Information Theory of Random Trees Induced by Stochastic Grammars

A DISSERTATION
SUBMITTED TO THE FACULTY OF THE GRADUATE SCHOOL
OF THE UNIVERSITY OF MINNESOTA
BY

Sang youn Oh

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
Doctor of Philosophy

John C. Kieffer, Advisor

May, 2012

© Sang youn Oh 2012
ALL RIGHTS RESERVED

Acknowledgements

I wish to express my deeply-felt thanks to my advisor, Professor John C. Kieffer for his warm encouragement, thoughtful guidance and patience. I appreciate all his contributions of time and ideas to make my Ph.D. experience productive. He guided me over a year of writing this thesis and gave me the moral support I need most during the hard time of writing. He took care of me until the very last moment before he retired. I hope he really enjoy having his full time as a productive researcher and a paper-writer after his retirement.

I also would like to thank Professor Tryphon Georgiou, Professor Jarvis Haupt, and Professor Greg Anderson for agreeing to serve as my committee members for final oral examination.

During my time as a Ph.D. student I met many nice and inspiring colleagues. Nohhyun encouraged and helped me a lot when I had hard time and we discussed valuable topics which gave me the chance to get to understand abstract knowledge better. Hyoungsuk is also a good friend who made my, sometimes a bit isolated, working life in Minnesota quite enjoyable.

Last, but most importantly, I thank my family and my husband, Seongwook. My family gave me continuing encouragement and love and I couldn't imagine my life in USA without them. Seongwook, thank you for your patience and support. It was your love that gave me strength and courage to go through this special journey. Above all, I am so happy that we have just become first-time parents at the time of writing this acknowledgement. I am so proud of my boy, Heechan. I love you so much.

Abstract

Previous research has been done on the information theory of binary random rooted tree models in which every nonleaf vertex in a tree has exactly two children. Let α be a positive integer parameter > 2. The main contribution of this thesis is to extend the information theory results for binary random tree models to α-random tree models, which are random tree models in which up to α children per each nonleaf vertex are allowed. An α-random random tree model consists of a sequence $\{T_n : n = 2, 3, \cdots\}$, in which each T_n is an α-random tree having n leaves, and there is a context-free grammar describing how each T_n is randomly generated. In particular, balanced α-random tree models are examined, meaning that the context-free grammars which are employed satisfy a type of balance condition. We obtain three types of results for α-random tree models, described as follows.

First, given a general α-random tree model $\{T_n\}$, the entropy $H(T_n)$ of each tree T_n is expressed as a recursive formula relating $H(T_n)$ to the entropies of its subtrees. This recursion can be explicitly solved for some random tree models, and we show how this can be done for the binary random search tree model.

Our second set of results, which are our main results, concern the balanced α-random tree model $\{T_n\}$. Defining the entropy rate of T_n as the ratio $H(T_n)/n$, we examine the asymptotic behavior of the entropy rate as n grows without bound. This asymptotic behavior is described via a continuous non-constant periodic function $P_\alpha \to [0, \infty)$, having period one, satisfying the property that $H(T_n)/n = P_\alpha(\log_\alpha n)$ for every n. The graph of P_α is seen as a fractal and its fractal properties along with its fractal dimension are investigated. We develop a direct and indirect way of describing P_α. The direct way expresses the graph of P_α as the attractor of an iterated function system which is explicitly given. The indirect way obtains P_α via a change of variable on a hierarchical entropy function discovered by J. Kieffer.

Our third and final set of results concern the development of compression methods for some α-random tree models $\{T_n\}$. In the case in which each random tree T_n is not equiprobable over the ensemble of trees in which it takes it values, we develop a encoding method via which T_n is uniquely represented by a variable-length binary codeword, so

that the expected codeword length is roughly equal to the entropy $H(T_n)$. In the case of the balanced α-random tree model, each T_n is equiprobable, which allows us to develop encoding methods via which each T_n is uniquely represented by a fixed-length binary codeword of minimal length $\lceil H(T_n) \rceil$. One of the encoding methods discussed employs an one-to-one mapping of the ensemble of T_n into one of the hierarchical type classes discovered by J. Kieffer, allowing Kieffer's method for encoding hierarchical type classes to be adapted to an encoding method for balanced α-random trees.

Contents

List of Tables

List of Figures

Chapter 1

Background

Our first chapter is devoted to the discussion of background material, giving the context for the original results to be obtained as the result of this dissertation.

1.1 Notation and abbreviations

$\|S\|$	For a finite set S, $\|S\|$ means the cardinality of S.
$\mathcal{T}$	A random tree ensemble.
$\|T\|$	The number of leaves of T descending from the root of T.
T^i	The i-th subtree rooted at the i-th child of tree T from the left. Likewise, $\|T^i\|$ is the number of leaves all of which are descending from the i-th child of its root vertex.
T^L,T^R	the left and right subtrees of a random binary tree T are denoted as T^L and T^R, respectively.

1.2 Introduction

With the development of high-speed modern computers, there is an increased demand for storage of a large amount of data in various formats or structures. Here are some examples of such demands.

- Biomedical data including genomic sequence data, gene expression data and multimedia medical data such as transverse CT, MR, and cryosection images

- Dynamics on complex networks e.g., internet traffic statistics and structural information of World Wide Web
- Topography and climate data
- Social data including census, health infrastructure, crime, education, and economic data
- Communication networks

To economize on storage space, compression of data before storage is an important consideration. Previous work in information theory has concentrated on the compression of one-dimensional data sets, each typically modeled as the output of an information source consisting of a random sequence of data samples. This approach has been extended to two-dimensional data (such as image data) and three-dimensional data (volumetric data), the easiest approach being to scan a higher dimensional data set into a one-dimensional data string via a scanning technique (such as the Morton scan) which preserves local structure, and then to compress this one-dimensional string. However, for more complicated forms of data structures, such as data distributed throughout a graph, previous information theory approaches will not achieve the best compression.

There is currently being developed an *information theory of data structures* as an approach to the most efficient compression of structured data. Structured data is viewed as consisting of two parts: (1) an underlying graph structure, and (2) the data which is distributed throughout the structure (as labels on edges and/or vertices of the graph structure). Compression of structured data will involve two computational tasks:

- task (1): compression of the underlying graph structure of the data;
- task (2): compression of the data embedded in the graph structure.

The compressed structured data file will then consist of a binary string decomposable into two parts, the first part of which is the result of compression task (1) and allows one to reconstruct the underlying graph structure, and the second part of which yields the result of compression task (2) and allows one to reconstruct the data that is to be distributed through the graph structure.

Initial efforts in the development of the information theory of data structures have been concerned with the task (1) mentioned above, namely, the compression of graphical structures for data (without consideration of the data itself that is to be distributed throughout the graph, which is to be later considered in task (2)). For example, the work of Szpankowski and Choi [18] has concerned itself with the compression of random Erdös-Rényi graph structures, and the work of Kieffer-Yang-Szpankowski [16] addressed the compression of certain binary random tree structures. Our work in the present thesis is a continuation of the KYS work, extending some of their results to certain nonbinary random tree structures.

In the remaining sections of Chapter 1, we give further background on the previous works described in the preceding, followed by background concerning what will be our original work in the present dissertation.

1.3 Information Theory of Structured Data

In this section, we survey the approach to the information theory of structured data in the research program laid out in 2008 by J. Kieffer of University of Minnesota and W. Szpankowski of Purdue University.

In a structured data source, there can be several measurements of information. The measurements are determined by such factors as: rules by which a data structure is formulated, the context in which each element of a data structure appears, and events which may happen to change a data structure and thereby affect the information conveyed by it. To investigate the behavior of structured data and develop the applications of such data, the first step to be taken is to determine how to properly measure the amount of information conveyed by a data structure. One possible measurement is based on descriptive information, via which probabilities are assigned to possible structures, and information measurement is calculated according to the resulting probability distribution.

As mentioned at the beginning of this thesis, structured data is considered as consisting of two parts. The first part is the structure of the data (think of this as an empty bottle), and the second part is the data conveyed by the structure (think of this as the water in the bottle). The data part is dependent on the structure part and cannot be

thought of as separated from the structure part (as the water without the bottle can not form its shape). This is the most prominent difference from conventional data, which does not have a specific structure other than a sequential stream, modeled via a one dimensional space. (One might also have conventional data, such as image or volumetric data, which is structured as a two dimensional or three dimensional array; however, such data is typically converted to stream form via a scanning procedure.) Corresponding to the two parts of structured data, the measurement of information can also be split into two parts, namely the information I_s conveyed by the structure, and the information I_d conveyed by the data conditioned on the structure. The total amount of information conveyed by the data structure is then $I = I_s + I_d$.

