seen before $t+i \leq 3B^2(p)$ and $E(l) \leq 2B(p)$. Therefore $M \leq 6B^3(p)$. By definition $A_l \leq 2B(p)l$. Combining the above yields that

[5] $F(p)$ is a *rational* function of p if $F(p) = \frac{P(p)}{Q(p)}$ where both $P(p)$ and $Q(p)$ are polynomials in p.

the degrees of both the numerators and denominators are upper-bounded by $12B^3(p)$. $\square$

Now, note that since $B(p)$ is nondecreasing in p, the number of cyclic trees that could possibly be optimal trees for any $p' \in (0,p]$ is bounded by a (huge but) finite 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 ($12B^3(p)$) 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. Combining the above shows 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 2.6 *(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 1.1) 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 will be discussed later, can be extended to many other generalizations).

We point out though that Theorem 2.6, being more general, is weaker than Gallager and Voorhis' result. Their theorem, also implies that in the standard Huffman case there is a $1-1$ correspondence between optimal trees and intervals. That is, each distinct optimal tree is optimal for exactly one interval. Theorem 2.6 does not require this. It could be the case that some tree T is optimal for all p in two disjoint intervals (p_1, q_1) and (p_3, q_3) with $q_1 \le p_3$ but some other tree T' is optimal for interval (p_2, q_2) with $q_1 \le p_2 \le q_2 \le p_3$. It remains open as to whether, in the generalized Huffman problems we are examining, it will always be true that, if T is optimal for both p and p', then T is optimal for all $p'' \in [p, p']$.

2.4 An Algorithm for Finding Optimal (1,2)-Lopsided Trees

In this section we will present an algorithm that, given p, finds an optimal $(1,2)$-lopsided tree for the geometric distribution $\mathcal{P}_p$. The idea, will be to generalize the top-down algorithm developed in [12] that found a min-cost lopsided tree for a finite source by finding a min-cost path in a finite directed graph.

Before discussing the algorithm we introduce the graph upon which we will work.

Definition 2.8 *(See Figure 10)*
$G = (V, E)$ is an infinite weighted graph with

$$V = \{(m; e, c) \; : \; m, e, c \text{ integers} \ge 0\};$$

Each vertex $(m; e, c)$ has $e+1$ directed edges leaving it. The edges in E leaving vertex $(m; e, c)$ are

$$((m; e, c),\ (m + e - q; c + q, q)), \quad 0 \le q \le e \tag{9}$$

The weight or cost of all *of the edges leaving* $(m; e, c)$ *will be identically*

$$Cost\Big((m; e, c),\ (m + e - q; c + q, q)\Big) = \frac{p^m}{1 - p}$$

The weight or cost of a path will be the sum of the weight of the edges on the path.

The vertices in this graph correspond to the description of potential levels in a tree; edge (u, v) corresponds to level v immediately following level u in the tree. This will create a $1 - 1$ correspondence between trees and paths in G starting from $s_0 = (0; 1, 1)$; 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*.

We can now define our correspondence:

Definition 2.9 *Let* $T = \{(I_l, C_l, E_l)\}_{l=0}^{\infty}$ *be a tree with* $(I_0, C_0, E_0) = (1, 0, 0)$. *Set* $A_l(T) = \sum_{j \le l} E_j(T)$. *The* path in G associated with T *will be* $P(T) = \{(m_l; e_l, c_l)\}_{l=1}^{\infty}$ *with*

$$(m_l; e_l, c_l) = \begin{cases} (0; 1, 1) & \text{if } l = 1 \\ (A_{l-1}; I_l + E_l, C_l) & \text{if } l > 1 \end{cases}$$

In the other direction, given path $P = \{(m_l; e_l, c_l)\}_{l=1}^{\infty}$ *in* G *with* $(m_1; e_1, c_1) = (0; 1, 1)$, *the* tree associated with P *will be* $T = \{(I_l, C_l, E_l)\}_{l=0}^{\infty}$ *with*

$$(I_l, C_l, E_l) = \begin{cases} (1, 0, 0) & \text{if } l = 0 \\ (c_{l+1}, c_l, e_l - c_{l+1}) & \text{if } l > 0 \end{cases}$$

It is not difficult to see that $P(T)$ and $T(P)$ are well defined. Furthermore, they are reversible and maintain cost.

Lemma 2.7 *Let T and P be, respectively, an infinite tree and an infinite path. Then*

$$T(P(T)) = T \quad \text{and} \quad P(T(P)) = P$$

and

$$Cost(P(T)) = Cost(T). \tag{10}$$

Proof. $T(P(T)) = T$ and $P(T(P)) = P$ follow almost directly from the definitions. To prove (10) note that, from Lemma 2.1,

$$\mathrm{Cost}(P(T)) = \sum_{1 \ge l} \frac{p^{m_l}}{1 - p} = \frac{1}{1 - p} \sum_{0 \le l} p^{A_l(T)} = \mathrm{Cost}(T). \tag{11}$$

□

The problem of trying to find an optimal (minimum-cost) tree is therefore equivalent to finding an optimal (minimum-cost) infinite path in G that starts at node $s_0 = (0; 1, 1)$.[6]

An immediate implication of the 1-1 correspondence between infinite trees and paths in G is that some optimal paths must cycle on the (e, c) values.

Corollary 2.8 *Let $P = \{(m_l; e_l, c_l)\}_{l=1}^{\infty}$ be an optimal path. Let (t, i) be a pair such that $(e_t, c_t) = (e_{t+i}, c_{t+i})$ and $m^* = m_{t+i} - m_t$. Define*

$$\begin{aligned} \mathit{Cycle}(P, t, t+i) = &\{(m_1; e_1, c_1), ..., (m_{t-1}; e_{t-1}, c_{t-1})\} \cdot \\ &\prod_{k=0}^{\infty} \{(m_t + km^*; e_t, c_t), ..., (m_{t+i-1} + km^*; e_{t+i-1}, c_{t+i-1})\} \end{aligned}$$

where $\prod$ denotes a sequence of concatenations. Then Cycle$(P, t, t+i)$ is also optimal (as a path).

Proof. Let $T = T(P) = \{(I_l, C_l, E_l)\}_{l=0}^{\infty}$ be the optimal tree corresponding to P. Since $(e_t, c_t) = (e_{t+i}, c_{t+i})$, we also have $(I_t + E_t, C_t) = (I_{t+i} + E_{t+i}, C_{t+i})$. By Lemma 2.4, $T' = \text{Cycle}(T, t, t+i)$ is also optimal. Observe that Cycle$(T, t, t+i)$ can be written as

$$\begin{aligned} \text{Cycle}(T, t, t+i) = &\{(I_0, C_0, E_0), ..., (I_{t-1}, C_{t-1}, E_{t-1})\} \cdot \\ &\prod_{k=0}^{\infty} \{(I_t, C_t, E_t), ..., (I_{t+i-1}, C_{t+i-1}, E_{t+i-1})\} \end{aligned}$$

The corresponding path will be

$$\begin{aligned} P' = &\{(0; 1, 1), ..., (A_{t-2}; I_{t-1} + E_{t-1}, C_{t-1})\} \cdot \\ &\prod_{k=0}^{\infty} \{(A_{t-1} + k(A_{t+i} - A_t); I_t + E_t, C_t), ... \\ &\qquad ..., (A_{t+i-1} + k(A_{t+i} - A_t); I_{t+i-1} + E_{t+i-1}, C_{t+i-1})\} \end{aligned}$$

which is exactly equal to Cycle$(P, t, t+i)$. Furthermore, from (10),

$$\text{Cost}(P) = \text{Cost}(T) = \text{Cost}(\text{Cycle}(T, t, t+i)) = \text{Cost}(\text{Cycle}(P, t, t+i)). \quad (12)$$

Since P is optimal, Cycle$(P, t, t+i)$ must then also be optimal. $\square$

Next observe that Lemma 2.3 permits us to restrict the vertices V of G by requiring $e \le 3B(p)$ and $c \le B(p)$. Because of this, every path of length $3B^2(p)$ or more, must contain some repeated (e, c) values in its first $3B^2(p)$ nodes. Corollary 2.8 then implies the existence of a minimum-cost infinite path in G (starting at node s_0) which is cyclic and whose first cycle repeats within its first $3B^2(p)$ nodes; we can therefore restrict ourselves to finding such a path.

Corollary 2.8 further permits us to write the cost of an optimal infinite path in terms of its finite cyclic prefix:

[6] In what follows we will usually omit the phrase "starting at node s_0". All paths, unless otherwise specifically noted, are assumed to start at s_0.

Lemma 2.9 *Let P be any path starting at s_0, where $u_i = (m_i; e_i, c_i)$ and $u_j = (m_j; e_j, c_j)$ are the i^{th} and j^{th} nodes on P and $(e_i, c_i) = (e_j, c_j)$ and $m_i > m_j$. Then*

$$Cost\big(Cycle(P, i, j)\big) = Cost(u_j) + \frac{Cost(u_i) - Cost(u_j)}{1 - p^{m_i - m_j}} \tag{13}$$

where $Cost(u)$ is the cost of the path on H from the path to u. Furthermore, if P is an optimal *infinite path then*

$$Cost(P) = Cost(u_j) + \frac{Cost(u_i) - Cost(u_j)}{1 - p^{m_i - m_j}}. \tag{14}$$

Proof. To derive (13) apply Lemma 2.1 to get

$$\begin{aligned}
\mathrm{Cost}\big(\mathrm{Cycle}(P, i, j)\big) &= \frac{1}{1-p} \sum_{0 \le l} p^{A_l} = \frac{1}{1-p} \sum_{1 \le l} p^{m_l} \\
&= \frac{1}{1-p} \left[\sum_{1 \le l \le j} p^{m_l} + \sum_{0 \le h} \sum_{j < l \le i} p^{m_l + h(m_i - m_j)} \right] \\
&= \mathrm{Cost}(u_j) + \frac{\mathrm{Cost}(u_i) - \mathrm{Cost}(u_j)}{1 - p^{m_i - m_j}}
\end{aligned} \tag{15}$$

(14) follows from Corollary 2.8 $\square$

Similar to the discussion in the previous section this immediately gives us an algorithm for finding an optimal infinite path. Simply examine *every* finite path of length $\le 3B^2(p)$, assume that it is the start of some infinite optimal path, use Lemma 2.9 to calculate its corresponding cost and then return the path found with minimum cost. Unfortunately, there will be exponential (in $B(p)$) such paths so this is not a feasible algorithm.

The difficulty is how to find the beginning part of a cyclic path without having to examine *all* cyclic paths. Considering the problem from a path perspective provides some new intuition, as encapsulated in the following lemma:

Lemma 2.10 *For any node $w \in V$ set $d(w)$ to be the cost of the shortest path from s_0 to w. Let P be an optimal infinite path. For node w on P let P_w be the path on P from s_0 to w and T_w be the path on P from w to infinity. Then*

$$Cost(P_w) = d(w). \tag{16}$$

Now let V_w be any shortest path from s_0 to w. Set $P' = V_w T_w$ to be the infinite path created by concatenating V_w with T_w. Then P' is also an optimal infinite path.

Proof. By definition

$$\mathrm{Cost}(P') = \mathrm{Cost}\,(V_w) + \mathrm{Cost}(T_w) = \mathrm{Cost}\,(P_w) + \mathrm{Cost}(T_w) = \mathrm{Cost}(P).$$

Since P is optimal we must have that $\mathrm{Cost}(P') = \mathrm{Cost}(P)$ and therefore $\mathrm{Cost}(P_w) = \mathrm{Cost}(V_w) = d(w)$. $\square$

This lemma can be read as saying that if w is on *any* optimal infinite path and V_w is *any* shortest path from s_0 to w then V_w is the start of some optimal infinite path.

This intuitively suggests that a good way of trying to solve our problem might be to try to find *shortest paths* to every node u in the graph looking at all cycles generated along the way, and returning the minimum cost cycle found. A well-known efficient procedure for finding shortest paths in a graph is Dijkstra's shortest path algorithm [5]. Given a directed graph G and a source node s_0, Dijkstra's algorithm constructs a spanning tree S rooted at s_0 such that the unique path in S from s_0 to node u is a shortest path from s_0 to u in the graph. Our algorithm will therefore be to run a modified version of Dijkstra's shortest path algorithm. The remainder of this section is devoted to describing this algorithm and proving its correctness.

Our algorithm will keep the vertices of G partitioned into two sets S and Q. Associated with every vertex u will be another vertex $\texttt{PRED}[u]$; the edges $\{(\texttt{PRED}[u], u) : u \in S - \{s_0\}$ will form a spanning tree of S rooted at s_0.

The algorithm starts by setting $S = \{s_0\}$, $Q = V - \{s_0\}$ and assigning

$$\texttt{COST}[u] = \begin{cases} 0 & \text{if } u = s_0 = (0; 1, 1) \\ \frac{1}{1-p} & \text{if } (s_0, u) \in E \\ \infty & \text{otherwise} \end{cases}, \quad \texttt{PRED}[u] = \begin{cases} \emptyset & \text{if } u = s_0 = (0; 1, 1) \\ s_0 & \text{if } (s_0, u) \in E \\ \emptyset & \text{otherwise} \end{cases}$$

It also keeps a variable C which stores the minimum cost among all infinite trees seen so far and a candidate optimum path which has cost C. In the beginning $C = \infty$ and no path is stored. The algorithm then repeats the following until Q is empty or all nodes remaining in Q have infinite cost.

1. Find $u = (m; e, c) \in Q$ with minimum $\texttt{COST}[u]$.

2. Add u and the edge ($\texttt{PRED}[u]$, u) to S.

3. Examine the unique path from s_0 to v in S to see if it contains a node $w = (m'; e, c)$. If it does contain such a w we will call u a *cycle-node*.

   - If u is not a cycle-node then perform the standard Dijkstra's algorithm operation by updating $\texttt{COST}[u']$ for all $u' \in Q$ for which the edge (u, u') exists.
     That is, for all such u', if $\texttt{COST}[u'] < \texttt{COST}[u] + \text{Cost}(u, u')$ then
     - Set $\texttt{COST}[u'] = \texttt{COST}[u] + \text{Cost}(u, u')$.
     - Set $\texttt{PRED}[u'] = u$.
   - If u is a cycle-node then use Lemma 2.9 to calculate the cost, C' of the infinite optimal cyclic path for which the unique path from s_0 to u in S would be a prefix.
     If $C' < C$ make this new infinite path the new candidate optimum and set $C = C'$.
     Do not update $\texttt{COST}[u']$ for the $u' \in Q$ for which the edge (u, u') exists

When the algorithm concludes, the candidate optimum infinite path is returned as the real optimum infinite path. Figure 9 gives pseudocode implementing the algorithm, and figure 10 shows the directed acyclic graph built by the algorithm for $p = 0.65$.

Note that even though the algorithm is being run on an infinite graph it must terminate in time polynomial in $B(p)$. This is because, any path in G longer than $3B^2(p)$ edges will contain some repeated (e, c) value *inside the path.* Since cycle-nodes must be leaves in S, S can not contain any path longer than $3B^2(p)$ Furthermore, given any path $\{(m_l; e_l, c_l)\}_{l=1}^n$ in G we have $m_n \leq \sum_{l=1}^n e_l \leq n \cdot 3B(p)$. Since $n \leq 3B^2(p)$ this implies $m_n \leq 9B^3(p)$ so the total number of possible nodes u that can be added to S is at most $27B^5(p)$.

We will also need the following observation concerning the algorithm.

Lemma 2.11 *Let $v_1, v_2, \ldots, v_m$ be the nodes of G that are added to S, indexed by the step at which they are added to S. Let $a(v)$ be the value of* $\texttt{COST}[v]$ *at the time v is added to S. Then*

$$\forall i,\ a(v_i) \leq a(v_{i+1}),$$

Proof. Consider the values of $\texttt{COST}[v_i]$ and $\texttt{COST}[v_{i+1}]$ at time step i, immediately before v_i is chosen to be added to S. Since v_i is chosen to be added we must have $a(v_i) = \texttt{COST}[v_i] \leq \texttt{COST}[v_{i+1}]$. After time step i is completed either $\texttt{COST}[v_{i+1}]$ remains unchanged or $\texttt{COST}[v_{i+1}] = a(v_i) + \mathrm{Cost}(v_i, v_{i+1})$. In either case, immediately after time step i, $a(v_{i+1}) = \texttt{COST}[v_{i+1}] \geq a(v_i)$ so we are done. □

We can now prove our main result:

Theorem 2.12 *When the algorithm described above (and in Figure 9) terminates the candidate optimum path returned is an actual optimal infinite path in G.*

Proof. To prove that the algorithm gives the correct answer we must show that it finds an optimal infinite path. We start by pointing out that even though our algorithm is similar to Dijkstra's it differs in that some paths in S might not be shortest paths. This can happen because some real shortest path to a node u' might contain a cycle-node u. Since edges leaving cycle-nodes are ignored the shortest path to u' will not be found. We therefore can not use the standard proof of the correctness of Dijkstra's algorithm but, instead, must prove correctness from scratch.

Let S be the spanning tree rooted at s_0 found by the algorithm. We say a node v is *good* if the path to v on S is a real shortest path. We say that v is *bad* if either the path to v on S is not a shortest path or v does not appear in S.

Consider some arbitrary infinite optimal path P. Let u and u' be two consecutive nodes on P such that u is a good node and u' is *bad.* (See Figure 11.) We can always find such u, u' because the root is good, and the number of good nodes is finite. Let P_u, $P_{u'}$, respectively, be the paths on P from s_0 to u and u'. Then, from Lemma 2.10

$$\mathrm{Cost}(P_u) = d(u) \quad \text{and} \quad \mathrm{Cost}(P_{u'}) = d(u').$$

Input: p

1. **Initialization**
Set $\texttt{COST}[m;e,c] := \infty$ and $\texttt{PRED}[m;e,c] := \emptyset$ for all entries
  with $m \leq 9B^3(p)$, $e \leq 3B(p)$ and $c \leq B(p)$.
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 $(m;e,c)$ stored in the $\texttt{PRED}[\cdot;\cdot,\cdot]$ pointers **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**
        $\texttt{COST}[m';e',c'] := \texttt{new\_cost}$.
        $\texttt{PRED}[m';e',c'] := [m;e,c]$.
      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 9: The algorithm for finding optimal infinite $(1,2)$-lopsided trees.

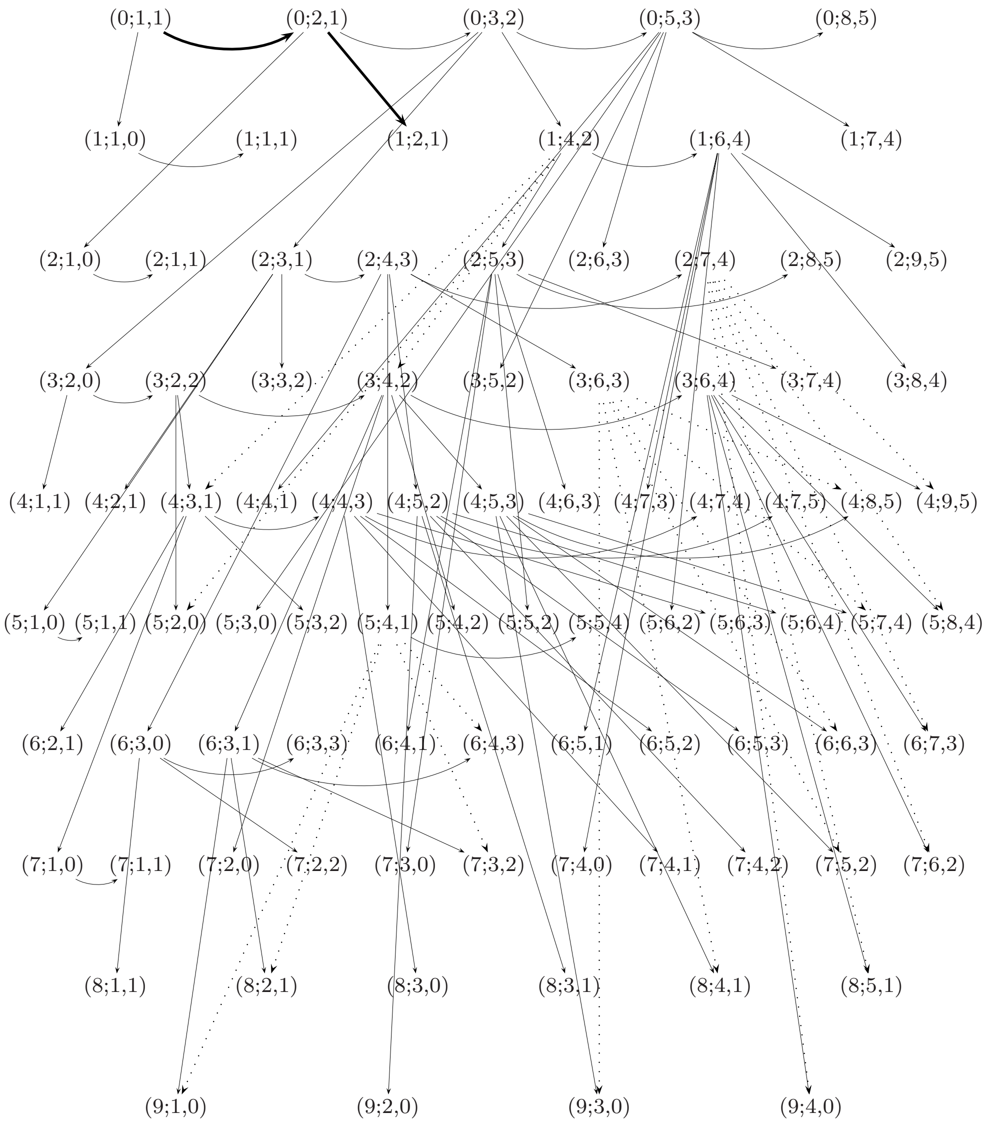


Figure 10: The DAG built by the algorithm for $p = 0.65$, where thick arrows indicate the prefix of the optimal infinite path, thin arrows indicate path tree edges, and dotted arrows indicate non-path tree edges. Although not written down, edges $\Big((m;e,c),\,(m+e-q;c+q,q)\Big)$ have cost $\frac{p^m}{1-p}$.