Let us illustrate how the measurements I_s and I_d might be made. Let A be a finite data alphabet. Let $\mathcal{G}^*$ be a finite set of finite graphs. Let $\mathcal{G}_A$ be a set of finitely many finite labeled graphs such that

- Each graph in $\mathcal{G}_A$ is labeled in the sense that each of its vertices carries a data label from A.
- When each graph in $\mathcal{G}_A$ is stripped of its labels, the underlying graph that is obtained belongs to $\mathcal{G}^*$.

The labeled graphs in $\mathcal{G}_A$ are the data structures of interest here. For each data structure G_A chosen from $\mathcal{G}_A$, the first part, the structure part, is the graph in $\mathcal{G}^*$ underlying G_A when all the data labels have been stripped away, and the second part, the data part, consists of the data labels assigned to the vertices of G_A. We suppose that we have specified the following probability distributions:

- A probability distribution on $\mathcal{G}^*$. For each G^* in $\mathcal{G}^*$, we let $P(G^*)$ be the probability assigned to G^*.
- For each $G^* \in \mathcal{G}^*$, a conditional probability distribution on the set of all $G_A \in \mathcal{G}_A$ such that the graph underlying G_A is G^*. We let $Q(G_A|G^*)$ denote the conditional probability of G_A given G^*.

Suppose data structure G_A is chosen from $\mathcal{G}_A$. Let $G^* \in \mathcal{G}^*$ be the graph underlying G_A. Decomposing G_A into its first and second parts, we can then measure the first part

information I_s and the second part information I_d as follows:

$$\begin{aligned} I_s &= -\log_2 P(G^*) \\ I_d &= -\log_2 Q(G_A|G^*) \end{aligned}$$

I_s and I_d have the following interpretation. By information theory, we know that that the information needed to build the first part or structure part G^* of data structure G_A can be stored in a database using roughly I_s bits (i.e., a binary codeword from a prefix set of codewords of length roughly I_s), and that the information needed to insert the second part or data part in the structure G^* can be represented in storage using roughly I_d bits (i.e., a binary codeword from a prefix set of codewords of length roughly I_d). The total amount of storage space needed to reconstruct all of G_A (structure plus data) is then roughly $I_s + I_d$ bits. This method of storing the information needed to reconstruct G_A is called two-step coding.

Example 1.3.1. *We employ the six-letter data alphabet $A = \{a, b, c, d, e, f\}$. Figure 1.1 describes a set $\mathcal{G}^* = \{G_1^*, G_2^*, G_3^*, G_4^*\}$ of four different unlabeled graphs and their probabilities*

$$P(G_1^*) = \frac{3}{8}, \;\; P(G_2^*) = P(G_3^*) = \frac{1}{4}, \;\; P(G_4^*) = \frac{1}{8}.$$

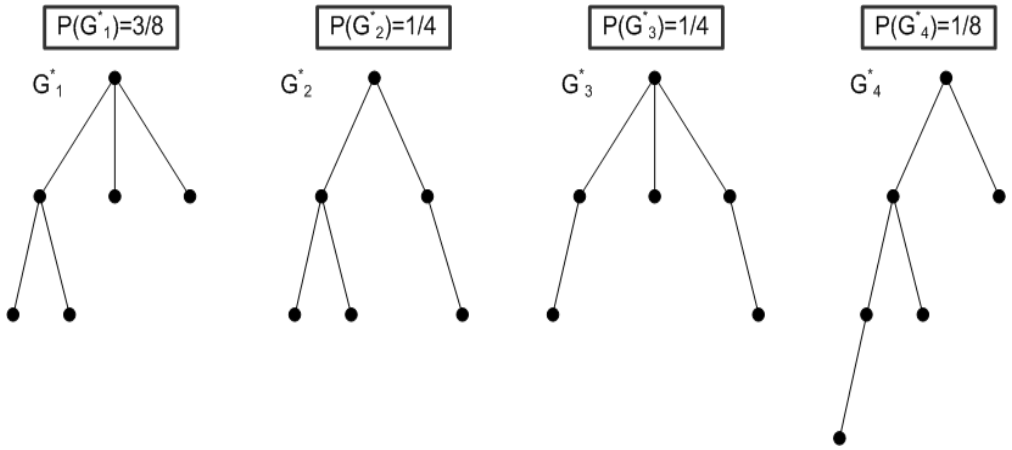


Figure 1.1: Specific unlabeled graphs $\{G_i^*\} \in \mathcal{G}^*$ with probability distribution $\{P(G_i^*) : i = 1, 2, 3, 4$ on $\mathcal{G}^*$.

Figure 1.2 illustrates a set of 11 data structures $\mathcal{G}_A$, each one a graph (in fact, a tree) consisting of six vertices labeled with the letters in the alphabet A. Stripping these data structures of labels to get the structure part, we obtain a set $\mathcal{G}^* = \{G_1^*, G_2^*, G_3^*, G_4^*\}$ in Figure 1.1. We have grouped the data structures $\mathcal{G}_A$ into 4 columns according to the first part structure so that each of them has the same underlying structure part: column i data structures have first part structure G_i^*. In the following, one can see that summing up all probabilities of the labeled graphs conditioned on a fixed underlying graph, the number 1 is obtained.

$$\sum_j Q(G_{A,j}|G_i^*) = 1,\ i = 1, 2, 3, 4.$$

Our principal concern in this thesis being the structure part (the first part) of structured data, we turn our attention to how a structure can be generated. We shall use context-free grammars to generate structures, in particular, structures which are trees. Roughly speaking, a grammar can be viewed as a finite set of simple rules and a data structure is constructed according to this set of rules. In other words, a set of rules can simply describe the (grammar-based) data structures. More concrete descriptions and definitions will be given in a section coming later (Section 1.7).

1.4 Information Theory of Erdös-Rényi Graphs

In this section, we survey the results of Szpankowski and Choi [18] on the information theory of Erdös-Rényi Graphs. An Erdös-Rényi (ER) graph is typically defined according to one of two different models, the $G(n, p)$ model or the $G(n, M)$ model. In the $G(n, p)$ model, $G(n, p)$ represents an undirected random graph with n vertices, in which for any two distinct vertices a random choice is made as to whether or not an edge shall be inserted connecting the two vertices, with an edge inserted with probability p; the selections of the edges are made independently of each other. In the $G(n, M)$ model, $G(n, M)$ represents a random graph selected randomly and uniformly from the set of all undirected graphs with no loops which have n vertices and M edges. The two models have many properties in common and they even coincide under certain conditions. Either way, ER graphs have played an important role in the development of random

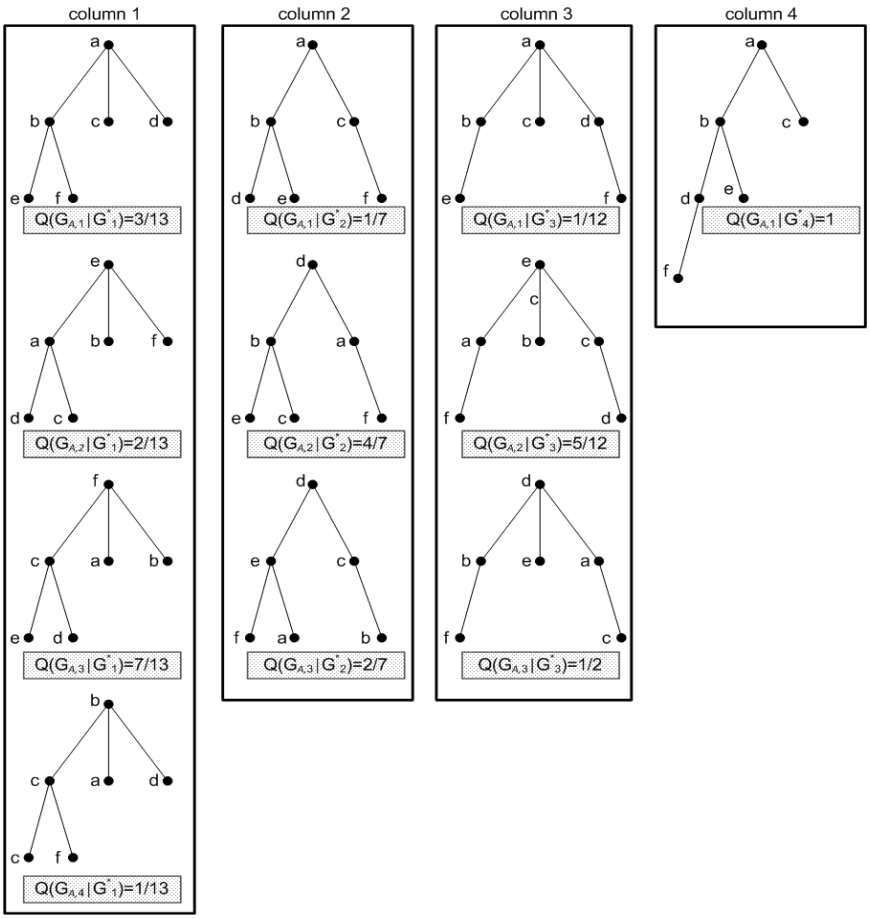


Figure 1.2: Labeled graphs $G_{A,i} \in \mathcal{G}_A$ with conditional probabilities $Q(G_{A,j}|G_i^*)$, $i, j = 1, 2, 3, 4$.

graph theory. Szpankowski and Choi consider what can be concluded about the ER graph model used as a model for the structure part of structured data, that is, they are also taking the approach of this thesis in which one views a data structure as a random labeled graph, and the structure part of the data structure is the unlabeled graph. For the random graph model $G(n, p)$, let $g_1, g_2, \cdots, g_k$ denote the different graph values taken on by the random graph $G(n, p)$. The Shannon entropy $H(G(n, p))$ of random graph $G(n, p)$ is defined as

$$H(G(n,p)) = \sum_{i=1}^{k} - \Pr[G(n,p) = g_i] \log_2 \Pr[G(n,p) = g_i],$$

which is the expected value of the negative of the logarithm of the probability that the random graph takes on its observed value. Szpankowski and Choi use $H(G(n,p))$ as the measure of the amount of structure information conveyed by the $G(n,p)$ model. They find a formula for $H(G(n,p))$ using automorphisms of sets of ER graphs as a way to keep track of structural symmetry, and then they establish the asymptotic expansion

$$H(G(n,p)) = \binom{n}{2} h(p) - n \log n + O(n),$$

valid as the number of vertices n of the ER graph is allowed to grow without bound, where $h(p) = -p \log p - (1-p) \log(1-p)$ is the binary Shannon entropy function. Szpankowski and Choi also propose an algorithm for compressing the ER graphs. The average length of the binary codeword assigned to $G(n,p)$ by their algorithm is shown to be less than or equal to

$$\binom{n}{2} h(p) - n \log n + (c + \Phi(\log n)) + o(1),$$

where $\Phi(\cdot)$ is a fluctuating function and c is an explicitly computable constant.

1.5 Information Theory of Binary Random Rooted Tree Structures

In this section, we survey the results of Kieffer-Yang-Szpankowski (KYS) on the information theory of binary random tree structures. We shall devote a later part of this dissertation to the complete development of this theory.

Definition of the Concept of Tree. In this thesis, by a tree T, other than the trivial case in which T consists of just one vertex, we shall always mean that it is a rooted tree with at least two vertices and finitely many vertices in total. The unique root vertex of T shall simply be called the "root". The vertices of T having no children shall simply be called "leaves". The vertices of T having children (one of which is the root) are called "nonleaf vertices". A pair of vertices (v_1, v_2) of T in which v_2 is one of the children of v_1 shall be called a "parent-child pair". There is a unique edge of T going from parent to child in a parent-child pair, and all edges of T arise in this way. For each vertex of T other than the root, there will exist a unique path from the root to

that vertex consisting of finitely many edges. We shall always suppose that the children of each nonleaf vertex v of T are ordered.

We introduce some further terminology regarding trees for later use. Given a vertex v of a tree T, the terminology "number of leaves descending from v" means the number of leaves of the subtree of T which is rooted at v. The number of leaves descending from v can be marked as a label on the v. Throughout the thesis, a vertex which is assigned a label $n \geq 1$ implies that the vertex has n leaves down from it. We let the number of leaves of a tree T be denoted as $|T|$. If v is a vertex of T, $T(v)$ shall denote the subtree of T rooted at v, and therefore $|T(v)|$ denotes the number of leaves descending from v. The depth of v is the length of a path from the root to v. Therefore, the depth of the root is 0. The set of all the vertices at a fixed depth is a level of T.

Some Types of Trees.

- (a): **The α-ary tree $T_k(\alpha)$.** For each integer $\alpha \geq 2$, and any positive integer k, $T_k(\alpha)$ is called the α-ary tree of depth k if every nonleaf vertex in $T_k(\alpha)$ has exactly α children and $T_k(\alpha)$ has α^k leaves. For instance, a 3-ary tree of depth 4 has 81 leaves and a 4-ary tree of depth 2 has 16 leaves.

- (b): **The bisection trees.** For an integer $n \geq 1$, $\{T_n\}$ is a bisection random tree model having n leaves when it satisfies the following. The root of a bisection tree T_n is labeled with n and every nonleaf vertex has two children. Let i be an integer such that $2 \leq i \leq n$. If the label i on a nonleaf vertex v of T_n is even, the pair of the two labels on the children of the v are $(i/2, i/2)$. If i is odd, the two children of the v have a pair of labels $(\lfloor i/2 \rfloor, \lceil i/2 \rceil)$ or $(\lceil i/2 \rceil, \lfloor i/2 \rfloor)$, each with probability $1/2$.

  Example 1.5.1. *Figure 1.7 illustrates bisection trees with* 7 *leaves. Because the label* 7 *on the root is odd, its two children are labeled with either* $(3, 4)$ *or* $(4, 3)$ *with* 50% *chance. One of the depth* 1 *labels* 3 *splits into the depth* 2 *labels* $(1, 2)$ *or* $(2, 1)$ *with probability* $1/2$. *The other depth* 1 *label* 4 *is even, so it gives the pair of the depth* 2 *labels* $(2, 2)$ *with probability* 1. *The vertices labeled with* 2 *have two children which are both labeled with* 1. *If a label on a vertex is* 1, *the vertex is a leaf and it does not have any children.*

- (c): **The α-balanced trees.** We define a finite rooted ordered tree to be an α-balanced tree if the following hold:
  - Each vertex has $\leq \alpha$ children.
  - If a vertex v has α children, and if the number of leaves descending from v is k, then the number of leaves descending from each of its children is either $\lfloor k/\alpha \rfloor$ or $\lceil k/\alpha \rceil$. Also the numbers of leaves from the children should sum to k.
  - If a vertex has less than α children, then these children are all leaves.

  Example 1.5.2. *Figure 1.3 illustrates a* 3*-balanced tree. It has* 14 *leaves descending from the root. All the nonleaf vertices with labels* ≥ 3 *have three children, whereas the vertices labeled with* 2 *have two children which are both leaves. For the vertices having three children, the labels of the children are either* $\lfloor k/3 \rfloor$ *or* $\lceil k/3 \rceil$ *such that the sum of the labels of the children is equal to the label of their parent vertex: the root has three children whose labels are* 4, 5, *and* 5 *and the vertex labeled with* 4 *has children with labels* 1,1,*and* 2 *from left to right.*

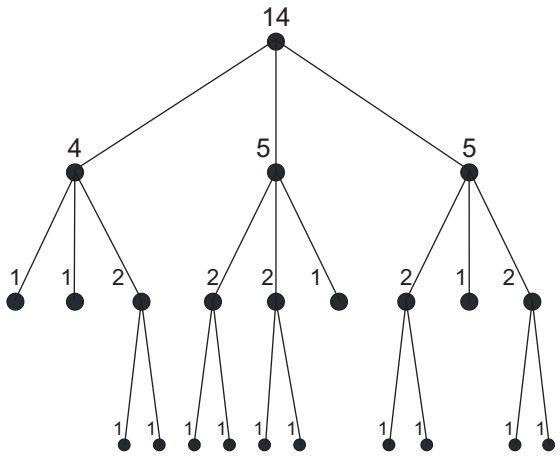


Figure 1.3: 3-balanced trees with 14 leaf vertices

Note that the bisection trees coincide with the α-balanced trees with $\alpha = 2$. Also for $\alpha > 2$, an α-balanced tree T is not α-ary generally, because it does not always have α^k leaves. If the T has n leaves where $n = k^\alpha$ for some integer $k \geq 1$, then T is an

α-ary tree $T_k(\alpha)$. Otherwise, it is not α-ary. The tree T, however, can reduce to an α-ary tree $T_k(\alpha)$ by removing all edges of maximal depth. To get the $T_k(\alpha)$ back to T, make each of the leaves with label $m > 1$ of the α-ary tree have m children, where $m \in \{2, 3, \cdots, \alpha\}$. Each of the children labeled with m then be a leaf of T.