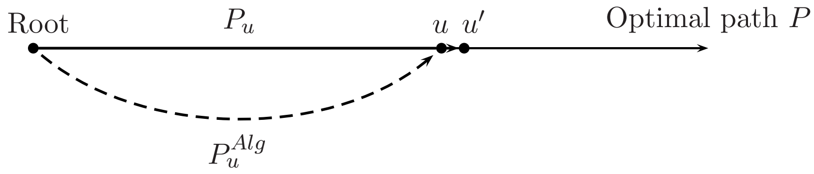


Figure 11: P_u is a subpath from root to u on optimal path P. P_u^{Alg} is the path to u found by the algorithm.

In particular, since $\mathrm{Cost}(P_{u'}) = \mathrm{Cost}(P_u) + \mathrm{Cost}(u, u')$ we have that

$$d(u') = d(u) + \mathrm{Cost}(u, u').$$

Since u is good, there is a shortest path P_u^{Alg} from s_0 to u that lies in S. Note that P_u^{Alg} is not necessarily the same as P_u but

$$\mathrm{Cost}\,(P_u) = d(u) = \mathrm{Cost}\left(P_u^{Alg}\right).$$

Let $P' = P_u^{Alg} T_u$ be the new infinite path created by concatenating P_u^{Alg} with T_u. Lemma 2.10 tells us that P' is also optimal.

Therefore P_u^{Alg}, the path to u found by the algorithm, is the start of optimal infinite path P'. Thus, by Theorem 2.4, if u was a cycle node, then the cyclic infinite path found by the algorithm when adding u to S would be optimum. Therefore, to prove correctness of the algorithm, it only remains to prove that u is a cycle node.

Suppose by contradiction, that u was not a cyclic node.

As in Lemma 2.11 let $a(v)$ be the value of $\mathtt{COST}[v]$ at the time v is added to S. Since $a(v)$ is the length of some path from s_0 to v we have $a(v) \geq d(v)$ for all v. Therefore

$$a(u) = d(u) < d(u) + \mathrm{Cost}(u, u') = d(u') \leq a(u').$$

We therefore see that u' can not have been added to S *before* u since, if it had been, then Lemma 2.11 says that $a(u') \leq a(u)$ leading to a contradiction.

But if u is not a cyclic-node and u' is still in Q at the time that u is processed then $\mathtt{COST}[u']$ is set to be the minimum of $\mathtt{COST}[u']$ and $d(u') = \mathtt{COST}[u] + \mathrm{Cost}(u, u')$. Since $\mathtt{COST}[u']$ can never be reduced lower than $d(u')$ this means that node u' must at some point be taken off of Q and added to S with value $\mathtt{COST}[u] = d[u']$. But this means that u' is good, contradicting the choice of u and u'. u must therefore be a cyclic-node and the proof is complete. $\square$

We have implemented the algorithm, constructing optimal $(1, 2)$-lopsided trees for all distributions $\mathcal{P}_p$ where p is of the form $(0.001) * k$, $k = 1, 2, \ldots, 920$. The output, presented both in path and tree form, can be found in the appendix.

3 Algorithms for $(\gamma_1, \gamma_2, \ldots, \gamma_r)$-Lopsided Trees

In this section we describe how the the technique of the previous section can be generalized to construct algorithms for finding optimal $(\gamma_1, \gamma_2, \ldots, \gamma_r)$-lopsided trees, where $\gamma_1 \le \gamma_2 \le \ldots \le \gamma_r$ are integers. We first generalize the combinatorial properties of 2.2 and 2.3 and then the algorithm of 2.4. Since most of what is done involves straightforward generalizations of the technique developed in the previous section we only state the differences between this section and the last without proving most of the generalized results.

Definition 3.1 *Given $\vec{\gamma} = (\gamma_1, \gamma_2, \ldots, \gamma_r)$, define $\vec{d}$ to be a vector of non-negative integers such that d_k is the number of γ's in $\vec{\gamma}$ with values equal k. For example, if $\vec{\gamma} = (2,3,3,5)$ then $\vec{d} = (0,1,2,0,1)$.*

Definition 3.2 *(See Figures 12 and 13)*
An $(\gamma_1, \gamma_2, \ldots, \gamma_r)$-lopsided forest

$$F = \{(I_l, E_l^0, E_l^1, ..., E_l^{\gamma_r - 1})\}_{l=0}^{\infty}$$

is an infinite sequence of tuples of non-negative integers satisfying

$$\forall i, l \qquad \begin{aligned} E_{l+1}^i &= E_l^{i+1} + I_l^{i+1} \cdot d_{i+1} \\ E_l^0 + I_l &= \sum_{k=1}^{r} I_{l-\gamma_k} \end{aligned} \tag{17}$$

where $I_l = 0$ and $E_l^i = 0$ when $l < 0$. Note that this equation simply states that, in the tree considered, all children of every internal node are in the tree (which is obviously always the case in optimal infinite trees).

An $(\gamma_1, \gamma_2, \ldots, \gamma_r)$-lopsided tree T is an $(\gamma_1, \gamma_2, \ldots, \gamma_r)$-lopsided forest satisfying

$$(I_0, E_0^0, E_0^1, ..., E_0^{\gamma_r - 1}) = (1, 0, 0, ..., 0).$$

Two levels $(I_i, E_i^0, E_i^1, ..., E_i^{\gamma_r - 1})$ and $(I_j, E_j^0, E_j^1, ..., E_j^{\gamma_r - 1})$ are compatible if

$$I_i + E_i^0 = I_j + E_j^0 \quad \text{and} \quad \forall k > 0, E_i^k = E_j^k.$$

Intuitively, for $k > 0$, E_l^k counts the number of tree edges that pass through level l and terminate in nodes on level $l + k$.

Similarly, we can define combinatorial operations on $(\gamma_1, \gamma_2, \ldots, \gamma_r)$-lopsided forests. Let $L_l = (I_l, E_l^0, E_l^1, ..., E_l^{\gamma_r - 1})$. In what follows we assume that

$$F = \{L_0, L_1, L_2, L_3, ...\} \text{ and } F' = \{L'_0, L'_1, L'_2, L'_3, ...\}.$$

Definition 3.3 *Let $i, j \ge 0$ be integers.*

$$\begin{aligned} F^{<i>} &= \{(I_0, i, E_0^1, ..., E_0^{\gamma_r - 1}), L_1, L_2, L_3, ...\} \\ \textit{Truncate}(F, j) &= \{L_j, L_{j+1}, L_{j+2}, L_{j+3}, ...\} \end{aligned}$$

Suppose L'_0 and L_j are compatible, then

$$\textit{Replace}(F, F', j) = \{L_0, L_1, L_2, ..., L_{j-1}, L'_0, L'_1, L'_2, L'_3, ...\}$$

Suppose, for $i < j$, L_i and L_j are compatible, then

$$\begin{aligned} \textit{Repeat}(F, i, j) &= \{L_0, L_1, ..., L_i, ..., L_{j-1}, L_i, ..., L_{j-1}, L_j, L_{j+1}, ...\} \\ \textit{Cycle}(F, i, j) &= \{L_0, L_1, ..., L_i, ..., L_{j-1}, L_i, ..., L_{j-1}, L_i, ..., L_{j-1}, ...\} \end{aligned}$$

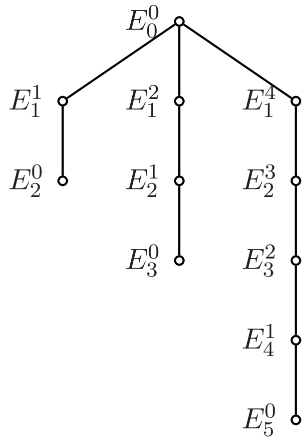
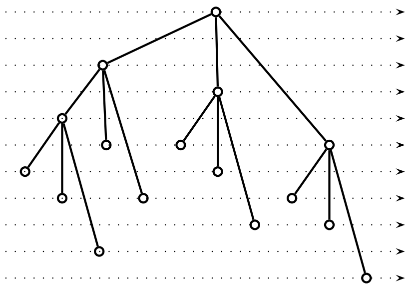


Figure 12: A $(2,3,5)$ tree. Each node contributes 1 unit on the counter next to it.



I_l	E_l^0	E_l^1	E_l^2	E_l^3	E_l^4
1	0	0	0	0	0
0	0	1	1	0	1
1	0	1	0	1	0
1	0	1	2	0	1
1	0	3	1	1	1
1	2	2	2	1	1
0	2	3	2	1	1
0	3	2	1	1	0
0	2	1	1	0	0
0	1	1	0	0	0
0	1	0	0	0	0

Figure 13: A $(2,3,5)$-lopsided tree and the associated tuples.

The next lemma states that with these new generalized definitions many of these operations still preserve optimality of forests. The proof of this lemma is almost exactly the same as that of Lemma 2.2,

Lemma 3.1 *If F and F' are optimal for p then*

1. *$\forall i \le E_0^0$, $F^{<i>}$ is optimal.*
2. *$\forall j \ge 0$, Truncate(F, j) is optimal.*
3. *If L'_0 and L_j are compatible, then Replace(F, F', j) is optimal.*
4. *If L_i and L_j are compatible, then Repeat(F, i, j) is optimal.*

We also define the following new operations in order to permit describing a special forest which will be used to prove Lemma 3.2.

Definition 3.4 *(Superposition of forests)*

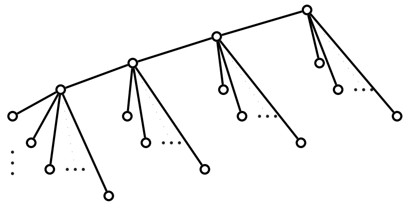


Figure 14: The tree H described in Definition 3.5.

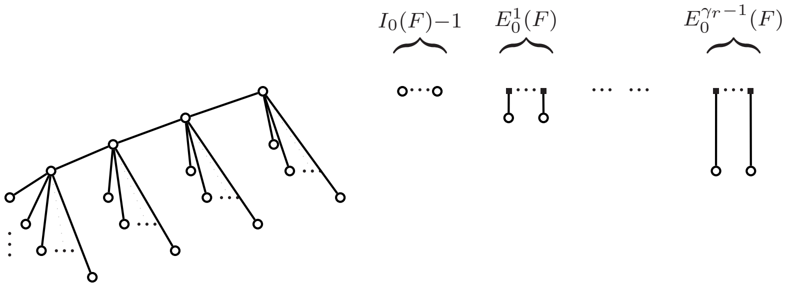


Figure 15: F^* defined in the proof of Lemma 3.2.

Let L, L' be two levels, F, F' two forests and k a nonnegative integer. Then

$$\begin{aligned} L + L' &= (I + I', E^0 + E^{0'}, E^1 + E^{1'}, ..., E^{\gamma_r - 1} + E^{\gamma_r - 1'}) \\ F + F' &= \{L_0 + L'_0, L_1 + L'_1, L_2 + L'_2, ...\} \\ k \times F &= \underbrace{F + F + ... + F}_{k \text{ times}} \end{aligned}$$

Definition 3.5 *(Special forests) For k, $0 \le k \le \gamma_r$ define forest f_k to be the path of length k terminating with one leaf at level k i.e., $\forall l \ge 0$, $I_l(f_k) = 0$ and*

$$\forall l \ge 0, \forall j \le \gamma_r, \quad E_l^j(f_k) = \begin{cases} 1 & j = k - l \\ 0 & \textit{otherwise.} \end{cases}$$

Let H be the unique tree with $(I_0, E_0^0, E_0^1, ..., E_0^{\gamma_r - 1}) = (1, 0, 0, ..., 0)$ and, $\forall t > 0$

$$I_t = \begin{cases} 1 & \textit{if } t \bmod \gamma_1 = 0 \\ 0 & \textit{otherwise} \end{cases}$$

Seen another way, H is recursively defined so that the child hanging off (one of) its smallest edge(s) is the root of another H while the children hanging off all of its other edges are leaves. See Figure 14.

The following lemma generalizes Lemma 2.3 for $(\gamma_1, \gamma_2, \ldots, \gamma_r)$-lopsided forests.

Lemma 3.2 *(Optimal $(\gamma_1, \gamma_2, \ldots, \gamma_r)$-lopsided forests have bounded width) Let p be fixed and $B(p) = \min\left\{k : p^k < \frac{1-p}{\gamma_r}\right\}$. If F is any optimal $(\gamma_1, \gamma_2, \ldots, \gamma_r)$-lopsided forest for $\mathcal{P}_p$, then $\forall l, I_l(F) \le B(p)$. Furthermore, $\forall l, E_l^0(F) \le rB(p)$.*

Proof. Suppose by contradiction that F is optimal and $I_l(F) > B(p)$ for some l. Without loss of generality we may assume that $l = 0$ and $E_l^0(F) = 0$. Otherwise, by Lemma 3.1, we may replace F by the optimal forest Truncate$(F, l)^{<0>}$. Let $L_0 = L_0(F) = (I_0, 0, E_0^1, ..., E_0^{\gamma_r - 1})$.

Since all leaves of F are below level 0, $\frac{1}{1-p} < \text{Cost}(F)$. Now (see Figure 15) let

$$F^* = H + (I_0 - 1) \times f_0 + \sum_{j=1}^{\gamma_r - 1} E_0^j \times f_j.$$

Notice that $L_0(F^*)$ is compatible to $L_0(F)$. But then

$$\begin{aligned}
\text{Cost}(F^*) &\le \text{Cost}(H) \\
&\le p^{I_0 - 1} \sum_{i=0}^{\infty} (i\gamma_1 + \gamma_r) p^i \\
&< \frac{p^B \gamma_r}{(1-p)^2} < \frac{1}{1-p} < \text{Cost}(F)
\end{aligned}$$

contradicting the optimality of F. The second part of the lemma follows from the fact that $E_l^0 \le \sum_{k=1}^r I_{l-\gamma_k} \le rB(p)$. □

Using Lemma 3.2 we can prove that optimal $(\gamma_1, \gamma_2, \ldots, \gamma_r)$-lopsided tree must cycle (Theorem 3.3) and describe the structure of the solution space (Theorem 3.4). The proofs are almost exactly the same as those of Theorems 2.4 and 2.6 for the $(1, 2)$-lopsided case so we do not provide them here.

Theorem 3.3 *(Cycle Theorem) Let p be fixed and F be an optimal forest. Let (t, i) be a pair with $t + i \le (r + 1)B^2(p)$ such that $L_t(F)$ and $L_{t+i}(F)$ are compatible. Then $F' =$ Cycle$(F, t, t + i)$ is also an optimal forest for $\mathcal{P}_p$. Note that Lemma 3.2 implies that some t, i as described must exist.*

Theorem 3.4 *(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 before , we model our problem as finding an infinite shortest path in an infinite weighted graph. The graph is now given by

Definition 3.6 *$G = (V, E)$ is an infinite weighted graph with*

$$V = \{(m; e^0, e^1, ..., e^{\gamma_r - 1}) : m, e^0, ..., e^{\gamma_r - 1} \textit{ integers } \ge 0\};$$

The edges in E leaving vertex $(m; e^0, e^1, ..., e^{\gamma_r - 1})$ are

$$((m; e^0, e^1, ..., e^{\gamma_r - 1}), (m + e^0 - q; e^1, e^2, ..., e^{\gamma_r - 1}, 0) + q \cdot \delta)$$

with $0 \le q \le e^0$ and

$$\delta = (0; d_1, d_2, ..., d_r)$$

where d_k is the number of γ's with values equal k. These edges all have weigh $\frac{p^m}{1-p}$.

Using almost the same correspondence as in 2.4 we find that a minimum cost infinite path in G starting at $s_0 = (0; 1, 0, \ldots, 0)$ corresponds to an optimal tree. As in the $(1, 2)$-case, we can then use a modified version of Dijkstra's algorithm (similar to that in Figure 9 with the obvious modifications) to examine shortest paths in G starting at s_0, checking them for cycles and reporting the one whose cycle creates a minimum cost infinite tree.

As an illustration, we have implemented the algorithms for constructing optimal $(2, 3, 5)$, $(1, 3)$, $(2, 3)$ and $(3, 5)$-lopsided trees. The output, presented both in path and tree form, can be found in the appendix.

4 Generalizations of geometric distributions

In this section we will discuss how to construct optimal Huffman trees for what we will call $\mathcal{K}$-*offset* geometric distributions. These are the distributions $\mathcal{P}_{p,\alpha}$ given by equation (3):

$$\begin{aligned}\mathcal{P}_{p,\alpha} = p^{\alpha_0}, p^{\alpha_1}, p^{\alpha_2}, \ldots, p^{\alpha_{\mathcal{K}-1}},\\ p^{1+\alpha_0}, p^{1+\alpha_1}, p^{1+\alpha_2}, \ldots, p^{1+\alpha_{\mathcal{K}-1}},\\ p^{2+\alpha_0}, p^{2+\alpha_1}, p^{2+\alpha_2}, \ldots, p^{2+\alpha_{\mathcal{K}-1}},\\ p^{3+\alpha_0}, p^{3+\alpha_1}, p^{3+\alpha_2}, \ldots, p^{3+\alpha_{\mathcal{K}-1}}, \ldots\end{aligned}$$

where $\alpha = (\alpha_0, \alpha_1, \ldots, \alpha_{\mathcal{K}})$, $0 \le \alpha_0 \le \alpha_1 \le \alpha_2 \le \cdots \le \alpha_{\mathcal{K}-1} \le 1$. The case $\mathcal{K} = 2$ was fully dealt with by Merhav, Seroussi and Weinberger in [20] but there do not seem to be any previous results known for $\mathcal{K} \ge 3$.

Although the trees constructed here are the normal binary ones and it is the *distributions* that are different we will see that we will still be able to use a very similar technique as the one developed in Section 2 for constructing optimal $(1, 2)$-trees for the geometric distribution $\mathcal{P}_p$.

4.1 Introduction

Section 2 introduced some combinatorial operations on forests (Definition 2.7 and 3.3). Most of these operations preserved optimality (Lemmas 2.2 and 3.1) based on the fact that when the first k probabilities are removed from a geometric distribution $\mathcal{P}_p = \{p^i\}_{i=0}^{\infty}$, the resulting distribution $\mathcal{P}_p^{[k]} = \{p^i\}_{i=k}^{\infty}$ is still essentially a geometric distribution. In particular we used the fact that $\mathcal{P}_p^{[k]}$ is a *scaled* version of $\mathcal{P}$. To be more precise, we introduce the following definitions. Let $\mathcal{P} = \{p_i\}_{i=0}^{\infty}$.

Definition 4.1 *(Scaling) Given a real c, define $c \cdot \mathcal{P} = \{cp_i\}_{i=0}^{\infty}$.*

Definition 4.2 *(Truncating) For $k \ge 0$, define $\mathcal{P}^{[k]} = \{p_i\}_{i=k}^{\infty}$.*

First note the straightforward

Lemma 4.1 *F is optimal for some distribution $\mathcal{P}$ if and only if $\forall c > 0$, F is optimal for $c \cdot \mathcal{P}$.*

Proof.

$$\mathrm{Cost}(F, c\cdot\mathcal{P}) = \sum_{i=0}^{\infty} cp_i d_i(F) = c \cdot \sum_{i=0}^{\infty} p_i d_i(F) = c \cdot \mathrm{Cost}(F, \mathcal{P})$$

so $cost(F', c\cdot\mathcal{P}) < cost(F', c\cdot\mathcal{P})$ if and only if $cost(F', \mathcal{P}) < cost(F', \mathcal{P})$. $\square$

For geometric distribution $\mathcal{P}_p$ we have

$$\mathcal{P}_p^{[i]} = p^i \cdot \mathcal{P}_p. \tag{18}$$

Thus if F is optimal for $\mathcal{P}_p$ then $\mathrm{Truncate}(F, l)$ is optimal for $\mathcal{P}_p^{[A_l]}$ and therefore optimal for $\mathcal{P}_p$. It was this simple fact which underlay most of what we previously developed.

This no longer remains true for $\mathcal{P} = \mathcal{P}_{p,\alpha}$, though. The best that can be said is

$${\mathcal{P}_{p,\alpha}}^{[i\mathcal{K}+k]} = p^i \cdot {\mathcal{P}_{p,\alpha}}^{[k]}. \tag{19}$$

Thus, given $\mathcal{P}_{p,\alpha}$, we form $\mathcal{K}$ classes of distributions ${\mathcal{P}_{p,\alpha}}^{[0]}$,${\mathcal{P}_{p,\alpha}}^{[1]}$,...,${\mathcal{P}_{p,\alpha}}^{[\mathcal{K}-1]}$; F is then optimal for ${\mathcal{P}_{p,\alpha}}^{[k]}$ if and only if, $\forall i > 0$, F is optimal for ${\mathcal{P}_{p,\alpha}}^{[i\mathcal{K}+k]}$.

4.2 New Notation and Properties

We will need to modify our definition of forest so that its tuples encode, at each level, which of the $\mathcal{K}$ distributions is starting at that level.

Definition 4.3 *(See Figure 16) Given $\mathcal{P}_{p,\alpha}$, a* $\mathcal{K}$-offset forest

$$F = \{(I_l, E_l, K_l)\}_{l=0}^{\infty}$$

is an infinite sequence of tuples of non-negative integers satisfying

$$\forall l \geq 0, \qquad \begin{aligned} E_{l+1} &= 2I_l \\ K_{l+1} &= (K_l + E_l) \mod \mathcal{K} \end{aligned} \tag{20}$$

where $K_l \in \{0, 1, ..., (\mathcal{K}-1)\}$.

A $\mathcal{K}$-offset tree T for $\mathcal{P}_{p,\alpha}$ is a forest satisfying

$$(I_0, E_0, K_0) = (1, 0, 0).$$

Two levels (I_i, E_i, K_i) and (I_j, E_j, K_j) are compatible if

$$I_i + E_i = I_j + E_j \quad \textit{and} \quad K_i = K_j.$$

Intuitively, K_l indicates which class of distribution (${\mathcal{P}_{p,\alpha}}^{[K_l]}$) is assigned to the forest starting at level l. Therefore the cost of a forest F is only well defined with distribution ${\mathcal{P}_{p,\alpha}}^{[K_0(F)]}$.

Definition 4.4 *Define*

$$\mathit{Cost}(F) = \mathit{Cost}\left(F, {\mathcal{P}_{p,\alpha}}^{[K_0(F)]}\right).$$

A forest F is said to be optimal if, for all forests F' with $L_0(F')$ compatible with $L_0(F)$,

$$\mathit{Cost}(F) \leq \mathit{Cost}(F').$$

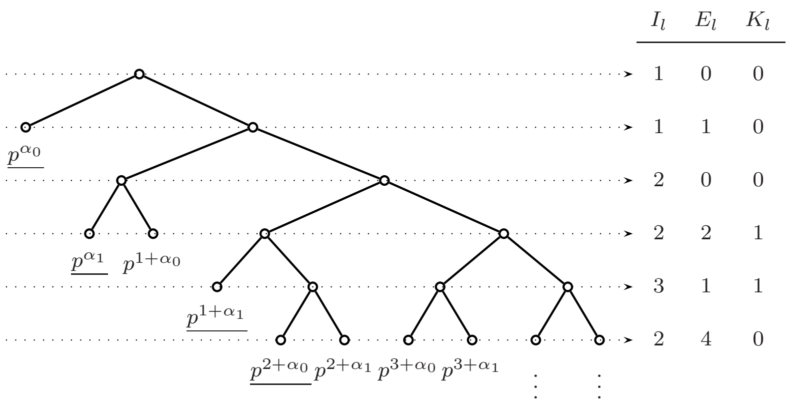


Figure 16: A $\mathcal{K}$-offset tree with $\mathcal{K} = 2$ and its associated tuples.

As before in the $(1,2)$-case, we can define combinatorial operations on $\mathcal{K}$-offset forests. Let $L_l = (I_l, E_l, K_l)$. In what follows we assume that

$$F = \{L_0, L_1, L_2, L_3, ...\} \text{ and } F' = \{L'_0, L'_1, L'_2, L'_3, ...\}.$$

Definition 4.5 *Let $i, j \geq 0$ be integers. Let $a \oplus_c b = (a + b) \mod c$.*

$$\textit{Truncate}(F, j) = \{L_j, L_{j+1}, L_{j+2}, L_{j+3}, ...\}$$
$$H^i_k = \{(1, i, k), (1, 1, k \oplus_{\mathcal{K}} i), (1, 1, k \oplus_{\mathcal{K}} (i + 1)), ...\}$$

Suppose L'_0 and L_j are compatible, then

$$\textit{Replace}(F, F', j) = \{L_0, L_1, L_2, ..., L_{j-1}, L'_0, L'_1, L'_2, L'_3, ...\}$$

Suppose, for $i < j$, L_i and L_j are compatible, then

$$\textit{Repeat}(F, i, j) = \{L_0, L_1, ..., L_i, ..., L_{j-1}, L_i, ..., L_{j-1}, L_j, L_{j+1}, ...\}$$
$$\textit{Cycle}(F, i, j) = \{L_0, L_1, ..., L_i, ..., L_{j-1}, L_i, ..., L_{j-1}, L_i, ..., L_{j-1}, ...\}$$

The following lemma, similar to Lemma 2.2, states that with these new generalized definitions many of these operations still preserve optimality of forests.

Lemma 4.2 *If F and F' are optimal for p then*

1. *$\forall j \geq 0$, Truncate(F, j) is optimal.*
2. *If L'_0 and L_j are compatible, then Replace(F, F', j) is optimal.*
3. *If L_i and L_j are compatible, then Repeat(F, i, j) is optimal.*
4. *If $i < j$ then Cost$(H^i_k) \geq$ Cost(H^j_k).*
5. *If $i < j$ then Cost$(H^k_i) \geq$ Cost(H^k_j).*

Proof. The proofs of (1),(2) and (3) are similar to the ones from Lemma 2.2 and 3.1 and therefore will be omitted. (4) holds since for all $n \geq 0$, $d_n(H^i_k) \geq d_n(H^j_k)$. (5) holds since for all $n \geq 0$, $p_n(\mathcal{P}_{p,\alpha}{}^{[i]}) \geq p_n(\mathcal{P}_{p,\alpha}{}^{[j]})$. $\square$

The next lemma generalizes Lemma 2.3 for $\mathcal{K}$-offset forests.

Lemma 4.3 *(Optimal $\mathcal{K}$-offset forests have bounded width)* *Let p be fixed and $B(p) = \min\left\{h : p^{\lfloor \frac{h}{\mathcal{K}} \rfloor - 2} < \frac{1-p}{\mathcal{K}}\right\}$. If F is any optimal $\mathcal{K}$-sided forest for $\mathcal{P}_{p,\alpha}$, then $\forall l, I_l(F) \leq B(p)$. Furthermore, $\forall l > 0, E_l(F) \leq 2B(p)$.*

Proof. Suppose by contradiction that F is optimal and $I_l(F) > B(p)$ for some l. Without loss of generality we may assume that $l = 0$ and $E_l(F) < \mathcal{K}$. Otherwise, by Lemma 4.2, we may replace F by the optimal forest $\text{Truncate}(F, l)^{<E_l \mod \mathcal{K}>}$. Since $d_i(F) \geq 0$ for $i < 2\mathcal{K} - K_0(F)$ and $d_i(F) \geq 1$ for $i \geq 2\mathcal{K} - K_0(F)$,

$$\begin{aligned}
\text{Cost}(F) &> \sum_{i=2}^{\infty} \sum_{k=0}^{\mathcal{K}-1} p^{\alpha_k + i} \\
&= \left(\sum_{i=2}^{\infty} p^i\right) \left(\sum_{k=0}^{\mathcal{K}-1} p^{\alpha_k}\right) = \frac{p^2 \sum_{k=0}^{\mathcal{K}-1} p^{\alpha_k}}{1-p}.
\end{aligned}$$

Let $F^* = H_{K_0(F)}^{I_0(F)-1+E_0(F)}$ where $L_0(F^*)$ and $L_0(F)$ are compatible. But then,

$$\begin{aligned}
\text{Cost}(F^*) &\leq \text{Cost}(H_0^{I_0(F)-1}) \\
&\leq \text{Cost}(H_0^{\mathcal{K}\lfloor \frac{I_0(F)-1}{\mathcal{K}} \rfloor}) \\
&= p^{\lfloor \frac{I_0(F)-1}{\mathcal{K}} \rfloor} \sum_{k=0}^{\mathcal{K}-1} \sum_{i=0}^{\infty} p^{i+\alpha_k} (i\mathcal{K} + k + 1) \\
&\leq \frac{\mathcal{K} p^{\lfloor \frac{I_0(F)-1}{\mathcal{K}} \rfloor} \sum_{k=0}^{\mathcal{K}-1} p^{\alpha_k}}{(1-p)^2} \\
&< \frac{\mathcal{K} p^{\lfloor \frac{B(p)}{\mathcal{K}} \rfloor} \sum_{k=0}^{\mathcal{K}-1} p^{\alpha_k}}{(1-p)^2} < \text{Cost}(F)
\end{aligned}$$

contradicting the optimality of F. The second part of the lemma follows from the fact that $E_l = 2I_{l-1} \leq 2B(p)$. □

Using Lemma 4.3 we can prove that optimal $\mathcal{K}$-offset trees must cycle (Theorem 4.4) and describe the structure of the solution space (Theorem 4.5). The proofs are almost exactly the same as those of Theorems 2.4 and 2.6 for the $(1, 2)$-lopsided case so we do not provide them here.

Theorem 4.4 *(Cycle Theorem) Let p be fixed and F be an optimal forest. Let (t, i) be a pair with $t + i \leq 3B^2(p)$ such that $L_t(F)$ and $L_{t+i}(F)$ are compatible. Then $F' = \text{Cycle}(F, t, t + i)$ is also an optimal forest for $\mathcal{P}_p$. Note that Lemma 4.3 implies that some t, i as described must exist.*

Theorem 4.5 *(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 before , we model our problem as finding an infinite shortest path in an infinite weighted graph. The graph is now given by

Definition 4.6 *$G = (V, E)$ is an infinite weighted graph with*

$$V = \{(m; e, k) : m, e, k \text{ integers } \geq 0, k < \mathcal{K}\};$$

The edges in E leaving vertex $(m; e, k)$ are

$$((m; e, k), (m + e - q; 2q, (k + e - q) \mod \mathcal{K}))$$

where $0 < q \leq e$.

The only new complication is in determining the weights of the edges. Working through the details gives:

Lemma 4.6 *If p is fixed then Cost$(T, \mathcal{P}_{p,\alpha})$ can be expressed as*

$$\begin{aligned}
&Cost(T, \mathcal{P}_{p,\alpha}) \\
&= \sum_{0 \leq k < \mathcal{K}} \sum_{0 \leq i} p^{i+\alpha_k} d_{(\mathcal{K}i+k)}(T) \\
&= \sum_{0 \leq k < \mathcal{K}} \sum_{0 \leq i} p^{i+\alpha_k} \min\{l : (\mathcal{K}i + k) < A_l(T)\} \\
&= \sum_{0 \leq k < \mathcal{K}} p^{\alpha_k} \sum_{0 \leq l} \sum_{A_l \leq \mathcal{K}j+k} p^j \\
&= \sum_{0 \leq l} \frac{1}{1-p} \sum_{0 \leq k < \mathcal{K}} p^{\alpha_k + \left\lceil \frac{A_l(T)-k}{\mathcal{K}} \right\rceil}
\end{aligned} \tag{21}$$

Therefore, we set the weight of all edges in G leaving $(m; e, k)$ to be

$$\frac{1}{1-p} \sum_{0 \leq k < \mathcal{K}} p^{\alpha_k + \lceil \frac{m-k}{\mathcal{K}} \rceil}.$$

Moreover, if $u_i = (m_i; e_i, k_i)$ and $u_j = (m_j; e_j, k_j)$ are two nodes on an infinite path P such that $(e_i, k_i) = (e_j, k_j)$ and $m_i > m_j$, then $\mathrm{Cost}(T, \mathcal{P}_{p,\alpha})$ can be recovered from the finite prefix of P as

$$\begin{aligned}
\mathrm{Cost}(T, \mathcal{P}_{p,\alpha}) &= \frac{1}{1-p} \left[\sum_{0 \leq l} \sum_{0 \leq k < \mathcal{K}} p^{\alpha_k + \left\lceil \frac{A_l(T)-k}{\mathcal{K}} \right\rceil} \right] \\
&= \frac{1}{1-p} \left[\sum_{0 \leq l \leq j} \sum_{0 \leq k < \mathcal{K}} p^{\alpha_k + \left\lceil \frac{m_l - k}{\mathcal{K}} \right\rceil} \right. \\
&\qquad \left. + \sum_{0 \leq h} \sum_{j < l \leq i} \sum_{0 \leq k < \mathcal{K}} p^{\alpha_k + \left\lceil \frac{m_l - k + h(m_i - m_j)}{\mathcal{K}} \right\rceil} \right] \\
&= \mathrm{Cost}(u_j) + \frac{\mathrm{Cost}(u_i) - \mathrm{Cost}(u_j)}{1 - p^{\frac{m_i - m_j}{\mathcal{K}}}}.
\end{aligned} \tag{22}$$

Note that the cyclic structure forces $\mathcal{K} | m_i - m_j$.

Using almost the same correspondence as in 2.4 we find that a minimum cost infinite path in G starting at $s_0 = (0; 1, 0)$ corresponds to an optimal tree. As in the $(1, 2)$-case, we can then use a modified version of Dijkstra's algorithm (similar to that in Figure 9 with the obvious modifications) to examine shortest paths in G starting at s_0, checking them for cycles and reporting the one whose cycle creates a minimum cost infinite tree. As usual, finding the shortest infinite path in G will get us the optimal tree.

We have implemented this algorithm for many 3-offset geometric distributions. The results can be found in the Appendix section.

4.3 More on two-offset geometric distributions

In [20] Merhav et al. give a description of optimal codes for the special two-offset geometric distributions $\mathcal{P}_{p,d}$ consisting of the infinite sequence

$$p_{x,d} = p^{|x+d|}, \quad x = 0, \pm 1, \pm 2, \pm 3, \ldots \tag{23}$$

where $0 < p < 1$ and $0 \leq d \leq 0.5$ is fixed. 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. In this subsection we use some simple transformations to show that their class of distributions is actually equivalent to the class we call the 2-offset geometric distributions.

Notice that for $x \geq 0$ the sequence (23) consists of $p^d, p^{1+d}, p^{2+d}, \cdots$, and for $x < 0$ it consists of $p^{1-d}, p^{2-d}, p^{3-d}, \cdots = p^{1-d}, p^{1+(1-d)}, p^{2+(1-d)}, \cdots$. Since $d \leq 0.5$, we have $d \leq 1 - d$. By setting $\alpha_0 = d$ and $\alpha_1 = 1 - d$ the distribution becomes

$$\mathcal{P}_{p,\alpha} = p^{\alpha_0}, p^{\alpha_1}, p^{1+\alpha_0}, p^{1+\alpha_1}, p^{2+\alpha_0}, p^{2+\alpha_1}, \ldots \tag{24}$$

with $0 \leq \alpha_0 \leq \alpha_1 \leq 1$ and $\alpha_0 + \alpha_1 = 1$. We can show that their results can actually be generalized to all values of α_0 and α_1. Given β_0 and β_1 where $0 \leq \beta_0 \leq \beta_1 \leq 1$, let $c = (1 - \beta_0 - \beta_1)/2$. Then

$$\begin{aligned} \mathcal{P}_{p,\beta} &= \{p^{i+\beta_0}, p^{i+\beta_1}\}_{i=0}^{\infty} \\ &= p^{-c} \cdot \{p^{i+\beta_0+c}, p^{i+\beta_1+c}\}_{i=0}^{\infty} \\ &= p^{-c} \cdot \{p^{i+\alpha_0}, p^{i+\alpha_1}\}_{i=0}^{\infty} \end{aligned} \tag{25}$$

where $\alpha_i = \beta_i + c$ and $\alpha_0 + \alpha_1 = 1$. By scaling (Def. 4.1), an optimal tree for $\mathcal{P}_{p,\alpha}$ is also an optimal tree for $\mathcal{P}_{p,\beta}$ (Lemma 4.1). Thus, the result from Merhav et al. actually provides optimal codes for *all* 2-offset distributions, not just the ones explicitly stated in their paper.

Using the same technique, we can show that given $\{\alpha_i\}_{i=0}^{\mathcal{K}-1}$ it is always possible to assume $\alpha_0 = 0$, since

$$\begin{aligned} \mathcal{P}_{p,\alpha} &= \{p^{i+\alpha_0}, p^{i+\alpha_1}, \ldots, p^{i+\alpha_{\mathcal{K}-1}}\}_{i=0}^{\infty} \\ &= p^{\alpha_0} \cdot \{p^i, p^{i+\alpha_1-\alpha_0}, \ldots, p^{i+\alpha_{\mathcal{K}-1}-\alpha_0}\}_{i=0}^{\infty} \\ &= p^{\alpha_0} \cdot \{p^i, p^{i+\beta_1}, \ldots, p^{i+\beta_{\mathcal{K}-1}}\}_{i=0}^{\infty} \\ &= p^{\alpha_0} \cdot \mathcal{P}_{p,\beta} \end{aligned} \tag{26}$$

where $\beta_i = \alpha_i - \alpha_0$ and $0 = \beta_0 \leq \beta_1 \leq ... \leq \beta_{\mathcal{K}-1} \leq 1$. Again, by scaling, an optimal tree for $\mathcal{P}_{p,\beta}$ is also an optimal tree for $\mathcal{P}_{p,\alpha}$. Therefore, we can always assume $\alpha_0 = 0$.

Because of this, we chose to implement our algorithm for 3-offset geometric distribution with $\alpha_0 = 0$. The results are shown in the Appendix.

5 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 generalizations of Huffman coding, e.g., the $(\gamma_1, \gamma_2, \ldots, \gamma_r)$-lopsided Trees modelling unequal cost letters, to which the previously known techniques could not be applied. This new technique also works for a larger set of generalizations of geometric sources than could be solved for previously, e.g., the $\mathcal{K}$-offset geometric distributions $\mathcal{P}_{p,\alpha}$ defined in equation (3).

We note that it is straightforward to combine the two problems and use the technique to develop an algorithm for constructing an optimal $(\gamma_1, \gamma_2, \ldots, \gamma_r)$-lopsided tree for a $\mathcal{P}_{p,\alpha}$ $\mathcal{K}$-offset geometric distribution. In particular, setting all of the $\gamma_i = 1$, would immediately specialize to an algorithm for constructing optimal infinite r-ary Huffman trees for $\mathcal{P}_{p,\alpha}$ $\mathcal{K}$-offset geometric distributions.

There are still many open questions remaining in this area. As an example, suppose the problem is modified to build an optimum binary prefix code C for $\mathcal{P}_p$ under the further constraint that *all words $w \in C$ must have the property that they end with a particular given pattern P but do not contain P anywhere else in w*. Such codes can still be modelled using a tree in which the codewords are the leaves of the tree. It is not hard to see that an optimal code tree for such a restriction will not satisfy the bounded width condition given by Lemma 2.3. Since our entire approach was based on Lemma 2.3 our techniques could not be applied here.

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]. As an example of a very simple such source consider

$$\mathcal{Q}_p = \{1, p, p, p^2, p^2, p^2, p^3, p^3, p^3, p^3, \ldots\},$$

i.e., the distribution in which p^i appears $i+1$ times. This is (a scaled version of) the distribution of $\{(X, Y)\}$ where X and Y are independent copies of $\mathcal{P}_p$. Again, it is not hard to see that optimal codes for $\mathcal{Q}_p$ will not satisfy the bounded width condition given by Lemma 2.3 and our techniques will therefore not work here as well.

Another, more technical problem, would be to improve the *Partition Theorem* (Theorem 2.6). 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 1.1) 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. In the appendix we present 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.

Acknowledgment: The first author would like to thank G. Seroussi for discussions concerning infinite Huffman codes.

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] T. H. Cormen, C. E. Leiserson, R. L. Rivest and C. Stein, *Introduction to Algorithms (2nd ed.),* The MIT Press (2001)

[6] Paul F. Dietz "Maintaining order in a linked list," *Proceedings of the 14th Annual ACM symposium on Theory of Computing,* (1982) 122 - 127.

[7] 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.

[8] 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.

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

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

[11] 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.

[12] 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.

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

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

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

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

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

[18] 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.

[19] 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.

[20] 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.