Specifics of a Binary Random Rooted Tree Model. In our random tree models, we typically fix the number n of leaves that are allowed, and are interested in a finite set of trees $\mathcal{T}_n$ in which each tree in $\mathcal{T}_n$ has n leaves. The random tree model can then be viewed as a family of probability distributions $\{P_n\}$, in which, for each possible n, P_n is a certain probability distribution on $\mathcal{T}_n$. For each n, selecting a random tree T_n from $\mathcal{T}_n$ according to P_n, we shall typically use the Shannon entropy $H(T_n)$ of random tree T_n as the measure of the structural information in T_n, and shall be interested in how $H(T_n)$ grows as n becomes large. The entropy $H(T_n)$ is computed as follows. First, let $\mathcal{T}_n$ be the set of trees with n leaves from which random tree T_n is selected. We suppose that a probability distribution has been defined on the set $\mathcal{T}_n$. Thus, each tree $t \in \mathcal{T}_n$ has a probability $P_n(t)$ assigned to it and these probabilities add up to one. The entropy $H(T_n)$ is computed as

$$H(T_n) = \sum_{t \in \mathcal{T}_n} -P_n(t) \log_2 P_n(t), \tag{1.1}$$

where $P_n(t)$ is the probability assigned to tree t (if $P_n(t) = 0$, we adopt the usual convention that $0 \log_2 0 = 0$).

Suppose we have a binary tree t with n leaves, $n \geq 2$. The root of t has two children, which are ordered. We call one of the children the *left child* and the other child the *right child*. Let t^L be the subtree of t rooted at the left child of the root of t, and let t^R be the subtree of t rooted at the right child of the root of t. Since

$$|t^L| + |t^R| = |t| = n,$$

the pair $(|t^L|, |t^R|)$ belongs to the $(n-1)$-point set

$$Q_n = \{(1, n-1), (2, n-2), (3, n-3), \cdots, (n-1, 1)\}. \tag{1.2}$$

Let $\mathcal{T}_1$ be the tree consisting of just one vertex, the root vertex. For $n \geq 2$, let $\mathcal{T}_n$ be the set of all binary trees with n leaves. Since $\mathcal{T}_1$ consists of just one tree, we assign it probability 1, giving us a (trivial) probability distribution P_1 on $\mathcal{T}_1$. Fix $n \geq 2$. We

are going to define a probability distribution P_n on $\mathcal{T}_n$ recursively using the previously defined distributions P_m for $1 \leq m \leq n-1$. First, choose a probability distribution q_n on Q_n. Then, select a random tree in $\mathcal{T}_n$ in the following steps:

- Step 1: Choose (i, j) from Q_n with probability $p_n(i, j)$.
- Step 2: Choose a tree t_1 from $\mathcal{T}_i$ with probability $P_i(t_1)$.
- Step 3: Choose a tree t_2 from $\mathcal{T}_j$ with probability $P_j(t_2)$.
- Step 4: Form the t in $\mathcal{T}_n$ such that $t^L = t_1$ and $t^R = t_2$. The probability assigned to tree t is then

$$P_n(t) = p_n(i, j)P_i(t_1)P_j(t_2). \tag{1.3}$$

Tree t arising from Step 4 is the observed value of random tree T_n taking its values in $\mathcal{T}_n$. We have

$$P_n(t) = \Pr[T_n = t] = q_n(|t^L|, |t^R|)P_i(t^L)P_j(t^R), \quad t \in \mathcal{T}_n.$$

In this way, we have defined a probability distribution P_n on each set of trees $\mathcal{T}_n$ $(n \geq 1)$. The set of probability distributions $\{P_n : n \geq 1\}$ is completely determined by the set of probability distributions $\{p_n : n \geq 2\}$, where for each $n \geq 2$, p_n is an arbitrary probability distribution on Q_n.

Example 1.5.3. *On Q_3, define*

$$p_3(1, 2) = p_3(2, 1) = 1/2.$$

There are then two trees in $\mathcal{T}_3$, each one of which has probability $1/2$. Now on Q_4, define

$$p_4(1, 3) = p_4(2, 2) = p(3, 1) = 1/3.$$

Then there are five trees in $\mathcal{T}_4$, four of which have probability $1/6$ (these are the trees t for which $(|t^L|, |t^R|)$ is either $(1, 3)$ or $(3, 1)$), and the fifth tree has probability $1/3$ (this is the unique tree t for which $|t^L| = 2$ and $|t^R| = 2$). Continuing in this way, assuming that each p_n on Q_n is equiprobable, we get a binary random tree model $\{T_n : n \geq 1\}$ called the binary search tree model, which arises from a problem in computer science whose significance will be discussed later in this thesis.

Computation of $H(T_n)$. Let $\mathcal{T}_n$ be one of our binary random tree models, and let T_n denote a random tree selected from $\mathcal{T}_n$ according to probability distribution P_n. We develop a formula for the entropy $H(T_n)$ of random tree T_n. For a discrete random variable X over finite alphabet χ with probability distribution $Pr(X = x) = p(x),\ x \in \chi$, the Shannon entropy $H(X)$ of X is defined as

$$H(X) = -\sum_{x \in \chi} p(x) \log_2 p(x) = E\left[\log_2 \frac{1}{p(X)}\right].$$

Taking $X = T_n$, we obtain the formula for the entropy $H(T_n)$ given earlier (1.1). Instead, we want a recursive formula for $H(T_n)$ which exploits the dependence of the distribution P_n of T_n on the probability distributions $p_n,\ P_1, P_2, \cdots, P_{n-1}$. By our stepwise construction procedure discussed earlier, T_n is completely determined by and determines a random quadruple (I, J, T_I, T_J), where (I, J) takes its values in Q_n and has distribution p_n, random tree T_I has conditional distribution P_i if $I = i$, and random tree T_J has conditional distribution P_j if $J = j$. If two random variables X and Y are functions of each other, it is well known [14] that $H(X) = H(Y)$. Thus, we have

$$H(T_n) = H((I, J, T_I, T_J)).$$

Given two discrete random variables U, V, the conditional entropy of U given V is

$$H(U|V) = E[-\log_2 P(U|V)],$$

where $P(U|V)$ is the random variable equal to the conditional probability $P(U = u|V = v)$ when $U = u$ and $V = v$. It is well known [14] that

$$H((V, U)) = H(V) + H(U|V).$$

Setting $V = (I, J)$, and $U = (T_I, T_J)$, we then have

$$H(T_n) = H((I, J, T_I, T_J)) = H((I, J)) + H((T_I, T_J)|(I, J)). \tag{1.4}$$

Let $h(p_n)$ denote the entropy of probability distribution p_n, defined by

$$h(p_n) = \sum_{(i,j) \in Q_n} -p_n(i, j) \log_2 p_n(i, j).$$

(In similar fashion, we define the entropy $h(q)$ of any discrete probability distribution $h(q)$.) Then, in (1.4), we may substitute

$$H((I,J)) = h(p_n).$$

It is well known [Cover and Thomas] that conditional entropy may be broken down as

$$H(U|V) = \sum_v P[V=v]H(U|V=v),$$

where the summation is over all values v of V, and $H(U|V=v)$ is obtained by computing the entropy of U under the conditional distribution of U given $V=v$. This gives us

$$H((T_I,T_J)|(I,J)) = \sum_{(i,j)\in Q_n} p_n(i,j)H((T_I,T_J)|(I,J)=(i,j)).$$

If random variables A, B are conditionally independent given random variable C, it is well known [14] that

$$H(A,B|C=c) = H(A|C=c) + H(B|C=c).$$

Therefore, taking $A=T_I$, $B=T_J$, and $C=(I,J)$,

$$H((T_I,T_J)|(I,J)=(i,j)) = H(T_I|(I,J)=(i,j)) + H(T_J|(I,J)=(i,j)).$$

Under the condition $(I,J)=(i,j)$, the conditional distributions of T_I and T_J are P_i and P_j, respectively. Therefore,

$$H(T_I|(I,J)=(i,j)) = h(P_i)$$

$$H(T_J|(I,J)=(i,j)) = h(P_j).$$

Substituting back into (1.4), we conclude that $H(T_n)$ can be computed recursively via the formula

$$H(T_n) = h(P_n) = h(p_n) + \sum_{(i,j)\in Q_n} q_n(i,j)[h(P_i)+h(P_j)], \quad n \geq 2. \tag{1.5}$$

We emphasize that the entropy $H(T_n)$ can be interpreted as the approximate length of a binary codeword into which the "tree data structure" T_n can be encoded, such that

T_n can be decoded from this codeword. We explain later on how this can be done (See Example 4.2.1 in Chapter 3).

By making different choices for the distributions $\{p_n : n \geq 2\}$, one obtains different binary random tree models $\{T_n : n \geq 1\}$, and in each case, one can use the recursion (1.5) to either directly solve for $H(T_n)$ or determine the asymptotic behavior of $H(T_n)$ as $n \to \infty$ (or both). We will be examining thoroughly two different choices of binary random tree model, namely, the 2-*balanced tree model* and the *random binary search tree model*. We consider this latter model first, in the following.