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

A Implementation Details and Results

To illustrate the technique developed in the paper we implemented the algorithm in various cases.

To implement the algorithms, e.g. the algorithm for finding optimal (1,2)-lopsided trees, we must be able, given a spanning tree S of a graph $G = (V, E)$ and a node u, find if the unique path on S from the root to u contains another node w with the same (e, c) label as u. Theoretically, this can be done in time $O(\log |V|)$ using sophisticated data structures from [6]. In our implementation, though, we simply did a naive scan, walking back the path and checking every node on the path for a cycle. Since our paths could not be longer than $3B^2(p)$ (Lemma 2.3), this worked well in practice.

Appendices A.1 to A.5 shows optimal $(1, 2)$, $(1, 3)$, $(2, 3)$, $(3, 5)$ and $(2, 3, 5)$ lopsided trees. In our implementations we ran the algorithms for all p starting from 0.001 with 0.001 increments. In each case we present a table: The *Interval* column lists entries in the form $\mathcal{P}_x - \mathcal{P}_y$. This denotes that the information in the associated row corresponds to all p on which the algorithm was run with $x \leq p < y$. The intervals presented were found by running the algorithm(s) for the given values of p; whenever a new $p = (0.001) * k$ created a new tree that was different from the tree previously created for $p = (0.001) * (k - 1)$, starting a new interval. The row information presents the optimal path and leaves per level for each $p = (0.001) * k$ in the given interval.

For ease of viewing a line is also presented, partitioned into the different intervals shown in the table.

Note that the results as presented seems to correspond to the statement of the Partition Theorem (Theorem 3.4). Recall that at the end of subsection 2.3 we state that Theorem 2.6 (or more generally, Theorem 3.4) does not necessarily impose a $1 - 1$ correspondence between optimal trees and intervals. However, from the results it seems quite plausible that a $1 - 1$ correspondence does exist.

Finally, in both cases, we give realizations of the first few trees. Note that for a given path expansion there are actually many (uncountable) ways to realize the trees; we chose ones that accentuated the cyclic structures.

B Optimal (1,2)-Lopsided Trees

Interval	Expansion	Leaves per level
$\mathcal{P}_{0.001}$ - $\mathcal{P}_{0.5}$	(0;1,1) (1;1,0) (1;1,1)	1 0
$\mathcal{P}_{0.5}$ - $\mathcal{P}_{0.683}$	(0;1,1) (0;2,1) (1;2,1)	0 1
$\mathcal{P}_{0.683}$ - $\mathcal{P}_{0.755}$	(0;1,1) (0;2,1) (0;3,2) (2;3,1) (3;3,2)	0 0 2 1
$\mathcal{P}_{0.755}$ - $\mathcal{P}_{0.809}$	(0;1,1) (0;2,1) (0;3,2) (1;4,2) (3;4,2)	0 0 1 2
$\mathcal{P}_{0.809}$ - $\mathcal{P}_{0.838}$	(0;1,1) (0;2,1) (0;3,2) (0;5,3) (3;5,2) (5;5,3)	0 0 0 3 2
$\mathcal{P}_{0.838}$ - $\mathcal{P}_{0.864}$	(0;1,1) (0;2,1) (0;3,2) (0;5,3) (2;6,3) (5;6,3)	0 0 0 2 3
$\mathcal{P}_{0.864}$ - $\mathcal{P}_{0.879}$	(0;1,1) (0;2,1) (0;3,2) (0;5,3) (1;7,4) (5;7,3) (8;7,4)	0 0 0 1 4 3
$\mathcal{P}_{0.879}$ - $\mathcal{P}_{0.887}$	(0;1,1) (0;2,1) (0;3,2) (0;5,3) (1;7,4) (4;8,4) (8;8,4)	0 0 0 1 3 4
$\mathcal{P}_{0.887}$ - $\mathcal{P}_{0.894}$	(0;1,1) (0;2,1) (0;3,2) (0;5,3) (0;8,5) (4;9,4) (9;8,4) (13;8,4)	0 0 0 0 4 5 4
$\mathcal{P}_{0.894}$ - $\mathcal{P}_{0.904}$	(0;1,1) (0;2,1) (0;3,2) (0;5,3) (0;8,5) (4;9,4) (8;9,5) (13;9,4)	0 0 0 0 4 4 5
$\mathcal{P}_{0.904}$ - $\mathcal{P}_{0.913}$	(0;1,1) (0;2,1) (0;3,2) (0;5,3) (0;8,5) (3;10,5) (8;10,5)	0 0 0 0 3 5
$\mathcal{P}_{0.913}$ - $\mathcal{P}_{0.92}$	(0;1,1) (0;2,1) (0;3,2) (0;5,3) (0;8,5) (2;11,6) (8;11,5) (13;11,6)	0 0 0 0 2 6 5
$\mathcal{P}_{0.92}$ - $\mathcal{P}_{0.923}$	(0;1,1) (0;2,1) (0;3,2) (0;5,3) (0;8,5) (2;11,6) (7;12,6) (13;12,6)	0 0 0 0 2 5 6
$\mathcal{P}_{0.923}$ - $\mathcal{P}_{0.927}$	(0;1,1) (0;2,1) (0;3,2) (0;5,3) (0;8,5) (1;12,7) (7;13,6) (14;12,6) (20;12,6)	0 0 0 0 1 6 7 6
$\mathcal{P}_{0.927}$ - $\mathcal{P}_{0.93}$	(0;1,1) (0;2,1) (0;3,2) (0;5,3) (0;8,5) (1;12,7) (7;13,6) (13;13,7) (20;13,6)	0 0 0 0 1 6 6 7
$\mathcal{P}_{0.93}$ - $\mathcal{P}_{0.931}$	(0;1,1) (0;2,1) (0;3,2) (0;5,3) (0;8,5) (0;13,8) (7;14,6) (14;13,7) (21;13,6) (27;13,7)	0 0 0 0 0 7 7 7 6
$\mathcal{P}_{0.931}$ - $\mathcal{P}_{?}$	(0;1,1) (0;2,1) (0;3,2) (0;5,3) (0;8,5) (0;13,8) (7;14,6) (14;13,7) (20;14,7) (27;14,7)	0 0 0 0 0 7 7 6 7

Table 1: (1,2)-lopsided trees: for each interval shown in the first column, the second column shows the optimal path for the p's sampled in that interval. Highlighted nodes are where level repetitions occur. The third column shows the number of leaves per level; highlighted numbers compose the cycle.

Figure 17: Sampled p's in the same interval share the same optimal tree.

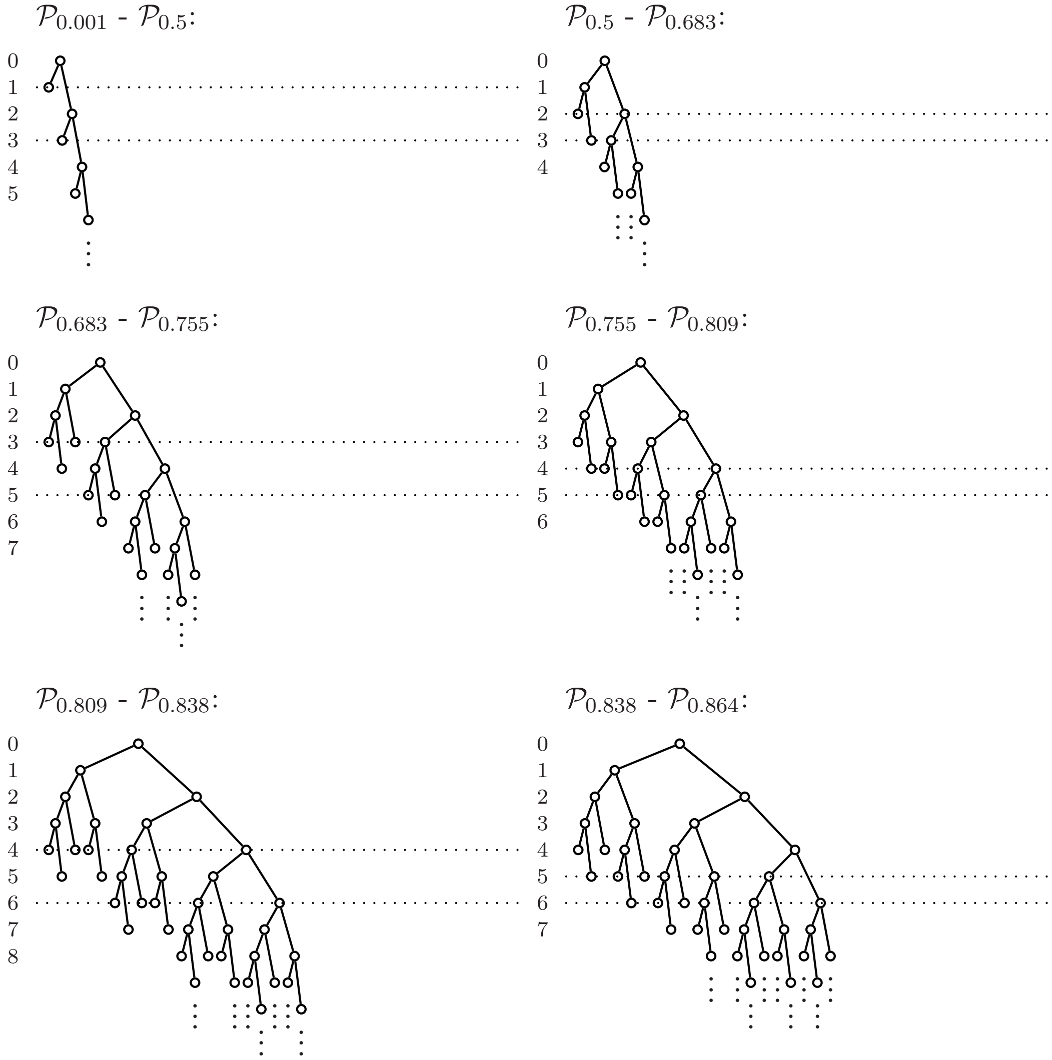
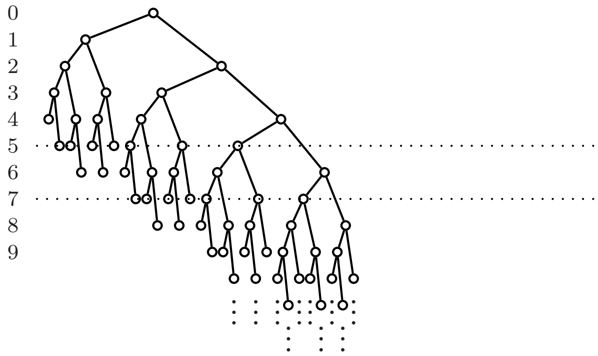
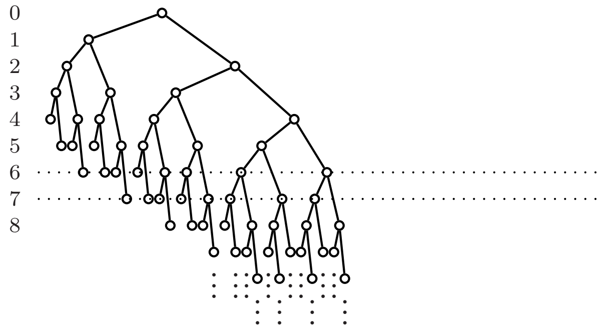


Figure 18: Optimal (1,2)-lopsided trees for different intervals.

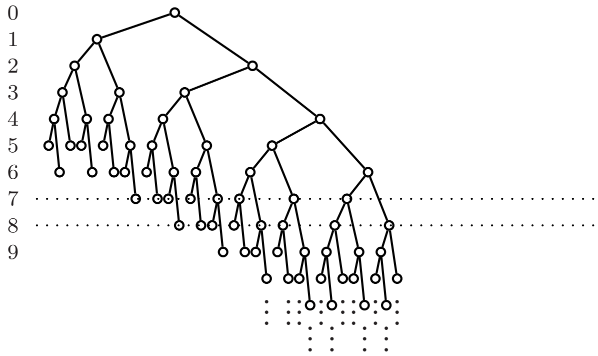
$\mathcal{P}_{0.864}$ - $\mathcal{P}_{0.879}$:



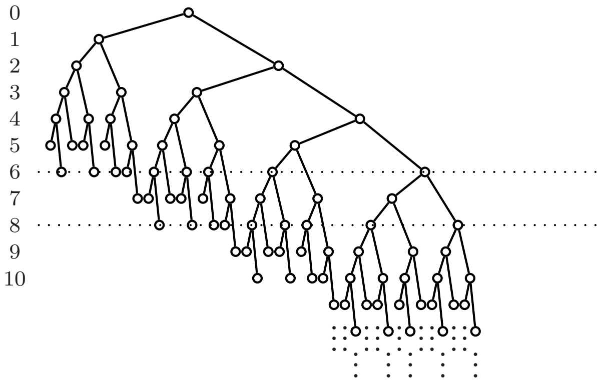
$\mathcal{P}_{0.879}$ - $\mathcal{P}_{0.887}$:



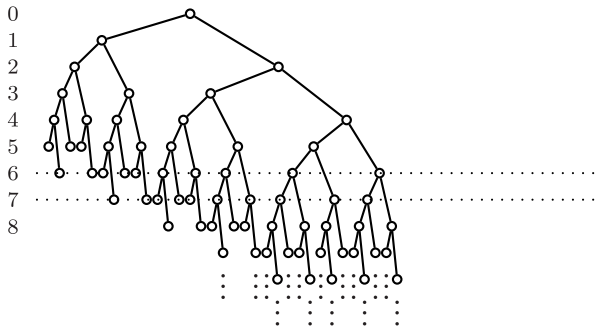
$\mathcal{P}_{0.887}$ - $\mathcal{P}_{0.894}$:



$\mathcal{P}_{0.894}$ - $\mathcal{P}_{0.904}$:



$\mathcal{P}_{0.904}$ - $\mathcal{P}_{0.913}$:



$\mathcal{P}_{0.913}$ - $\mathcal{P}_{0.92}$:

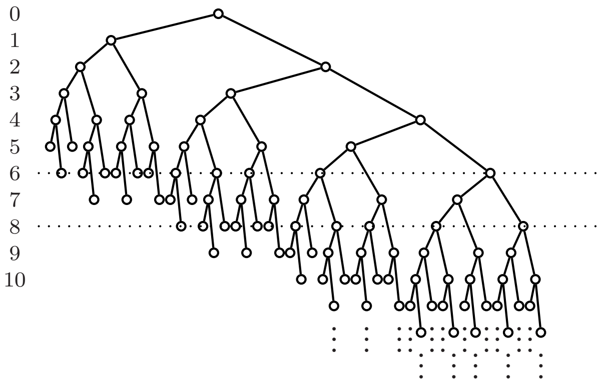


Figure 19: Optimal (1,2)-lopsided trees for different intervals (con't).

C Optimal (1,3)-Lopsided Trees

Interval	Expansion	Leaves per level
$\mathcal{P}_{0.001}$ - $\mathcal{P}_{0.415}$	(0;1,0,1) (1;0,1,0) (1;1,0,0) (1;1,0,1)	1 0 0
$\mathcal{P}_{0.415}$ - $\mathcal{P}_{0.5}$	(0;1,0,1) (0;1,1,1) (1;1,1,0) (2;1,0,0) (2;1,0,1)	0 1 1 0
$\mathcal{P}_{0.5}$ - $\mathcal{P}_{0.619}$	(0;1,0,1) (0;1,1,1) (1;1,1,0) (1;2,0,1) (2;1,1,1)	0 1 0 1
$\mathcal{P}_{0.619}$ - $\mathcal{P}_{0.725}$	(0;1,0,1) (0;1,1,1) (0;2,1,1) (1;2,1,1)	0 0 1
$\mathcal{P}_{0.725}$ - $\mathcal{P}_{0.767}$	(0;1,0,1) (0;1,1,1) (0;2,1,1) (0;3,1,2) (2;2,2,1) (3;3,1,1) (4;3,1,2)	0 0 0 2 1 1
$\mathcal{P}_{0.767}$ - $\mathcal{P}_{0.783}$	(0;1,0,1) (0;1,1,1) (0;2,1,1) (0;3,1,2) (1;3,2,2) (3;3,2,1) (5;3,1,1) (6;3,1,2)	0 0 0 1 2 2 1
$\mathcal{P}_{0.783}$ - $\mathcal{P}_{0.809}$	(0;1,0,1) (0;1,1,1) (0;2,1,1) (0;3,1,2) (1;3,2,2) (3;3,2,1) (4;4,1,2) (6;3,2,2)	0 0 0 1 2 1 2
$\mathcal{P}_{0.809}$ - $\mathcal{P}_{0.827}$	(0;1,0,1) (0;1,1,1) (0;2,1,1) (0;3,1,2) (1;3,2,2) (2;4,2,2) (4;4,2,2)	0 0 0 1 1 2
$\mathcal{P}_{0.827}$ - $\mathcal{P}_{0.84}$	(0;1,0,1) (0;1,1,1) (0;2,1,1) (0;3,1,2) (0;4,2,3) (2;4,3,2) (4;5,2,2) (7;4,2,2) (9;4,2,2)	0 0 0 0 2 2 3 2
$\mathcal{P}_{0.84}$ - $\mathcal{P}_{0.856}$	(0;1,0,1) (0;1,1,1) (0;2,1,1) (0;3,1,2) (0;4,2,3) (2;4,3,2) (4;5,2,2) (6;5,2,3) (9;4,3,2)	0 0 0 0 2 2 2 3
$\mathcal{P}_{0.856}$ - $\mathcal{P}_{0.8612}$	(0;1,0,1) (0;1,1,1) (0;2,1,1) (0;3,1,2) (0;4,2,3) (1;5,3,3) (4;5,3,2) (7;5,2,2) (9;5,2,3) (11;5,3,3)	0 0 0 0 1 3 3 2 2
$\mathcal{P}_{0.8612}$ - $\mathcal{P}_{0.8725}$	(0;1,0,1) (0;1,1,1) (0;2,1,1) (0;3,1,2) (0;4,2,3) (1;5,3,3) (4;5,3,2) (6;6,2,3) (9;5,3,3)	0 0 0 0 1 3 2 3
$\mathcal{P}_{0.8725}$ - $\mathcal{P}_{0.8804}$	(0;1,0,1) (0;1,1,1) (0;2,1,1) (0;3,1,2) (0;4,2,3) (1;5,3,3) (3;6,3,3) (6;6,3,3)	0 0 0 0 1 2 3
$\mathcal{P}_{0.8804}$ - $\mathcal{P}_{?}$	(0;1,0,1) (0;1,1,1) (0;2,1,1) (0;3,1,2) (0;4,2,3) (0;6,3,4) (3;6,4,3) (6;7,3,3) (10;6,3,3) (13;6,3,3)	0 0 0 0 0 3 3 4 3

Table 2: (1,3)-lopsided trees: for each interval shown in the first column, the second column shows the optimal path for the p's sampled in that interval. Highlighted nodes are where level repetitions occur. The third column shows the number of leaves per level; highlighted numbers compose the cycle.

Figure 20: Sampled p's in the same interval share the same optimal tree.

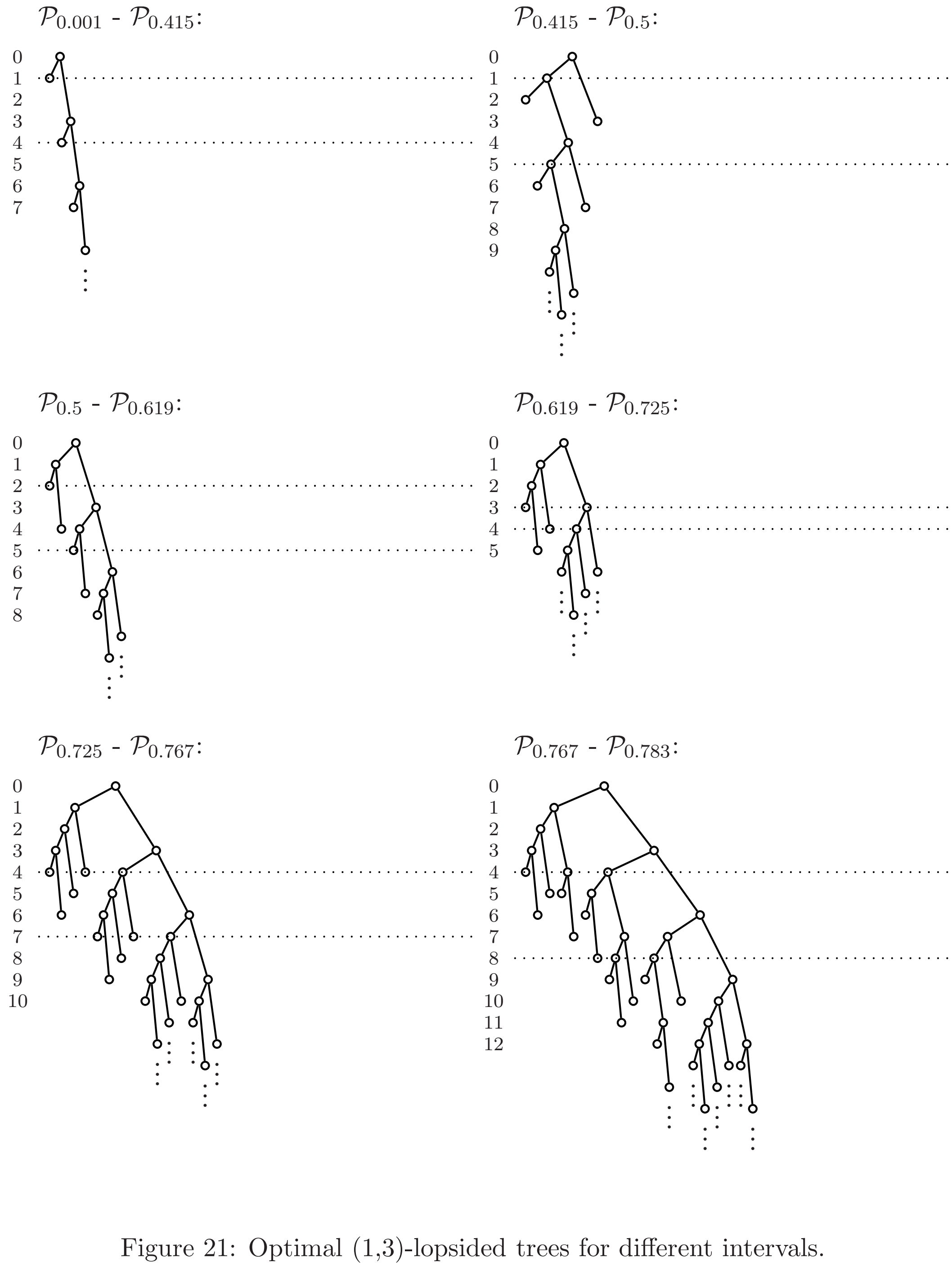


Figure 21: Optimal (1,3)-lopsided trees for different intervals.

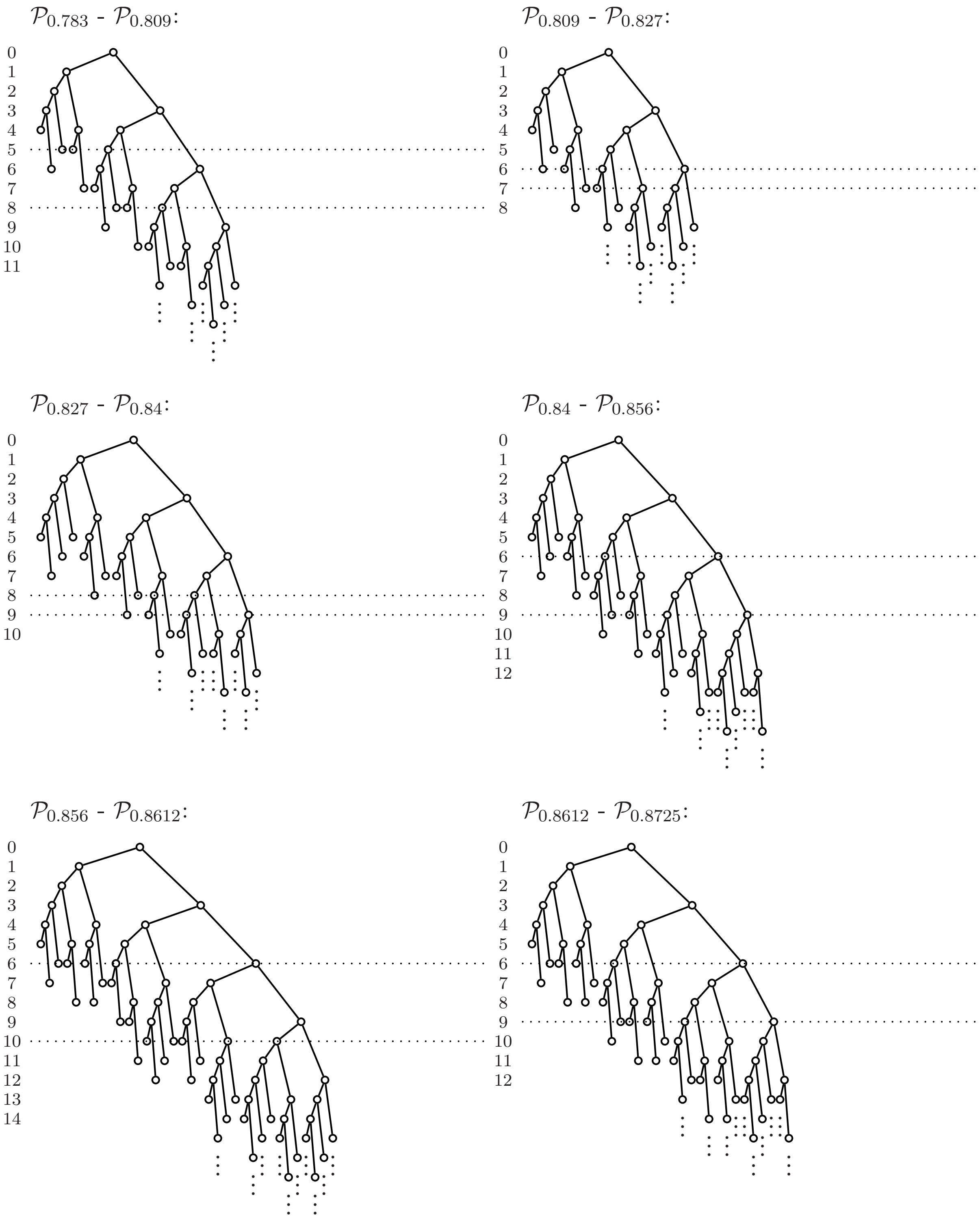


Figure 22: Optimal (1,3)-lopsided trees for different intervals (con't).

D Optimal (2,3)-Lopsided Trees

Interval	Expansion	Leaves per level
$\mathcal{P}_{0.001}$ - $\mathcal{P}_{0.5}$	(0;0,1,1) (0;1,1,0) (1;1,0,0) (1;0,1,1)	0 1 0
$\mathcal{P}_{0.5}$ - $\mathcal{P}_{0.619}$	(0;0,1,1) (0;1,1,0) (0;1,1,1) (1;1,1,0)	0 0 1
$\mathcal{P}_{0.619}$ - $\mathcal{P}_{0.683}$	(0;0,1,1) (0;1,1,0) (0;1,1,1) (0;1,2,1) (1;2,1,0) (2;1,1,1)	0 0 0 1 1
$\mathcal{P}_{0.683}$ - $\mathcal{P}_{0.725}$	(0;0,1,1) (0;1,1,0) (0;1,1,1) (0;1,2,1) (1;2,1,0) (1;1,2,2) (2;2,2,0) (3;2,1,1) (4;1,2,1)	0 0 0 1 0 1 1 1
$\mathcal{P}_{0.725}$ - $\mathcal{P}_{0.779}$	(0;0,1,1) (0;1,1,0) (0;1,1,1) (0;1,2,1) (0;2,2,1) (1;2,2,1)	0 0 0 0 1
$\mathcal{P}_{0.779}$ - $\mathcal{P}_{0.801}$	(0;0,1,1) (0;1,1,0) (0;1,1,1) (0;1,2,1) (0;2,2,1) (0;2,3,2) (2;3,2,0) (3;2,2,2) (4;2,3,1) (5;3,2,1) (6;2,3,2)	0 0 0 0 0 2 1 1 1 1
$\mathcal{P}_{0.801}$ - $\mathcal{P}_{0.82}$	(0;0,1,1) (0;1,1,0) (0;1,1,1) (0;1,2,1) (0;2,2,1) (0;2,3,2) (1;3,3,1) (3;3,2,1) (4;2,3,2)	0 0 0 0 0 1 2 1
$\mathcal{P}_{0.82}$ - $\mathcal{P}_{0.838}$	(0;0,1,1) (0;1,1,0) (0;1,1,1) (0;1,2,1) (0;2,2,1) (0;2,3,2) (1;3,3,1) (2;3,3,2) (4;3,3,1)	0 0 0 0 0 1 1 2
$\mathcal{P}_{0.838}$ - $\mathcal{P}_{0.851}$	(0;0,1,1) (0;1,1,0) (0;1,1,1) (0;1,2,1) (0;2,2,1) (0;2,3,2) (0;3,4,2) (2;4,3,1) (4;3,3,2) (5;3,4,2)	0 0 0 0 0 0 2 2 1
$\mathcal{P}_{0.851}$ - $\mathcal{P}_{0.861}$	(0;0,1,1) (0;1,1,0) (0;1,1,1) (0;1,2,1) (0;2,2,1) (0;2,3,2) (0;3,4,2) (2;4,3,1) (3;3,4,3) (5;4,4,1) (7;4,3,2) (9;3,4,2)	0 0 0 0 0 0 2 1 2 2 2
$\mathcal{P}_{0.861}$ - $\mathcal{P}_{0.8759}$	(0;0,1,1) (0;1,1,0) (0;1,1,1) (0;1,2,1) (0;2,2,1) (0;2,3,2) (0;3,4,2) (1;4,4,2) (3;4,4,2)	0 0 0 0 0 0 1 2
$\mathcal{P}_{0.8759}$ - $\mathcal{P}_{0.8831}$	(0;0,1,1) (0;1,1,0) (0;1,1,1) (0;1,2,1) (0;2,2,1) (0;2,3,2) (0;3,4,2) (0;4,5,3) (3;5,4,1) (5;4,4,3) (7;4,5,2) (9;5,4,2) (11;4,5,3)	0 0 0 0 0 0 0 3 2 2 2 2
$\mathcal{P}_{0.8831}$ - $\mathcal{P}_{0.8899}$	(0;0,1,1) (0;1,1,0) (0;1,1,1) (0;1,2,1) (0;2,2,1) (0;2,3,2) (0;3,4,2) (0;4,5,3) (2;5,5,2) (5;5,4,2) (7;4,5,3)	0 0 0 0 0 0 0 2 3 2
$\mathcal{P}_{0.8899}$ - $\mathcal{P}_{0.8971}$	(0;0,1,1) (0;1,1,0) (0;1,1,1) (0;1,2,1) (0;2,2,1) (0;2,3,2) (0;3,4,2) (0;4,5,3) (2;5,5,2) (4;5,5,3) (7;5,5,2)	0 0 0 0 0 0 0 2 2 3
$\mathcal{P}_{0.8971}$ - $\mathcal{P}_{?}$	(0;0,1,1) (0;1,1,0) (0;1,1,1) (0;1,2,1) (0;2,2,1) (0;2,3,2) (0;3,4,2) (0;4,5,3) (1;5,6,3) (4;6,5,2) (7;5,5,3) (9;5,6,3)	0 0 0 0 0 0 0 1 3 3 2

Table 3: (2,3)-lopsided trees: for each interval shown in the first column, the second column shows the optimal path for the p's sampled in that interval. Highlighted nodes are where level repetitions occur. The third column shows the number of leaves per level; highlighted numbers compose the cycle.

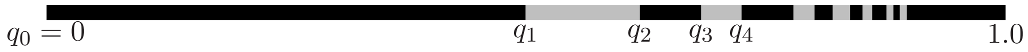


Figure 23: Sampled p's in the same interval share the same optimal tree.

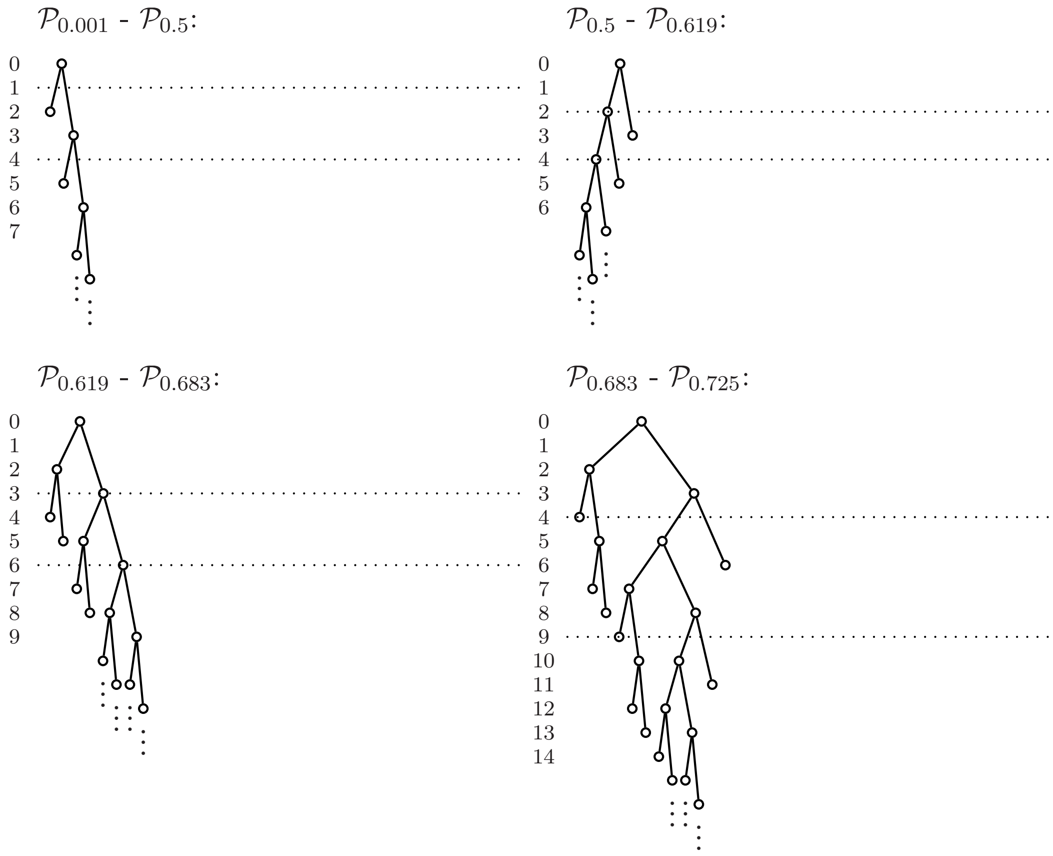


Figure 24: Optimal (2,3)-lopsided trees for different intervals.

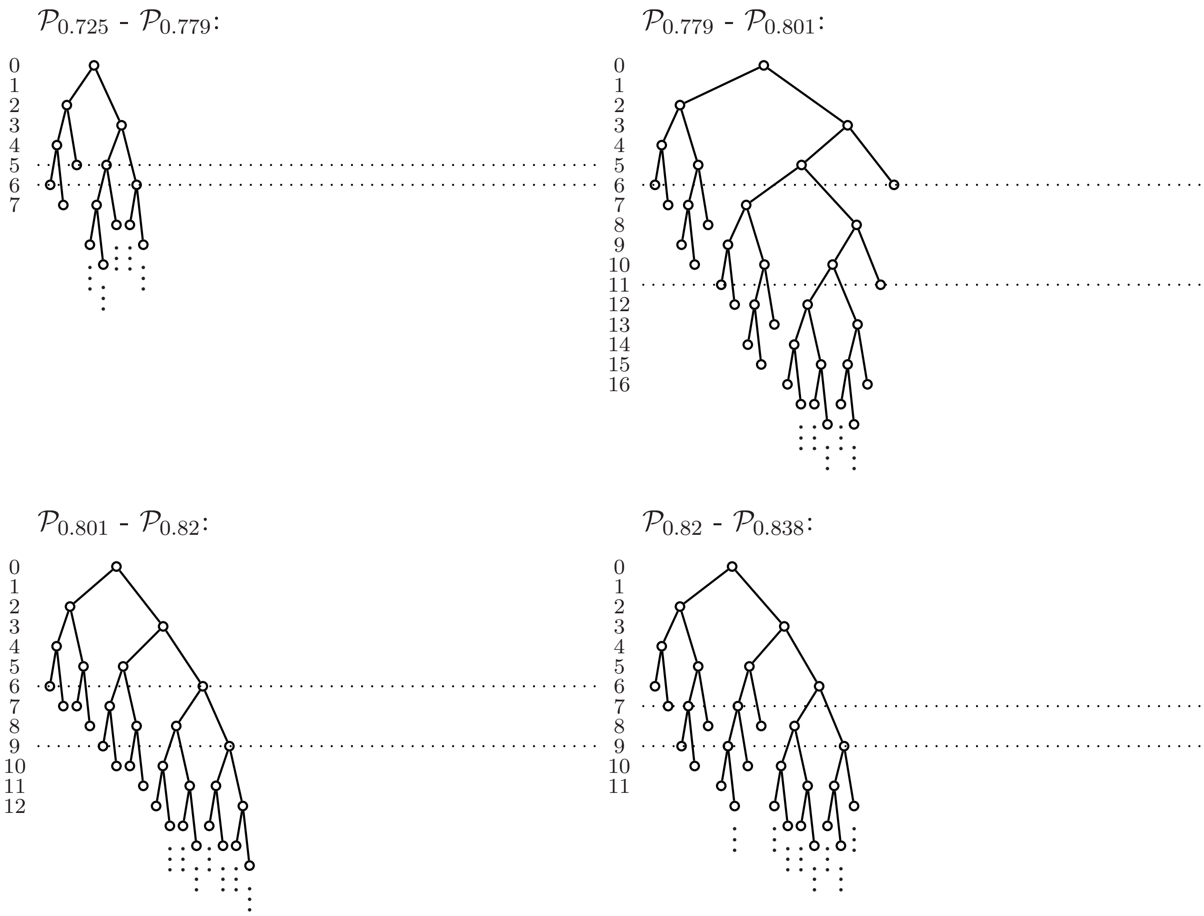


Figure 25: Optimal (2,3)-lopsided trees for different intervals (con't).

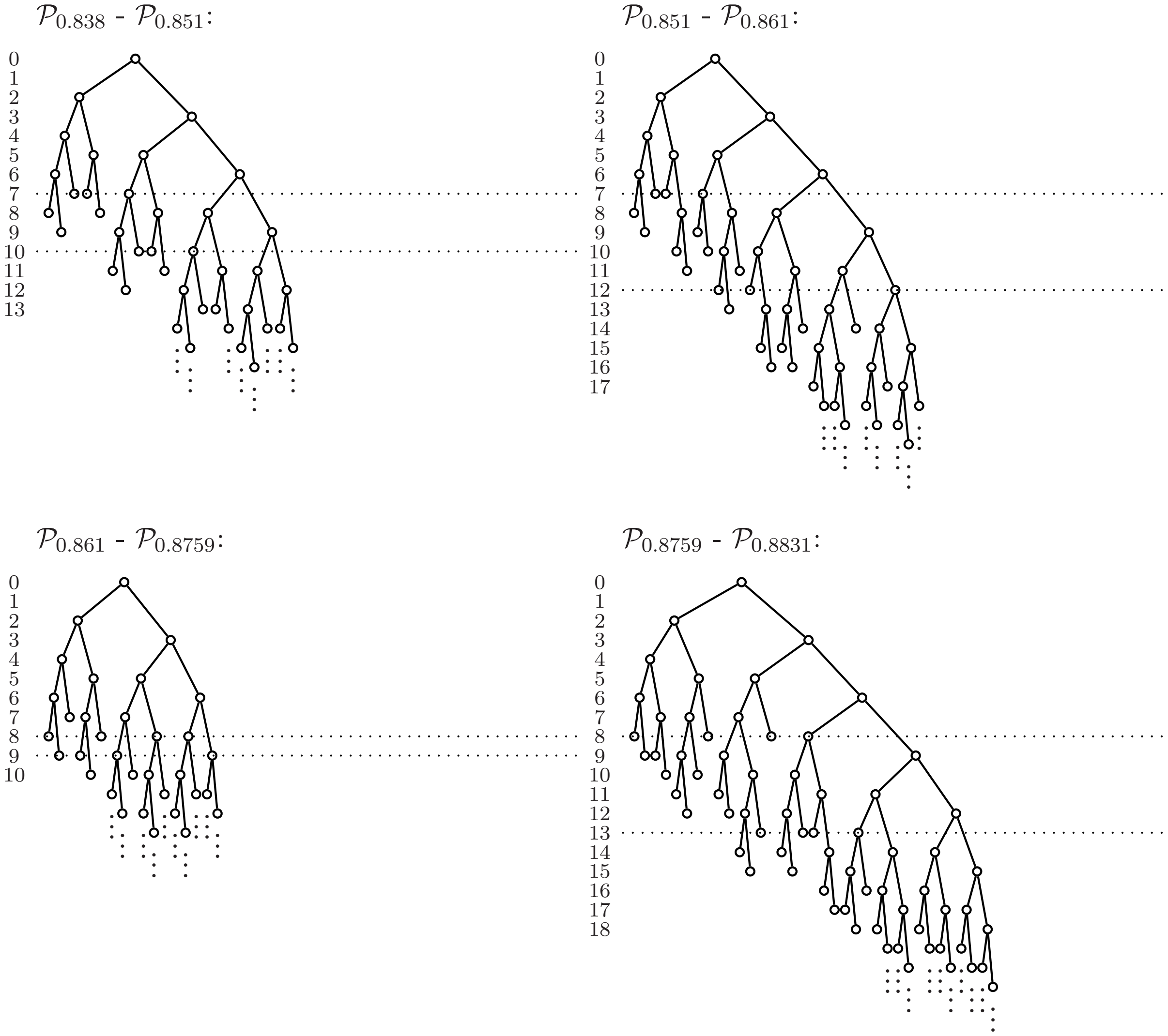


Figure 26: Optimal (2,3)-lopsided trees for different intervals (con't).