1.5.1 Random Binary Search Tree Model

In the example 1.5.3, we examined the distribution of random trees T_3 and T_4 in the so-called random binary search tree model $\{T_n : n \geq 1\}$, which arises from the choice of p_n's in which each p_n is equiprobable on Q_n. In the present section, we give a thorough discussion of this model. Binary search trees are used over a wide range of computer science applications. We explain how a binary search tree arises. Suppose we have a set of keys $\{1, 2, \cdots, k\}$. One observes a random ordering $(i_1, i_2, \cdots, i_k)$ of $(1, 2, \cdots, k)$, with each ordering assigned a probability of $1/k!$. Here is how we form a binary search tree from an ordering $(i_1, \cdots, i_k)$. The search tree is grown as one processes $(i_1, i_2, \cdots, i_k)$ from left to right. First, grow the root of the tree and label it i_1. Then, examine i_2. If $i_2 > i_1$, grow a right edge from the root and label the vertex at the end of this edge i_2; otherwise, $i_2 < i_1$, and you grow a left edge from the root and label the vertex at the end of this edge i_2. Suppose $i_1, \cdots, i_m$ have been processed, where $m < k$. Then at this point, we have a tree with m vertices and $m - 1$ edges, with the vertices labeled $i_1, \cdots, i_m$. Compare i_{m+1} to the root label. If i_{m+1} is bigger than the root label, do the following:

- If there is not a right edge from the root, grow such an edge and label the vertex at the end of the edge with label i_{m+1};
- If there is a right edge from the root, follow it to the vertex at the end of the edge, and then compare i_{m+1} with the label on this vertex, moving further down in the tree from this vertex as was already described for the root vertex. You will eventually arrive at a previously labeled vertex such that when you try to move

along a right or left edge from that vertex based on the comparison of i_{m+1} with the label at that vertex, you find that edge not to be present. Then you grow that edge and label its terminal vertex with i_{m+1}.

If i_{m+1} is smaller than the root label, do the following:

- If there is not a left edge from the root, grow such an edge and label the vertex at the end of the edge with label i_{m+1};
- If there is a right edge from the root, follow it to the vertex at the end of the edge, and then compare i_{m+1} with the label on this vertex, moving further down in the tree from this vertex as was already described for the root vertex. You will eventually arrive at a previously labeled vertex such that when you try to move along right or left edge from that vertex based on the comparison of i_{m+1} with the label at that vertex, you find that edge not to be present. Then you grow that edge and label its terminal vertex with i_{m+1}.

Following the preceding construction algorithm, in processing each key in $(i_1, i_2, \cdots, i_k)$ left to right beyond the first entry, exactly one edge is appended to the tree you are growing and the label on the vertex at the end of that edge will be the key you are currently processing; this labeling procedure is called "key insertion". At the end of the processing of the keys in $(i_1, \cdots, i_k)$, you will have a tree with $k-1$ edges and k vertices, with the vertices labeled $i_1, \cdots, i_k$. You then append $k+1$ edges to this tree to obtain a binary tree with $k+1$ leaves, since the vertices labeled with the keys must not be leaves; this final tree is the binary search tree yielded by key permutation $(i_1, \cdots, i_k)$. Let $\mathcal{T}_{k+1}$ be the set of all binary trees with $k+1$ leaves. For each tree t in $\mathcal{T}_{k+1}$, define $P_{k+1}(t)$ to be the total probability of all the key permutations $(i_1, \cdots, i_k)$ which yield binary search tree t (which is the number of such permutations times $1/k!$). It turns out that probability distribution P_{k+1} is precisely the same as the probability distribution on $\mathcal{T}_{k+1}$ in the random binary search tree model. We illustrate this fact for $k = 4$. In Figure 5.9 of [15] (which is given in Figure 1.4), it is shown that the three key permutations 3214, 3214, 3421 (and no others) all yield the same binary search tree, namely, the unique tree $t \in \mathcal{T}_5$ for which

$$|t^L| = 3, \;\; |t^R| = 2, \;\; |(t^L)^L| = 2, |(t^L)^R| = 1.$$

Therefore, the P_5 probability of t should be $3/4! = 1/8$. This checks out because:

$$P_5(t) = p_5(3,2)P_3(t^L)P_2(t^R) = (1/4)(1/2)(1) = 1/8.$$

Note that each p_5 probability is $1/4$ because Q_5 is of size 4.

Example 1.5.4. *Figure 1.5 illustrates the binary search trees generated by the six ordered sequences of the keys* $1, 2, 3$. *Table 1.1 gives these six key sequences and the search tree from Figure 1.5 generated in each case. The equal probabilities* $1/6$ *for the six key sequences are used to compute the probabilities of the five search trees as follows:*

$$P_4(①) = P_4(②) = P_4(④) = P_4(⑤) = \frac{1}{6}, \;\; P_4(③) = \frac{1}{3}.$$

From Example 1.5.3, the reader see that this probability distribution P_4 *is the same one as the probability distribution of random tree* T_4 *in the random binary search tree model* $\{T_n : n \geq 1\}$ *defined earlier.*

Search Tree From Figure 1.5	*Key Sequence*
(1,2,3)	①
(1,3,2)	②
(2,1,3) *(2,3,1)*	③
(3,1,2)	④
(3,2,1)	⑤

Table 1.1: Key sequences and search trees (3 Keys)

The following theorem giving the entropy of the random trees in the random binary search tree model was stated in the KYS paper without proof. Theorem 1.5.5 which follows gives the entropy of the random trees in the random binary search tree model and was stated in the KYS paper without proof. We give a proof in Appendix A of this thesis by mathematical induction.

Theorem 1.5.5. *Let* $\{T_n : n \geq 1\}$ *be the random trees in the random binary search tree model. Then* $H(T_1) = H(T_2) = 0$ *and*

$$H(T_n) = 2n \sum_{i=1}^{n-2} \frac{\log_2 i}{(i+1)(i+2)} + \log_2(n-1), \;\; n \geq 3.$$

Remark. For the random binary search tree model $\{T_n\}$, we see from the previous result that

$$\lim_{n\to\infty} H(T_n)/n = 2\sum_{i=1}^{\infty} \frac{\log_2 i}{(i+1)(i+2)} \approx 1.736.$$

Let us interpret what this statement means. Suppose n is large. Then $H(T_n) \approx (1.736)n$. Assume that each day one observes a key sequence of a randomly selected key sequence using $n-1$ keys, computes the resulting binary search tree, and then stores this tree in bits using the most efficient compression algorithm (the reason being that at some future time one may want to know what binary search tree was recorded on a given day). Averaged over many days, the amount of storage space in bits used per day is then approximately $(1.736)n$ bits. In the paper KYS, a binary random tree model $\{T_n\}$ is called *stable* if $\lim_{n\to\infty} H(T_n)/n$ exists. Thus, we have shown that the random binary search tree model is stable. Suppose we have a general binary random tree model induced by the sequence of distributions $\{p_n : n \geq 1\}$. It is an open problem to find a workable necessary and sufficient condition on the $\{p_n\}$ sequence so that the resulting binary random tree model will be stable.

1.5.2 Bisection Random Tree Model

We have just discussed the random binary search tree model introduced in the KYS paper, and extended the KYS treatment of this model by proving the formula announced in KYS for the entropies of the random trees in the model. One of the other random tree models discussed in KYS is the bisection random tree model defined in earlier in Section 1.5. In this subsection, we extend the results in KYS on the bisection random tree model. Let us abbreviate the entropy $H(T_n)$ as $H(n)$. Throughout the thesis, subscript 2 will be added to the $H(n)$ such as $H_2(n)$ for the entropy of a bisection tree. KYS prove the entropy recursion

$$H_2(n) = \begin{cases} 2H_2(n/2), & n \text{ even} \\ 1 + H_2(\lfloor n/2 \rfloor) + H_2(\lceil n/2 \rceil), & n \text{ odd.} \end{cases} \tag{1.6}$$

This recursion is valid for all $n \geq 2$. Since $H_2(1) = 0$, the recursion allows every entropy $H_2(n)$ to be computed. KYS announced without proof that there exists a continuous periodic function P_2 on the real line, with period 1, such that

$$H_2(n)/n = P_2(\log_2 n), \quad n \geq 1. \tag{1.7}$$

From Figure 1.6, the reader sees that the function P_2 is nonconstant. Therefore the sequence $\{H_2(n)/n\}$ does not have a limit, but instead will converge to different limits along different subsequences, and using formula (1.7) we can easily choose such subsequences. For example, the point $(t, P_2(t)) \approx (0.3, 0.42)$ tells us that if we take the number of leaves n of a bisection tree to be of the form $\lfloor 2^{k+0.3} \rfloor$ for k sufficiently large, then $H_2(n)/n$, the logarithm of the number of such trees divided by the number of leaves, is roughly 0.42, which in a data compression interpretation tells us that each such tree can be uniquely represented by a binary codeword whose length is roughly 42% of the number of leaves. On the other hand, taking $(t, P_2(t)) \approx (0.415, 0.5)$, if we take the number of leaves n of a bisection tree to be $\lfloor 2^{k+0.415} \rfloor$ for k sufficiently large, then the codeword length will be relatively longer, namely, about 50% the number of leaves.