E Optimal (3,5)-Lopsided Trees

Interval	Expansion	Leaves per level
$\mathcal{P}_{0.001}$ - $\mathcal{P}_{0.619}$	(0;0,0,1,0,1) (0;0,1,0,1,0) (0;1,0,1,0,0) (0;0,1,1,0,1) (0;1,1,0,1,0) (1;1,0,1,0,0)	0 0 0 0 1
$\mathcal{P}_{0.619}$ - $\mathcal{P}_{0.683$	(0;0,0,1,0,1) (0;0,1,0,1,0) (0;1,0,1,0,0) (0;0,1,1,0,1) (0;1,1,0,1,0) (0;1,0,2,0,1) (1;0,2,0,1,0) (1;2,0,1,0,0) (2;0,1,1,0,1)	0 0 0 0 0 1 0 1
$\mathcal{P}_{0.683}$ - $\mathcal{P}_{0.704}$	(0;0,0,1,0,1) (0;0,1,0,1,0) (0;1,0,1,0,0) (0;0,1,1,0,1) (0;1,1,0,1,0) (0;1,0,2,0,1) (1;0,2,0,1,0) (1;2,0,1,0,0) (1;0,1,2,0,2) (1;1,2,0,2,0) (2;2,0,2,0,0) (3;0,2,1,0,1) (3;2,1,0,1,0) (4;1,0,2,0,1)	0 0 0 0 0 1 0 0 0 1 1 0 1
$\mathcal{P}_{0.704}$ - $\mathcal{P}_{0.71}$	(0;0,0,1,0,1) (0;0,1,0,1,0) (0;1,0,1,0,0) (0;0,1,1,0,1) (0;1,1,0,1,0) (0;1,0,2,0,1) (0;0,2,1,1,1) (0;2,1,1,1,0) (2;1,1,1,0,0) (2;1,1,1,0,1) (2;1,1,1,1,1) (3;1,1,1,1,0) (4;1,1,1,0,0)	0 0 0 0 0 0 0 2 0 0 1 1
$\mathcal{P}_{0.71}$ - $\mathcal{P}_{0.725}$	(0;0,0,1,0,1) (0;0,1,0,1,0) (0;1,0,1,0,0) (0;0,1,1,0,1) (0;1,1,0,1,0) (0;1,0,2,0,1) (0;0,2,1,1,1) (0;2,1,1,1,0) (1;1,1,2,0,1) (2;1,2,0,1,0) (3;2,0,1,0,0) (3;0,1,2,0,2) (3;1,2,0,2,0) (4;2,0,2,0,0) (5;0,2,1,0,1) (5;2,1,0,1,0) (6;1,0,2,0,1)	0 0 0 0 0 0 0 1 1 1 0 0 1 1 0 1
$\mathcal{P}_{0.725}$ - $\mathcal{P}_{0.755}$	(0;0,0,1,0,1) (0;0,1,0,1,0) (0;1,0,1,0,0) (0;0,1,1,0,1) (0;1,1,0,1,0) (0;1,0,2,0,1) (0;0,2,1,1,1) (0;2,1,1,1,0) (1;1,1,2,0,1) (2;1,2,0,1,0) (2;2,0,2,0,1) (3;0,2,1,1,1)	0 0 0 0 0 0 0 1 1 0 1
$\mathcal{P}_{0.755}$ - $\mathcal{P}_{0.779}$	(0;0,0,1,0,1) (0;0,1,0,1,0) (0;1,0,1,0,0) (0;0,1,1,0,1) (0;1,1,0,1,0) (0;1,0,2,0,1) (0;0,2,1,1,1) (0;2,1,1,1,0) (1;1,1,2,0,1) (1;1,2,1,1,1) (2;2,1,1,1,0)	0 0 0 0 0 0 0 1 0 1
$\mathcal{P}_{0.779}$ - $\mathcal{P}_{0.8001}$	(0;0,0,1,0,1) (0;0,1,0,1,0) (0;1,0,1,0,0) (0;0,1,1,0,1) (0;1,1,0,1,0) (0;1,0,2,0,1) (0;0,2,1,1,1) (0;2,1,1,1,0) (0;1,1,3,0,2) (1;1,3,0,2,0) (2;3,0,2,0,0) (3;0,2,2,0,2) (3;2,2,0,2,0) (4;2,0,3,0,1) (5;0,3,1,1,1) (5;3,1,1,1,0) (6;1,1,3,0,2)	0 0 0 0 0 0 0 0 1 1 1 0 1 1 0 1
$\mathcal{P}_{0.8001}$ - $\mathcal{P}_{0.8088}$	(0;0,0,1,0,1) (0;0,1,0,1,0) (0;1,0,1,0,0) (0;0,1,1,0,1) (0;1,1,0,1,0) (0;1,0,2,0,1) (0;0,2,1,1,1) (0;2,1,1,1,0) (0;1,1,3,0,2) (1;1,3,0,2,0) (1;3,0,3,0,1) (3;0,3,1,1,1) (3;3,1,1,1,0) (4;1,1,3,0,2)	0 0 0 0 0 0 0 0 1 0 2 0 1
$\mathcal{P}_{0.8088}$ - $\mathcal{P}_{0.817}$	(0;0,0,1,0,1) (0;0,1,0,1,0) (0;1,0,1,0,0) (0;0,1,1,0,1) (0;1,1,0,1,0) (0;1,0,2,0,1) (0;0,2,1,1,1) (0;2,1,1,1,0) (0;1,1,3,0,2) (0;1,3,1,2,1) (1;3,1,2,1,0) (3;1,2,2,0,1) (3;2,2,1,1,1) (4;2,1,2,1,1) (5;1,2,2,1,1) (6;2,2,1,1,0) (7;2,1,2,0,1) (7;1,2,2,1,2) (8;2,2,1,2,0) (9;2,1,3,0,1) (10;1,3,1,1,1) (11;3,1,1,1,0) (12;1,1,3,0,2)	0 0 0 0 0 0 0 0 0 1 2 0 1 1 1 1 0 1 1 1 1 1
$\mathcal{P}_{0.817}$ - $\mathcal{P}_{?}$	(0;0,0,1,0,1) (0;0,1,0,1,0) (0;1,0,1,0,0) (0;0,1,1,0,1) (0;1,1,0,1,0) (0;1,0,2,0,1) (0;0,2,1,1,1) (0;2,1,1,1,0) (0;1,1,3,0,2) (0;1,3,1,2,1) (1;3,1,2,1,0) (2;1,2,3,0,2) (3;2,3,0,2,0) (4;3,0,3,0,1) (5;0,3,2,1,2) (5;3,2,1,2,0) (7;2,1,3,0,1) (8;1,3,1,1,1) (9;3,1,1,1,0) (10;1,1,3,0,2)	0 0 0 0 0 0 0 0 0 1 1 1 1 1 0 2 1 1 1

Table 4: (3,5)-lopsided trees: for each interval shown in the first column, the second column shows the optimal path for the p's sampled in that interval. Highlighted nodes are where level repetitions occur. The third column shows the number of leaves per level; highlighted numbers compose the cycle.

Figure 27: Sampled p's in the same interval share the same optimal tree.

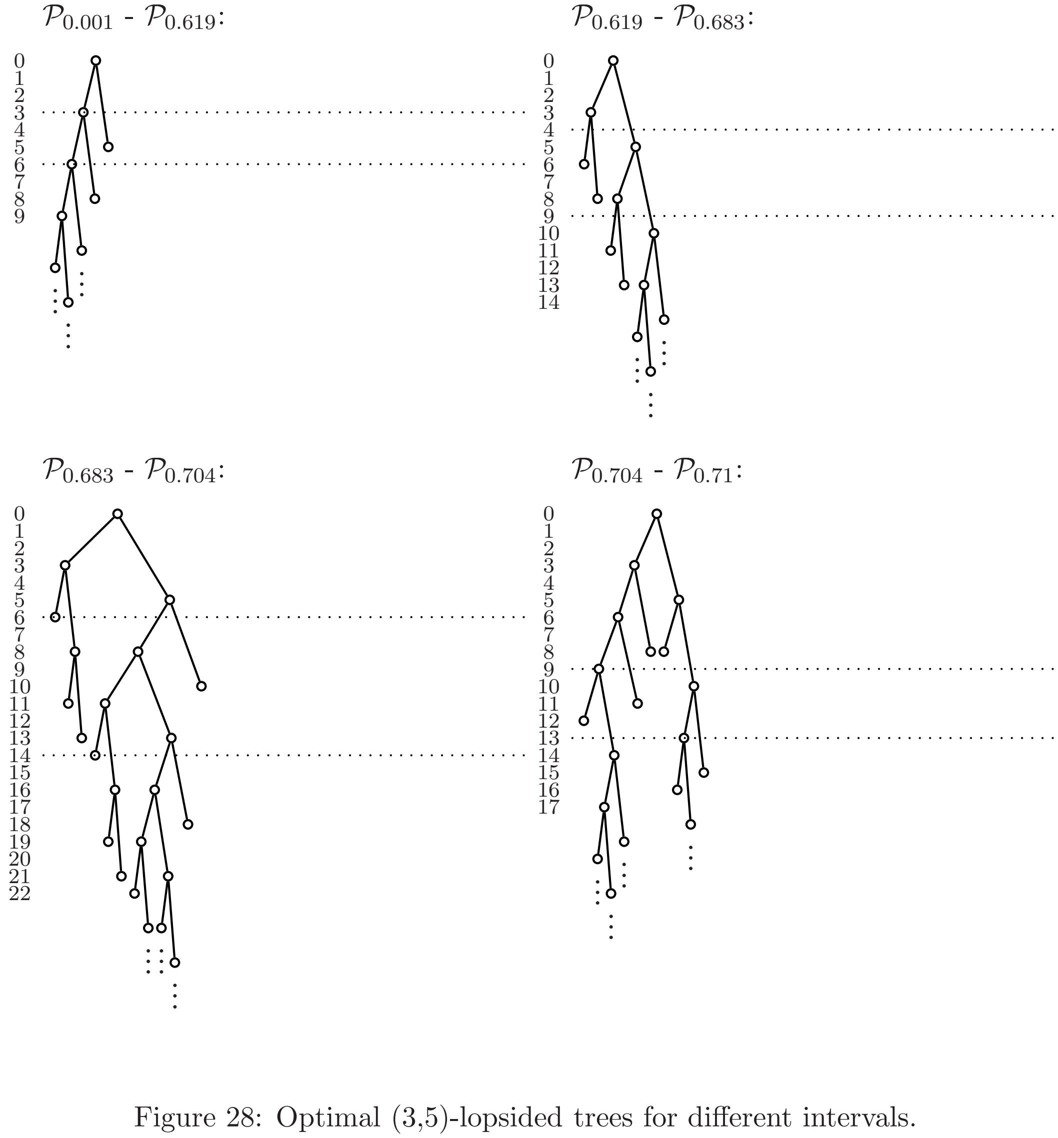


Figure 28: Optimal (3,5)-lopsided trees for different intervals.

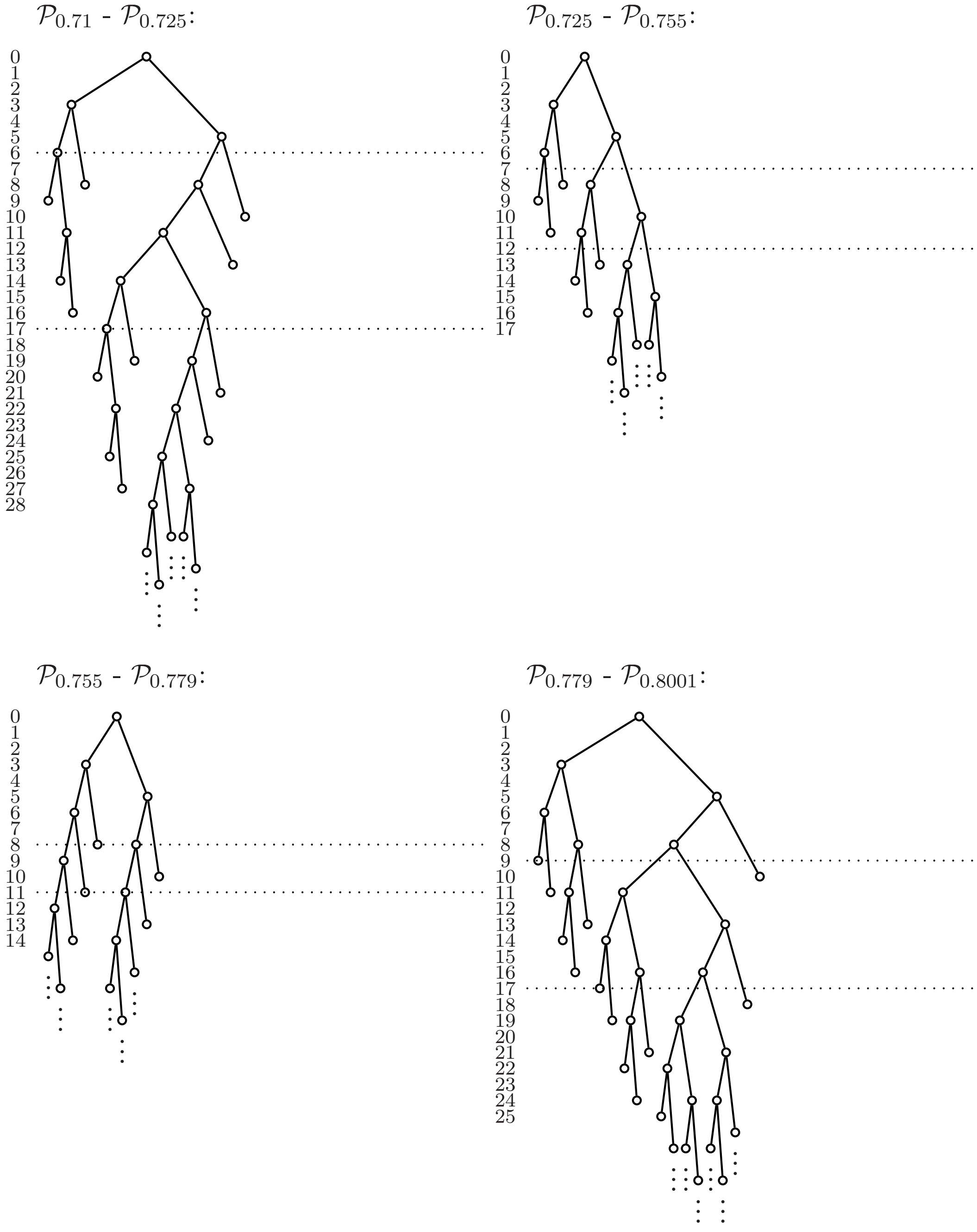


Figure 29: Optimal (3,5)-lopsided trees for different intervals (con't).

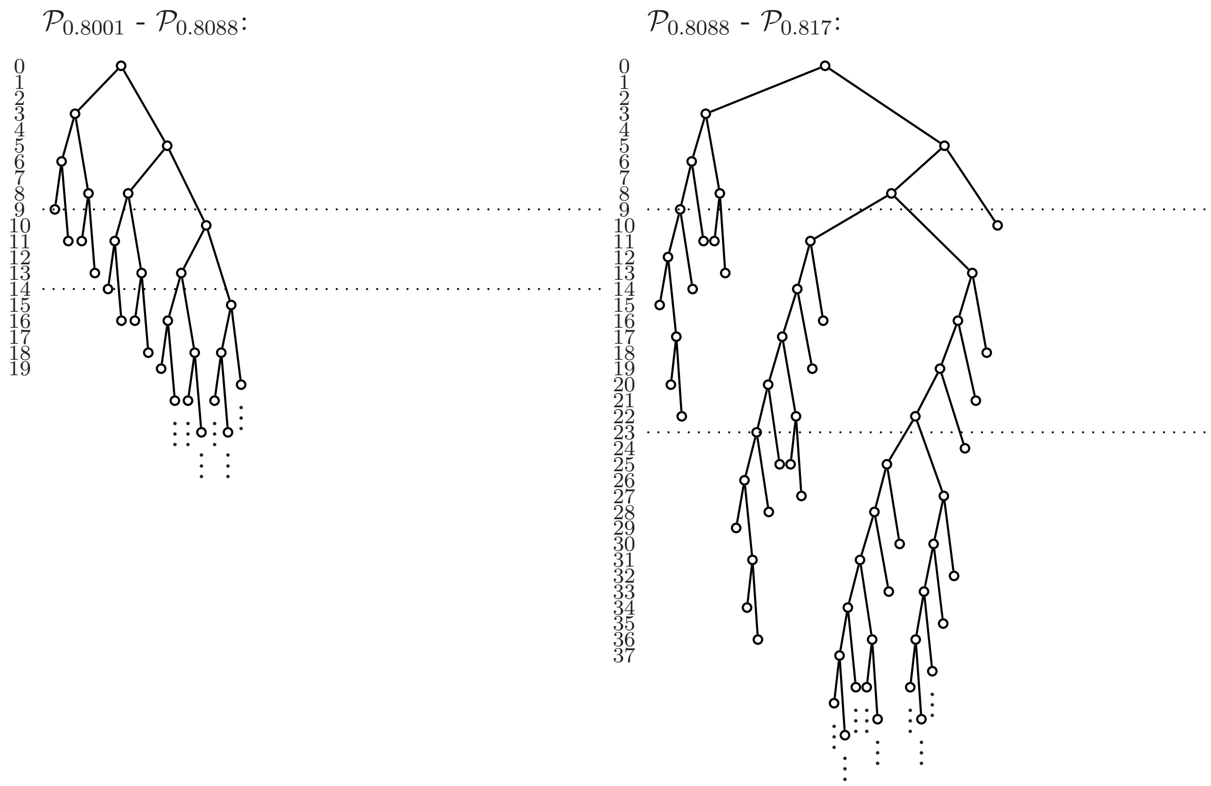


Figure 30: Optimal (3,5)-lopsided trees for different intervals (con't).

F Optimal (2,3,5)-Lopsided Trees

Interval	Expansion	Leaves per level
$\mathcal{P}_{0.001}$ - $\mathcal{P}_{0.501}$	(0;0,1,1,0,1) (0;1,1,0,1,0) (1;1,0,1,0,0) (2;0,1,0,0,0) (2;1,0,0,0,0) (2;0,1,1,0,1)	0 1 1 0 0
$\mathcal{P}_{0.501}$ - $\mathcal{P}_{0.57}$	(0;0,1,1,0,1) (0;1,1,0,1,0) (1;1,0,1,0,0) (1;0,2,1,0,1) (1;2,1,0,1,0) (3;1,0,1,0,0)	0 1 0 0 2
$\mathcal{P}_{0.57}$ - $\mathcal{P}_{0.683}$	(0;0,1,1,0,1) (0;1,1,0,1,0) (0;1,1,2,0,1) (1;1,2,0,1,0) (2;2,0,1,0,0) (3;0,2,1,0,1) (3;2,1,0,1,0) (4;1,1,2,0,1)	0 0 1 1 1 0 1
$\mathcal{P}_{0.683}$ - $\mathcal{P}_{0.713}$	(0;0,1,1,0,1) (0;1,1,0,1,0) (0;1,1,2,0,1) (1;1,2,0,1,0) (1;2,1,2,0,1) (3;1,2,0,1,0)	0 0 1 0 2
$\mathcal{P}_{0.713}$ - $\mathcal{P}_{0.767}$	(0;0,1,1,0,1) (0;1,1,0,1,0) (0;1,1,2,0,1) (0;1,3,1,1,1) (1;3,1,1,1,0) (3;1,2,2,0,1) (4;2,2,0,1,0) (5;2,1,2,0,1) (6;1,3,1,1,1)	0 0 0 1 2 1 1 1
$\mathcal{P}_{0.767}$ - $\mathcal{P}_{0.773}$	(0;0,1,1,0,1) (0;1,1,0,1,0) (0;1,1,2,0,1) (0;1,3,1,1,1) (1;3,1,1,1,0) (3;1,2,2,0,1) (3;2,3,1,1,1) (5;3,1,1,1,0)	0 0 0 1 2 0 2
$\mathcal{P}_{0.773}$ - $\mathcal{P}_{0.799}$	(0;0,1,1,0,1) (0;1,1,0,1,0) (0;1,1,2,0,1) (0;1,3,1,1,1) (1;3,1,1,1,0) (2;1,3,3,0,2) (3;3,3,0,2,0) (5;3,1,3,0,1) (7;1,4,1,1,1) (8;4,1,1,1,0) (10;1,3,3,0,2)	0 0 0 1 1 1 2 2 1 2
$\mathcal{P}_{0.799}$ - $\mathcal{P}_{0.819}$	(0;0,1,1,0,1) (0;1,1,0,1,0) (0;1,1,2,0,1) (0;1,3,1,1,1) (0;3,2,2,1,1) (2;2,3,2,1,1) (4;3,2,1,1,0) (5;2,3,3,0,2) (7;3,3,0,2,0) (9;3,1,3,0,1) (11;1,4,1,1,1) (12;4,1,1,1,0) (14;1,3,3,0,2) (15;3,3,0,2,0)	0 0 0 0 2 2 1 2 2 2 1 2 1
$\mathcal{P}_{0.819}$ - $\mathcal{P}_{0.821}$	(0;0,1,1,0,1) (0;1,1,0,1,0) (0;1,1,2,0,1) (0;1,3,1,1,1) (0;3,2,2,1,1) (2;2,3,2,1,1) (4;3,2,1,1,0) (5;2,3,3,0,2) (7;3,3,0,2,0) (8;3,2,4,0,2) (11;2,4,0,2,0) (12;4,1,3,0,1) (15;1,4,1,1,1) (16;4,1,1,1,0) (18;1,3,3,0,2) (19;3,3,0,2,0)	0 0 0 0 2 2 1 2 1 3 1 3 1 2 1
$\mathcal{P}_{0.821}$ - $\mathcal{P}_{0.837}$	(0;0,1,1,0,1) (0;1,1,0,1,0) (0;1,1,2,0,1) (0;1,3,1,1,1) (0;3,2,2,1,1) (2;2,3,2,1,1) (3;3,3,2,1,1) (5;3,3,2,1,1)	0 0 0 0 2 1 2
$\mathcal{P}_{0.837}$ - $\mathcal{P}_{0.845}$	(0;0,1,1,0,1) (0;1,1,0,1,0) (0;1,1,2,0,1) (0;1,3,1,1,1) (0;3,2,2,1,1) (1;2,4,3,1,2) (3;4,3,1,2,0) (5;3,3,4,0,2) (8;3,4,0,2,0) (10;4,1,3,0,1) (12;1,5,2,1,2) (13;5,2,1,2,0) (16;2,3,4,0,2) (18;3,4,0,2,0)	0 0 0 0 1 2 2 3 2 2 1 3 2
$\mathcal{P}_{0.845}$ - $\mathcal{P}_{0.848}$	(0;0,1,1,0,1) (0;1,1,0,1,0) (0;1,1,2,0,1) (0;1,3,1,1,1) (0;3,2,2,1,1) (1;2,4,3,1,2) (3;4,3,1,2,0) (5;3,3,4,0,2) (8;3,4,0,2,0) (9;4,2,4,0,2) (12;2,5,1,2,1) (14;5,1,2,1,0) (17;1,4,3,0,2) (18;4,3,0,2,0) (20;3,2,4,0,2) (22;2,5,1,2,1)	0 0 0 0 1 2 2 3 1 3 2 3 1 2 2
$\mathcal{P}_{0.848}$ - $\mathcal{P}_{?}$	(0;0,1,1,0,1) (0;1,1,0,1,0) (0;1,1,2,0,1) (0;1,3,1,1,1) (0;3,2,2,1,1) (1;2,4,3,1,2) (3;4,3,1,2,0) (5;3,3,4,0,2) (8;3,4,0,2,0) (9;4,2,4,0,2) (12;2,5,1,2,1) (14;5,1,2,1,0) (16;1,5,4,0,3) (17;5,4,0,3,0) (21;4,1,4,0,1) (23;1,6,2,1,2) (24;6,2,1,2,0) (28;2,3,4,0,2) (30;3,4,0,2,0)	0 0 0 0 1 2 2 3 1 3 2 2 1 4 2 1 4 2

Table 5: (2,3,5)-lopsided trees: for each interval shown in the first column, the second column shows the optimal path for the p's sampled in that interval. Highlighted nodes are where level repetitions occur. The third column shows the number of leaves per level; highlighted numbers compose the cycle.

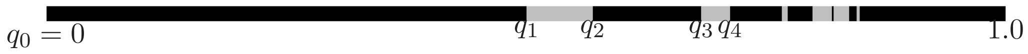


Figure 31: Sampled p's in the same interval share the same optimal tree.

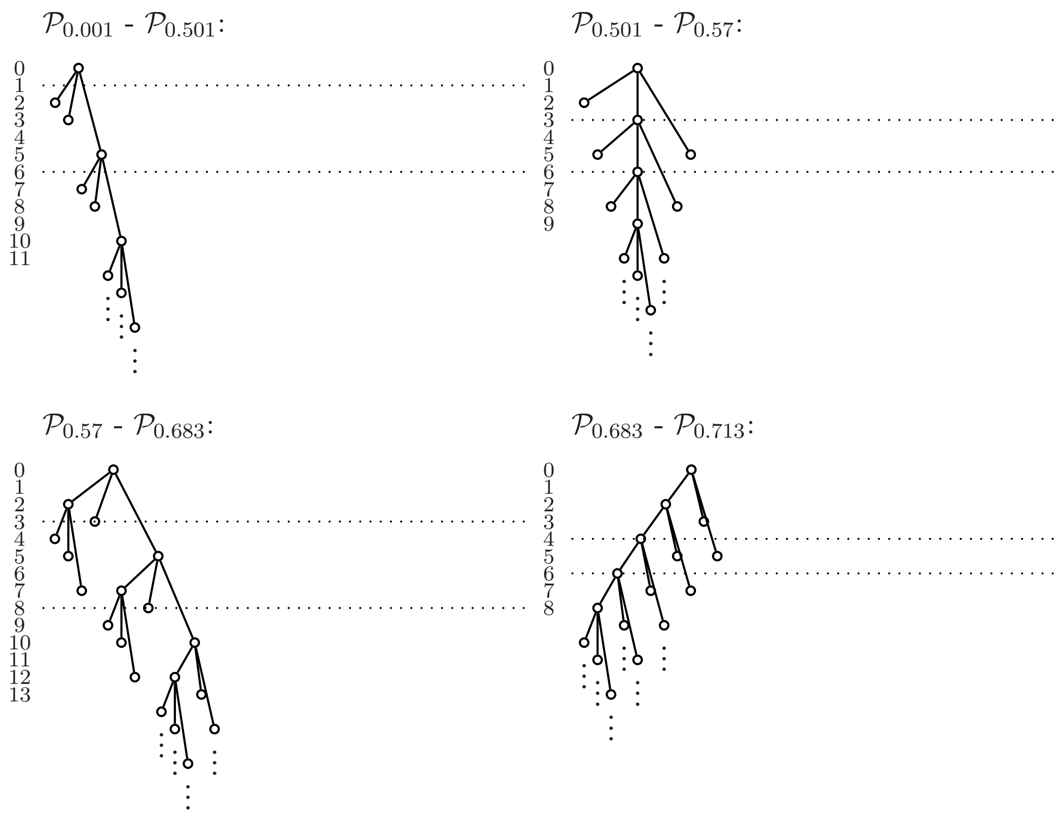


Figure 32: Optimal (2,3,5)-lopsided trees for different intervals.

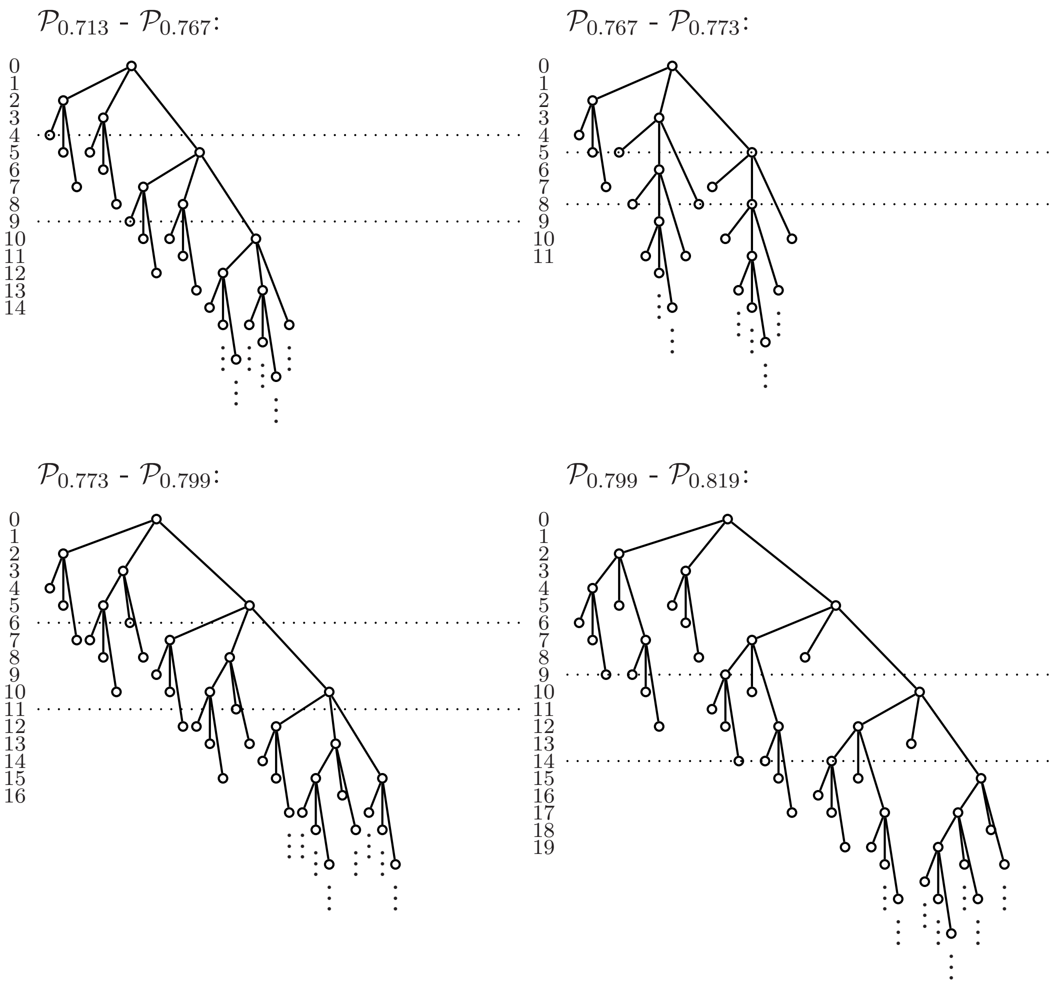


Figure 33: Optimal (2,3,5)-lopsided trees for different intervals (con't).

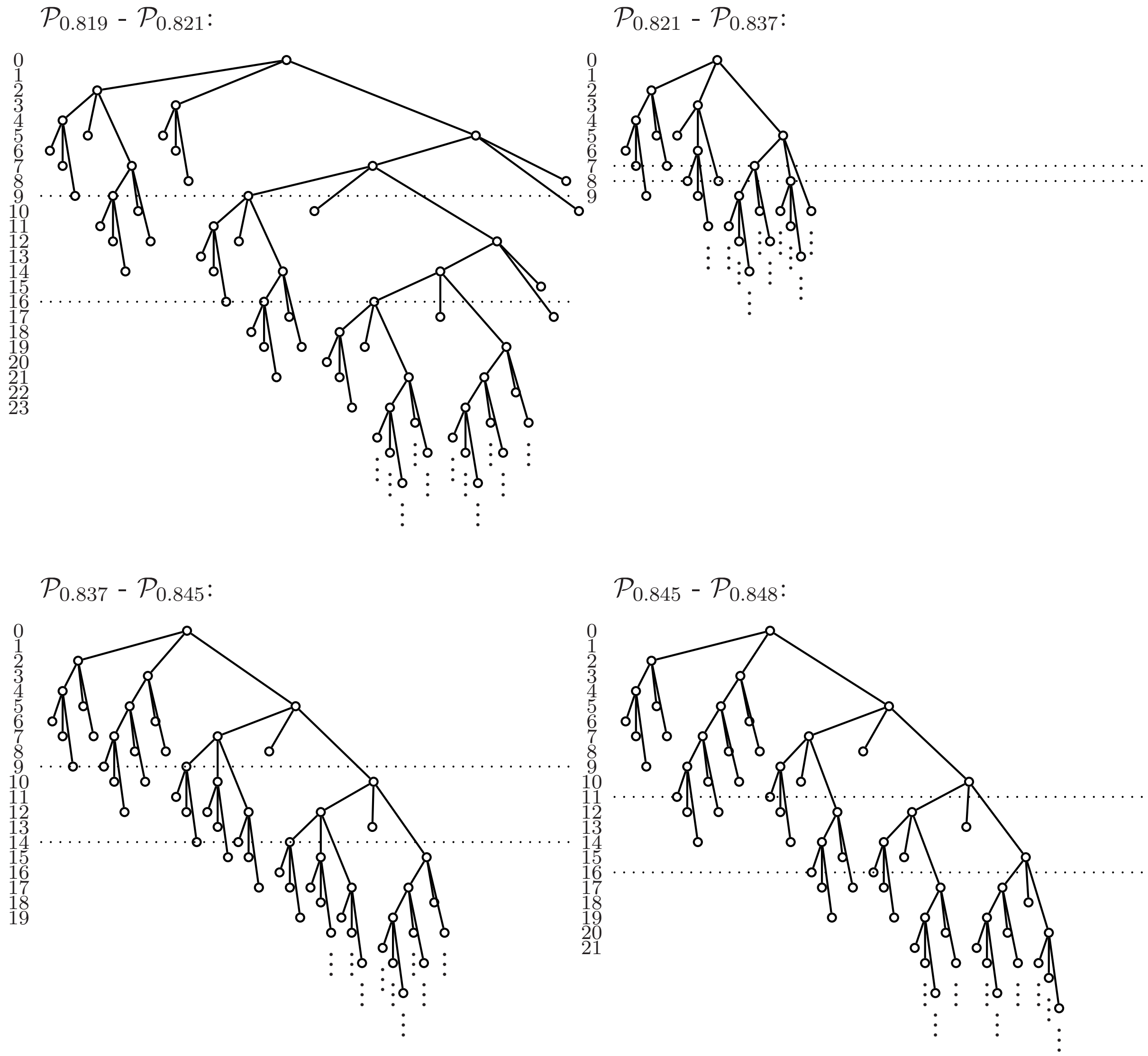


Figure 34: Optimal (2,3,5)-lopsided trees for different intervals (con't).

Appendix A.6 shows optimal 3-offset trees for $\alpha_0 = 0$ and α_1, α_2 ranging from 0 to 0.95 with 0.05 increments, and p ranging from 0.001 to a certain point close to 1 [7] with 0.001 increments. We first show, given values of α_1 and α_2, how (0,1) is partitioned such that probabilities in each partition share the same optimal tree. Labels of intervals are label of optimal tree for that interval. Finally we give realizations of these trees.

[7] In this implementation the algorithm stops when 8 intervals are found.

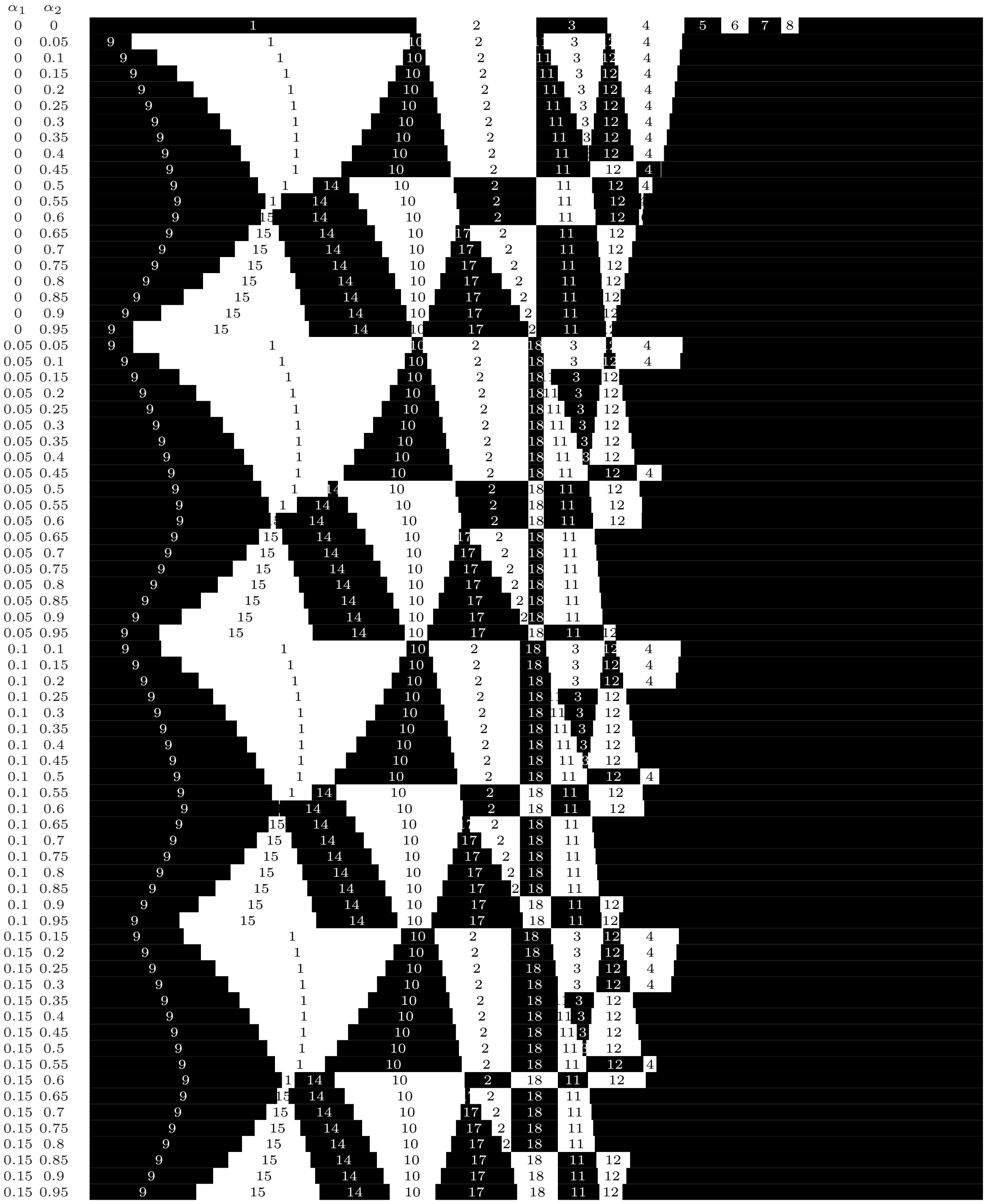


Figure 35: Partitions of (0,1) given α_1 and α_2.