Compression Rate Curve C_2. The function P_2 is periodic with period 1. The plot of one period of the graph of the function P_2 over the interval $[0, 1]$ is called compression rate curve C_2 and is given as

$$C_2 = \{(t, y) : 0 \leq t \leq 1,\ y = P_2(t)\}. \tag{1.8}$$

The plot C_2 is fractal in the sense that it has infinitely magnifiable character [11]: C_2 shows the same shape with any resolutions.

The bisection tree model, although not a stable model like the random binary search tree model, is one of a class of models $\{T_n\}$ called *oscillatory models*, meaning that there exists a periodic function P_2 such that the random tree entropies behave asymptotically as

$$H_2(T_n) = nP_2(\log_2 n) + o(n), \tag{1.9}$$

where the notation $o(n)$ means an error term such that $o(n)/n$ converges to zero as $n \to \infty$. Although oscillatory behavior of the type exhibited on the right side of (1.9) may seem unusual on one's first exposure to it, such oscillatory behavior is often seen in the area of computer science known as *analysis of algorithms*. Various researchers in analysis of algorithms have explained why oscillatory behavior often occurs; for one good explanation see Problem 9.5 on page 436 of [17].

In Appendix A, we prove Theorem 1.5.6 below, which implies that the bisection random tree model is oscillatory. Consider all pairs (i, j) of non-negative integers for which the sum $i+j$ is of the form 2^k for some $k \geq 0$. For each such pair (i, j), we define a non-negative real number $H^*(i, j)$ as follows. If (i, j) is either $(1, 0)$ or $(0, 1)$, define $H^*(i, j) = 0$. Otherwise, define $H^*(i, j)$ by recursion as

$$H^*(i,j) = \begin{cases} 2H^*(i/2, j/2), & i \text{ even} \\ 1 + H^*(\lfloor i/2 \rfloor + 1, \lfloor j/2 \rfloor) + H^*(\lfloor i/2 \rfloor, \lfloor j/2 \rfloor + 1), & i \text{ odd.} \end{cases}$$

In the paper [1], J. Kieffer showed that there is a non-constant continuous function $h_2 : [0, 1] \to [0, 1]$ such that

$$h_2(i/2^k) = H^*(i, 2^k - i)/2^k, \;\; k = 0, 1, \cdots ; i = 0, 1, \cdots , 2^k.$$

Theorem 1.5.6. *Let P_2 be the unique continuous periodic function on the real line, with period 1, such that*

$$h_2(x) = (x + 1)P_2(\log_2(x + 1)), \;\; 0 \leq x \leq 1.$$

Then (1.7) holds.

1.6 Hierarchical Type Classes

Fix an integer $\alpha \geq 2$, and let k be any positive integer. Think of a picture of tree $T_k(\alpha)$ with the root at the top and the leaves at the bottom, with the α^k leaves ordered from left to right in the picture. Let x be a binary string of length α^k, and place the entries of x as labels on the leaves of $T_k(\alpha)$; the i-th entry of x is placed as label on the i-th leaf of $T_k(\alpha)$ in the leaf ordering. The *hierarchical type class* generated by x is the set of strings obtained by permuting the entries of x according to the isomorphisms of tree $T_k(\alpha)$ into itself; that is, each isomorphism of $T_k(\alpha)$ produces a permutation of its leaves and therefore a permutation of its leaf labels, yielding a permutation of the entries of x. For example, if $\alpha = 2$, the hierarchical type class generated by string 0110 is the set $\{0110, 0101, 1001, 1010\}$. Here is why the notion of a hierarchical type class is useful for the present thesis. Later on, we will be studying compression for the class of so-called *α-balanced trees* having a fixed number of leaves. We will find it useful that

this class of trees is in one-to-one correspondence with a certain hierarchical type class. Therefore, we can compress these trees by instead compressing the hierarchical type class strings corresponding to these trees, using compression methodology for hierarchical type classes put forth by Kieffer [1], [2]. Also Kieffer has investigated the asymptotic compression rate performance that is achieved as the size of the hierarchical type class is allowed to grow without bound; this will as a special case give us the asymptotic compression rate performance in compressing α-balanced trees. In the following two examples and later on in this thesis, we let $\mathcal{S}_\alpha(x)$ denote the hierarchical type class generated by string x of length α^j for some $j \geq 1$.

Example 1.6.1. *We illustrate the one-to-one correspondence between the set of four bisection trees having* 7 *vertices and the hierarchical type class*

$$S = \mathcal{S}_2(AAAB) = \{AAAB, AABA, ABAA, BAAA\}.$$

One starts by finding the powers of two that 7 *is caught between, namely, we have*

$$4 \leq 7 \leq 8.$$

Form the ordered pair $(7-4, 8-7) = (3, 1)$*, whose entries sum to* 4*, a power of two. For each string* x *in the class* S*, there is a simple algorithm via which one can label each vertex of tree* $T_2(2)$ *with a pair of non-negative integers* (i, j) *summing to a power of two; the root (depth zero) label is* $(i, j) = (3, 1)$*, the depth one* (i, j) *labels sum to* 2*, and the* (i, j) *leaf labels (depth two) sum to* 1*. From this labeling of* $T_2(2)$*, one uniquely builds a bisection tree, which is the bisection tree corresponding to* x*. We illustrate the procedure for* $x = ABAA$*. Think of* A *as representing the* (i, j) *pair* $(1, 0)$ *and* B *as representing the* (i, j) *pair* $(0, 1)$*. Label the four leaves of* $T_2(2)$ *left to right with* $(1, 0)$*,* $(0, 1)$*,* $(1, 0)$*,* $(1, 0)$*, representing the members of string* $ABAA$*.* (i, j) *vertex labels are then propagated upward in the tree via vector sums. Thus, the left to right labels on the depth one vertices are the vector sums*

$$(1, 0) + (0, 1) = (1, 1), \quad (1, 0) + (1, 0) = (2, 0),$$

and the vector sum $(1, 1) + (2, 0) = (3, 1)$ *is indeed the the root label. Here is how the labeled tree* $T_2(2)$ *we have just formed represents a unique bisection tree with* 7 *leaves,*

which can be grown downward from the root. The root label of this bisection tree is 7*. The two labels on the depth one vertices are either* $3, 4$ *or* $4, 3$*. As we did for* 7*, writing*

$$2 \leq 3 \leq 4, \quad 2 \leq 4 \leq 4,$$

we represent $3, 4$ *as the pairs*

$$3 \leftrightarrow (3-2, 4-3) = (1, 1), \quad 4 \leftrightarrow (4-2, 4-4) = (2, 0).$$

Since the left-to-right depth one labels on $T_2(2)$ *are* $(1, 1)$*,* $(2, 0)$ *and not* $(2, 0)$*,* $(1, 1)$*, we see that the labels at depth one on the bisection tree are respectively* $3, 4$*. The subtree of the bisection tree rooted at the depth one vertex labeled* 4 *is completely determined, since* 4 *is a power of two. The vertex labeled* 3 *has children labels either* $1, 2$ *or* $2, 1$*. Since*

$$1 \leq 1 \leq 2, \quad 1 \leq 2 \leq 2,$$

we have the representations of $1, 2$ *as the pairs*

$$1 \leftrightarrow (1-1, 2-1) = (0, 1), \quad 2 \leftrightarrow (2-1, 2-2) = (1, 0).$$

The children of vertex $(1, 1) = 3$ *in the* $T_2(2)$ *tree have labels* $(1, 0)$*,* $(0, 1)$ *in that order, so these children are labeled in the bisection tree as* $2, 1$ *and not* $1, 2$*. The bisection tree can now be uniquely completed since* 2 *is a power of two, and label* 1 *indicates a leaf. In Figure 1.7 which follows, we have written down the four bisection trees and the hierarchical type class strings corresponding to them.*

Example 1.6.2. *Figure 1.3 gives a* 3*-balanced tree having* 14 *leaves. We illustrate how the set of all* 3*-balanced trees with* 14 *leaves is in natural one-to-one correspondence with the hierarchical type class* $\mathcal{S}_3(BBAAABABA)$*. Let* $T_2(3)$ *be the tree obtained from the Figure 1.3 tree by removing the edges at the bottom level. If a* 3*-balanced tree has* n *leaves, we can write one of the following two relations:*

$$3^k \leq n < 2(3^k), \quad 2(3^k) < n < 3^{k+1}.$$

In the first case, the (i, j) *pair corresponding to* n *is*

$$(i, j) = (n - 3^k, 2(3^k) - n),$$

and in the second case it is

$$(i, j) = (n - 2(3^k), 3^{k+1} - n).$$

In this particular case, $n = 14$ *is the root label of the* $T_2(3)$ *tree, and we have* $9 < 14 < 2(9) = 18$*, so the corresponding* (i, j) *pair is* $(5, 4)$*. All vertex labels in the* $T_2(3)$ *tree at depth one are similarly represented via an* (i, j) *pair in which we squeeze the label between* $9/3 = 3$*and* $18/3 = 6$*. Finally, all vertex labels of the leaves of the* $T_2(3)$ *tree (depth two) are represented by an* (i, j) *pair in which we squeeze the label between* $9/9 = 1$ *and* $18/9 = 2$*. The labels on the leaves of the* $T_2(3)$ *tree are each either* 1 *or* 2*. Squeezing each of these between the lower and upper bounds* 1, 2*, we have corresponding* (i, j) *pairs*

$$1 \leftrightarrow (1 - 1, 2 - 1) = (0, 1), \quad 2 \leftrightarrow (2 - 1, 2 - 2) = (1, 0).$$

The sequence of leaf labels

$$1, 1, 2, 2, 2, 1, 2, 1, 2$$

then converts into

$$(0, 1), (0, 1), (1, 0), (1, 0), (1, 0), (0, 1), (1, 0), (0, 1), (1, 0).$$

Representing $(1, 0)$ *as* A *and* $(0, 1)$ *as* B*, we conclude that the Figure 1.3 tree maps into the string* $BBAAABABA$ *and therefore that the set of all* 3*-balanced trees with* 14 *leaves is in one-to-one correspondence with the hierarchical type class* $\mathcal{S}_3(BBAAABABA)$*.*

1.7 Grammar-Based Structured Random Trees

In this section, we describe grammar-based structured random trees, the special type of structured random tree which is the subject of this dissertation. A structured tree is a finite rooted ordered tree in which for each nonleaf vertex there are constraints placed

- (i): on the number of tree leaves descending from this vertex;
- (ii): on the number of children of this vertex;
- (iii): on the number of leaves descending from each ordered child of this vertex.

We will model constraints of the type (i)-(iii) via production rules of a stochastic context-free grammar. For example, a production rule of the grammar of the form

$$n \to (n_1, n_2, \cdots, n_k), \tag{1}$$

where left member n and right member entries $n_1, \cdots, n_k$ are positive integers with $n = n_1 + \cdots + n_k$, could mean the presence of a nonleaf vertex in the tree in which (i) there are n leaves descending from it, (ii) k children of the nonleaf vertex, and (iii) the number of leaves descending from the i-th ordered child is n_i ($i = 1, 2, \cdots, k$). These production rules, together with probabilities that each of them is employed, constitute a stochastic context-free grammar (we give a formal definition of this concept later in this section). Given such a grammar G, we can randomly construct a tree with n leaves from the root downward to the leaves as follows. First, one randomly selects a production rule of G with left member n from all production rules of G having left member n. If this is rule (1), draw k children of the root labeled $n_1, n_2, \cdots, n_k$ respectively. If a child label $n_i = 1$, it is a leaf; otherwise, $n_i > 1$ and one randomly chooses another production rule of G with left member n_i to determine the subtree structures of that child and its children, independently of the previously selected production rules. Continuing in this way, one will eventually have randomly generated a finite rooted tree whose leaves are each labeled with the terminal symbol 1, and whose nonleaf vertices are each labeled with an integer label > 1, indicating that a production rule of G whose left member is that label was used to propagate the children of that vertex. This tree will be a particular structured tree belonging to the ensemble of all possible trees constructible via G.

Definition of Context-Free Grammar. A context-free grammar (CFG) is defined as a quadruple

$$G = (M, N, S, R),$$

where

- M, N are disjoint nonempty finite sets, with the elements of M called terminal symbols and the elements of N called non-terminal symbols (or variables).
- S is a fixed element of N called the start symbol.

- R is the finite set of production rules of grammar G. Each production rule is of the form

$$v \to (v_1, v_2, \cdots, v_k), \tag{2}$$

where v is non-terminal, $k \geq 2$, and each $v_i \in M \cup N$. v is called the left member of production rule (2), $(v_1, v_2, \cdots, v_k)$ is called the right member, and k, the number of terms in the right member, is a positive integer which can vary from production rule to production rule. To avoid potentially bothersome situations, we require that for each $v \in N$, there exists at least one production rule in R whose left member is v.

There are more general definitions of context-free grammar in the literature. For example, some authors allow the right member of a production rule to be an empty string. We have adopted the preceding definition because it is sufficiently general for the purposes of this thesis.

Definition of Stochastic Context-Free Grammar. A CFG $G = (M, N, S, R)$ becomes a stochastic context-free grammar (SCFG) if to each production rule (2) in R a probability in $[0, 1]$ is assigned, so that for each $v \in N$, the probabilities assigned to the production rules with left member v sum to 1. If production rule of form (2) has been assigned probability p, we shall use the notation

$$v \xrightarrow{p} (v_1, v_2, \cdots, v_k)$$

to denote this.

Random Structured Tree Induced by SCFG. Let n be a fixed positive integer ≥ 2. Consider any SCFG $G_n = (M, N, S, R)$ in which $N = \{2, 3, \cdots, n\}$, $M = \{1\}$, and $S = n$. Then G_n induces a random structured tree $T(G_n)$ with n leaves. We can think of a particular realization $T^*(G_n)$ of random tree $T(G_n)$ as being pictorially generated as follows. Draw the root vertex v of $T^*(G_n)$ and assign it the label n. Randomly select production rule

$$n \xrightarrow{p} (n_1, \cdots, n_k)$$

of G_n with left member n. Below v, draw k children $v_1, v_2, \cdots, v_k$ of v ordered left to right in the picture, and draw edges from v to each of its children. Child v_i is assigned label n_i, $i = 1, 2, \cdots, k$. If a child v_i has label 1, then it is a leaf of $T^*(G_n)$. Otherwise,

for each child v_i with label > 1, repeat the above construction procedure to obtain its children. That is, randomly select a production rule with left member n_i, and from the right member of this production rule, you will see how many children v_i has and what their assigned labels should be. Inductively, repetitions of this procedure will produce the picture of a tree $T^*(G_n)$ in which every leaf has label 1, and it can be proved that there will be exactly n leaves. (In fact, the label on each vertex v of $T^*(G_n)$ will be the number of leaves descending from v, that is, the number of leaves of the subtree of $T^*(G_n)$ which is rooted at v. In particular, the root of $T^*(G_n)$ having label n means that there are n leaves of $T^*(G_n)$, and the leaves of $T^*(G_n)$ are precisely the vertices which are labeled 1.) One can now remove the labels from the vertices of $T^*(G_n)$ to complete the pictorial representation of $T^*(G_n)$. The probability assigned to realization $T^*(G_n)$ of random tree $T(G_n)$ is defined to be

$$P[T(G_n) = T^*(G_n)] = P[T^*(G_n)] = \Pi_v p_v^*,$$

where v ranges through the nonleaf vertices of $T^*(G_n)$ and p_v^* is the probability of the production rule selected in forming the children of v and their labels. (Note that from tree $T^*(G_n)$, one can reconstruct precisely which production rule of G_n was used for each nonleaf vertex of $T^*(G_n)$ to obtain its children and their labels, and therefore each probability p_v will be uniquely determined from knowledge of $T^*(G_n)$.) Let $\mathcal{T}(G_n)$ be the ensemble of all realizations $T^*(G_n)$ of random tree $T(G_n)$. It can be shown that

$$\sum_{T^*(G_n) \in \mathcal{T}(G_n)} P[T(G_n) = T^*(G_n)] = 1,$$

so that we have a legitimate random tree model defined over the ensemble $\mathcal{T}(G_n)$.