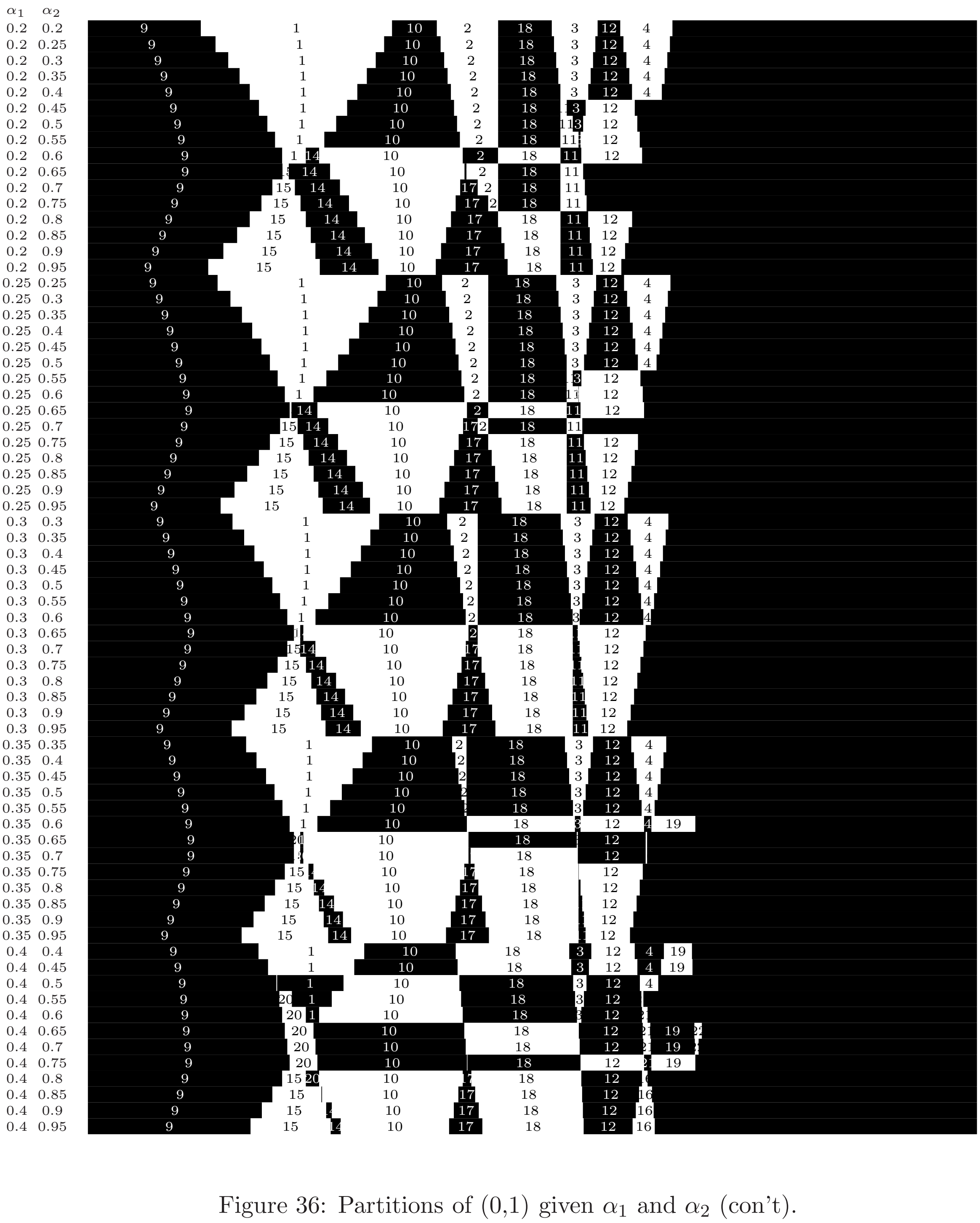


Figure 36: Partitions of (0,1) given α_1 and α_2 (con't).

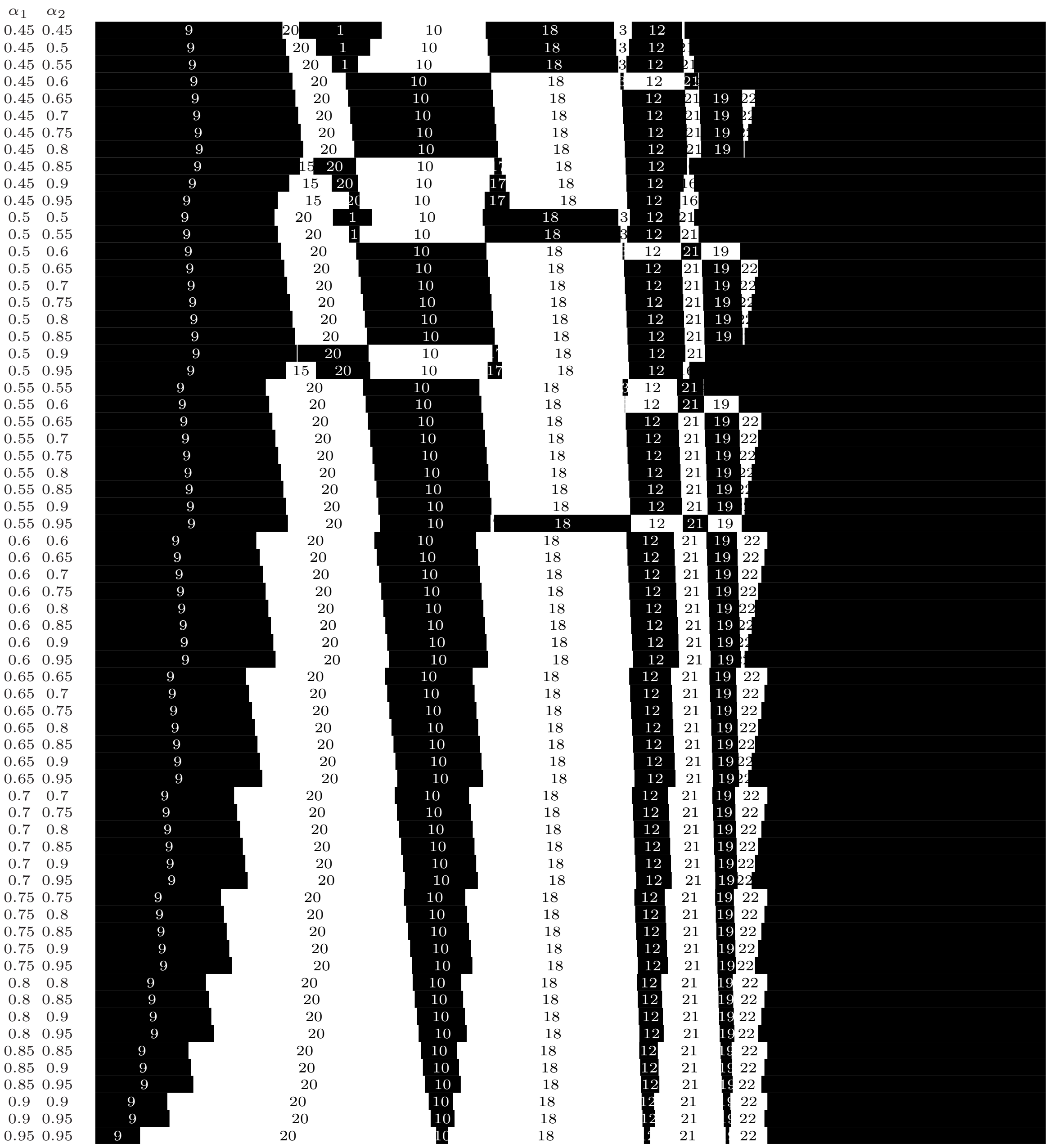


Figure 37: Partitions of (0,1) given α_1 and α_2 (con't).

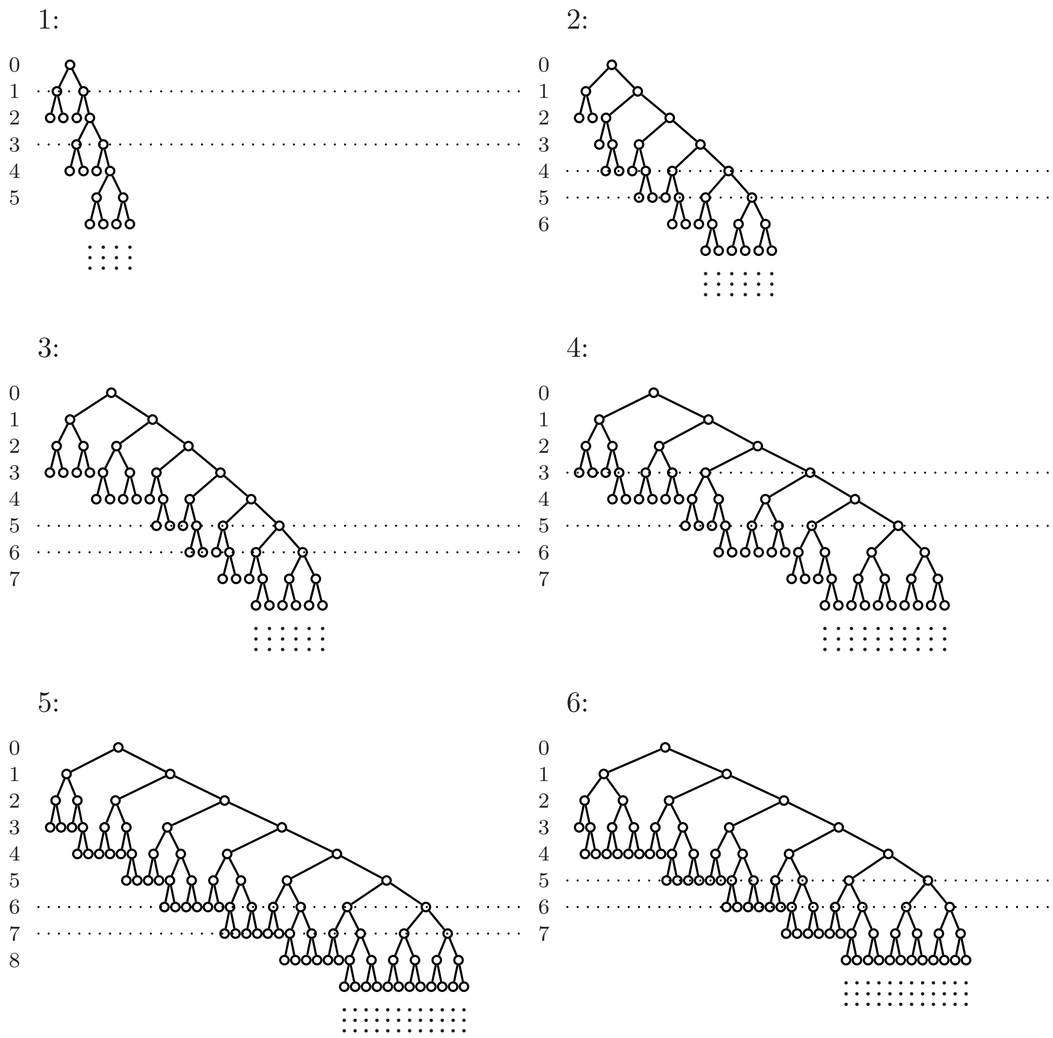


Figure 38: Optimal 3-offset trees for different intervals.

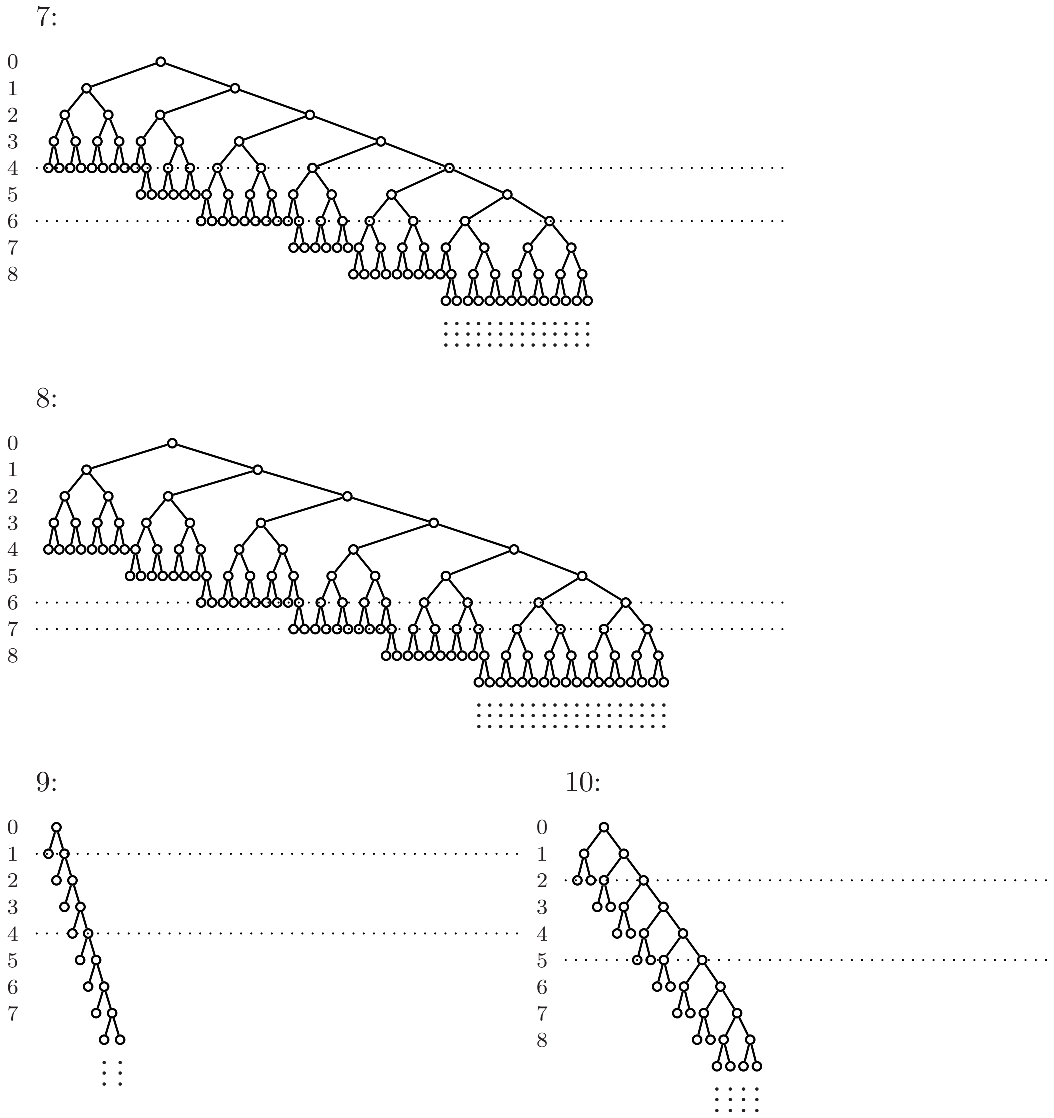


Figure 39: Optimal 3-offset trees for different intervals (con't).

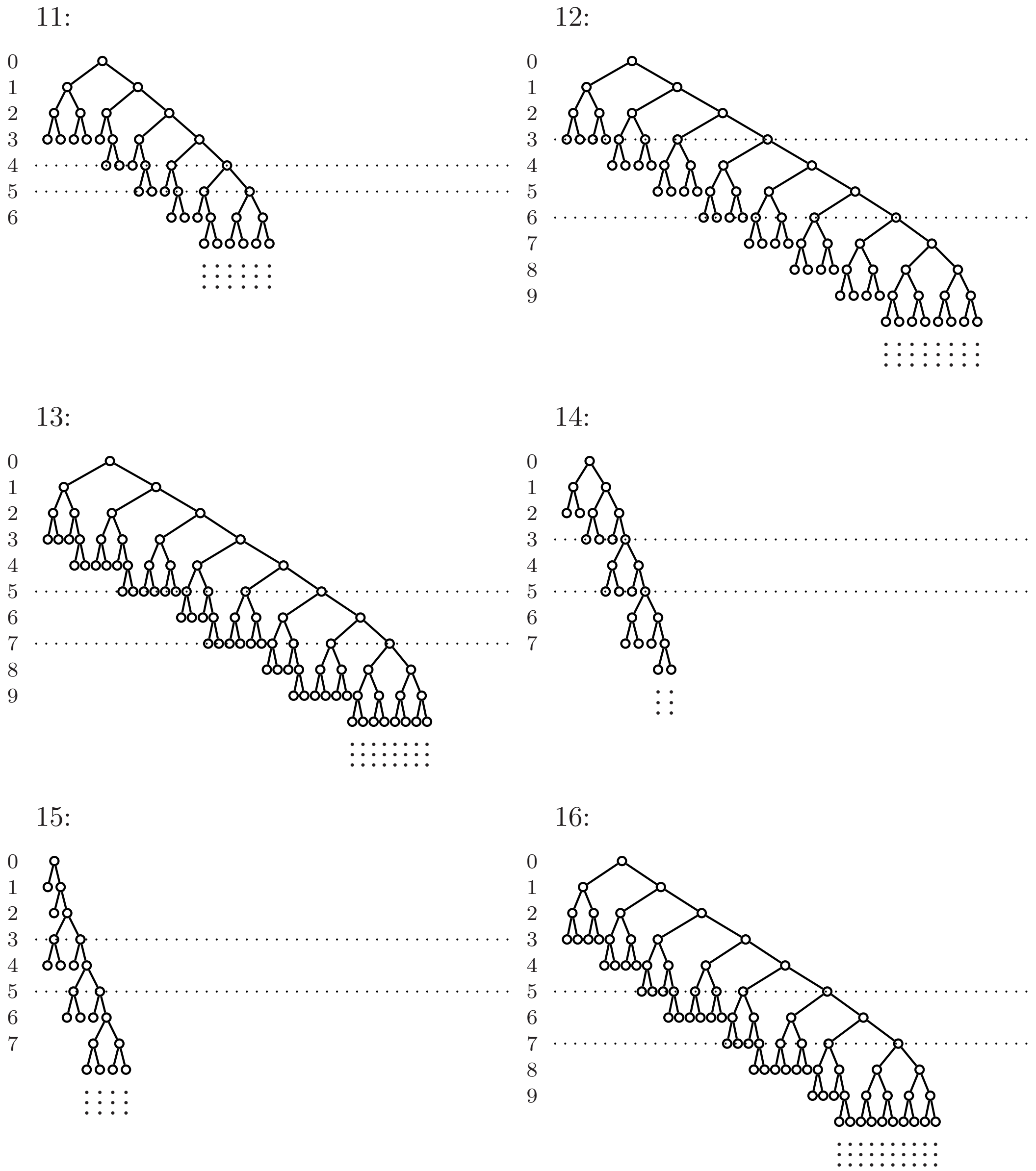


Figure 40: Optimal 3-offset trees for different intervals (con't).

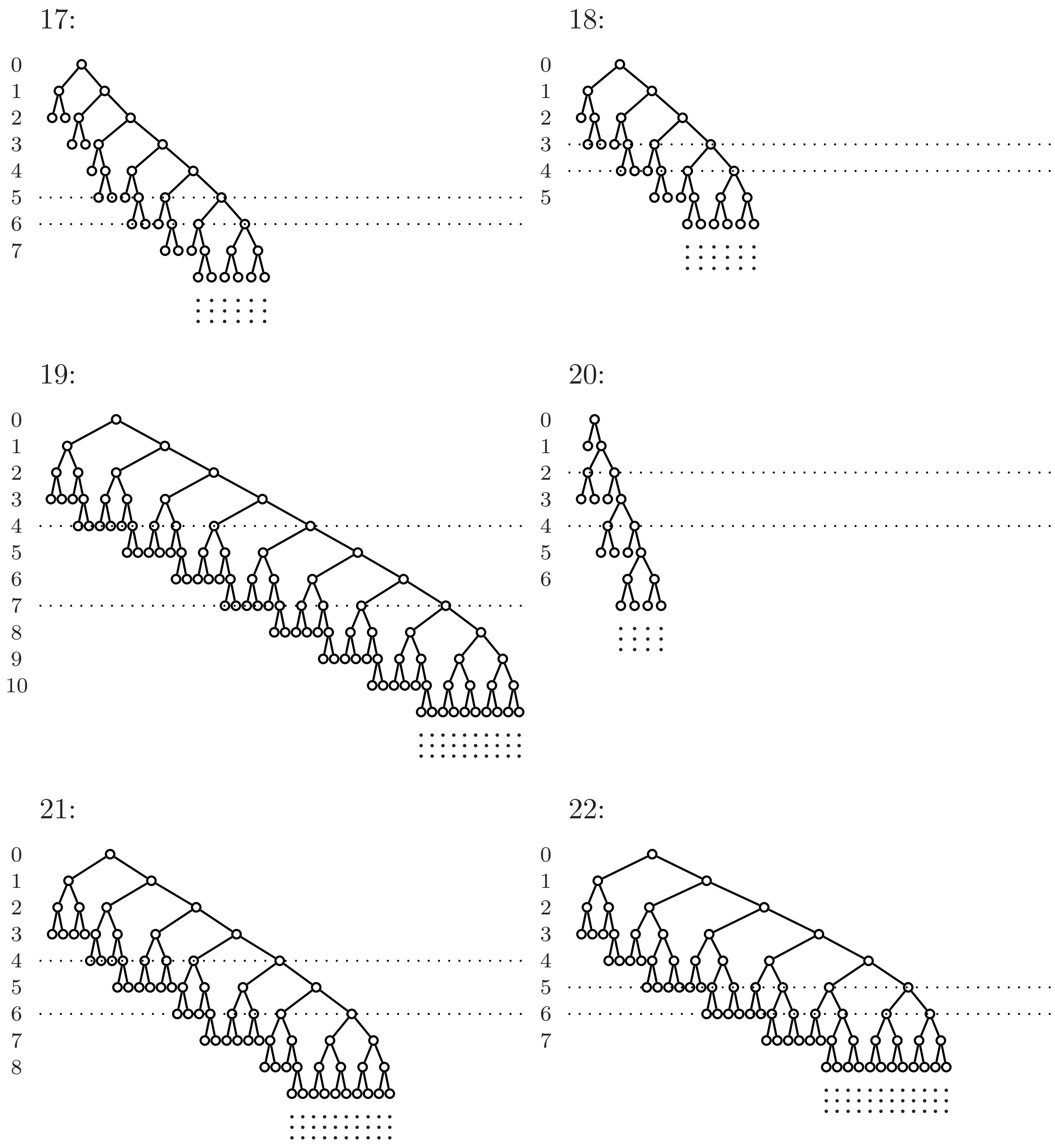


Figure 41: Optimal 3-offset trees for different intervals (con't).