Example 1.7.1. *Let $n = 23$. One of the trees that one obtains as a realization of the random tree induced by SCFG G_{23} is the tree $T(G_{23})$ in Figure 1.8. Start symbol S of the grammar is 23, which is the root vertex label and indicates that the tree $T(G_{23})$, when constructed via the SCFG, will have 23 leaves. Each nonleaf vertex with label in the set $\{23, 5, 9, 6, 3, 2\}$ indicates that a production rule of the SCFG whose left member is this label is being used to propagate the children of that vertex. Each leaf vertex is labeled 1, the terminal symbol of the SCFG. In the following, we have listed the production rules*

of the SCFG G_{23}.

$$
\begin{gathered}
23 \xrightarrow{\frac{1}{3}} (5,9,6,3) \\
23 \xrightarrow{\frac{1}{3}} (4,1,7,11) \\
23 \xrightarrow{\frac{1}{3}} (5,5,4,9) \\
11 \xrightarrow{\frac{1}{2}} (2,3,3,3) \\
11 \xrightarrow{\frac{1}{2}} (3,3,2,3) \\
9 \xrightarrow{\frac{5}{7}} (2,1,5,1) \\
9 \xrightarrow{\frac{2}{7}} (2,2,1,1) \\
7 \xrightarrow{\frac{2}{5}} (1,3,1,2) \\
7 \xrightarrow{\frac{2}{5}} (2,3,1,1) \\
7 \xrightarrow{\frac{1}{5}} (1,2,3,1) \\
6 \xrightarrow{1} (2,2,1,1) \\
5 \xrightarrow{\frac{1}{3}} (1,2,1,1) \\
5 \xrightarrow{\frac{2}{3}} (1,1,1,2) \\
4 \xrightarrow{1} (1,1,1,1) \\
3 \xrightarrow{1} (1,1,1) \\
2 \xrightarrow{1} (1,1).
\end{gathered}
$$

These production rules allow us to compute the probability that the particular tree $T(G_{23})$ gets chosen out of the ensemble of trees $\mathcal{T}(G_{23})$ as

$$
\begin{aligned}
Pr\left[T(G_{23})\right] &= \left(\tfrac{1}{3}\right)^2 \cdot \tfrac{5}{7} \cdot (1)^4 \cdot \tfrac{2}{3} \cdot (1)^3 \\
&= \tfrac{10}{189}.
\end{aligned}
$$

Discussion. Some of the previous works on using grammars for compression of data strings are [4]-[9]. Rather than the direct compression of a data string, the approach in these works is to compress the rules of a context-free grammar from which the data string can be obtained, which can allow the data string to be compressed more efficiently. A data string might have hierarchical patterns within it which can be modeled via production rules of a grammar. Various ways have been studied via which a grammar uniquely generating a data string can be built up via an examination of all

such hierarchical patterns. There have been some initial efforts [12] in extending this grammar-based approach from the compression of data strings to the compression of data structures. In this approach, given a data structure on an underlying graph, one would construct a "graph grammar" uniquely generating that graph structure and all the data within it; one would then indirectly compress the data structure by compressing the rules of this graph grammar. Instead, we have taken an alternate approach to data structure compression, namely, we start with a stochastic context-free grammar and then encoder and decoder use this SCFG as a tool in the compression of any of the data structures constructible via the grammar. What we learn from this thesis is that there is a particular type of SCFG which, for data compression purposes, effectively models the class of α-balanced trees. Ultimately, one would want to learn about types of grammars which effectively model other classes of data structures going beyond classes of trees; this is a potentially fruitful area for future research.

In the next chapter, we learn more about balanced trees. In particular, we see how they arise from a particular type of SCFG.

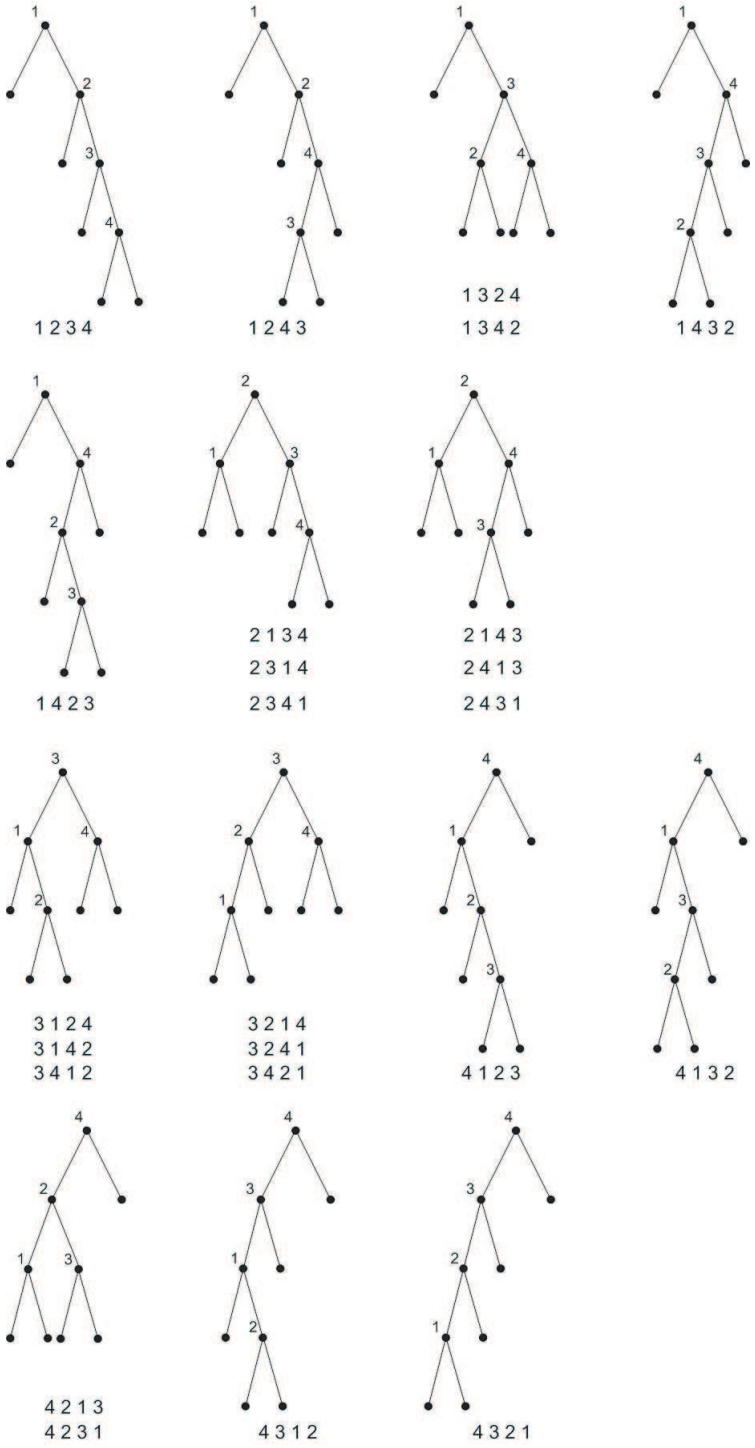


Figure 1.4: Binary search trees given in Figure 5.9 of [15].

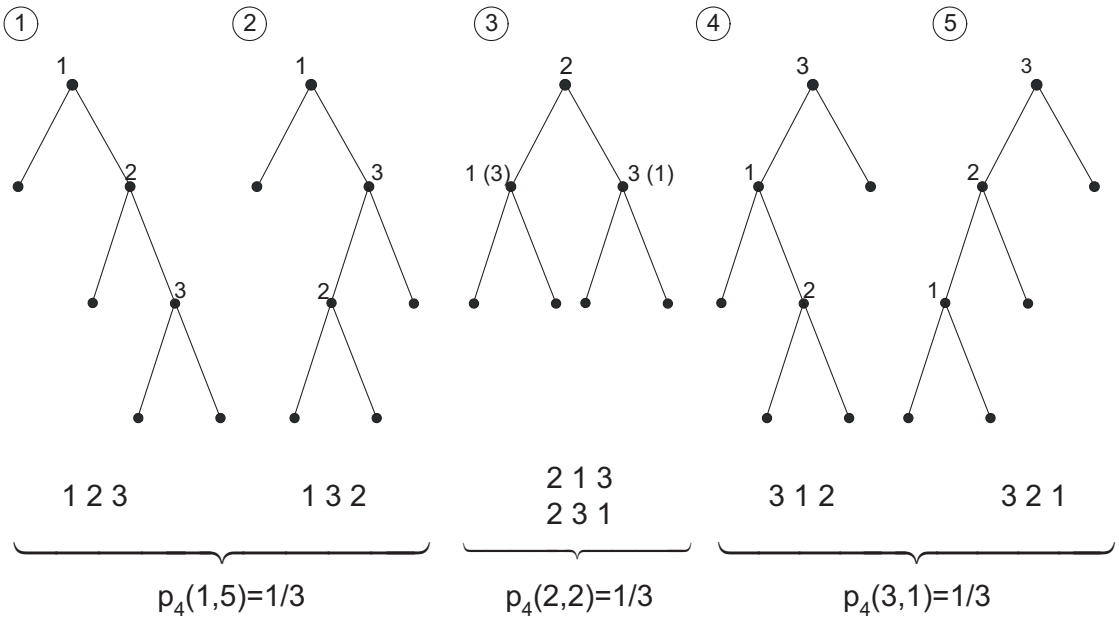


Figure 1.5: Binary search trees with probability distribution.

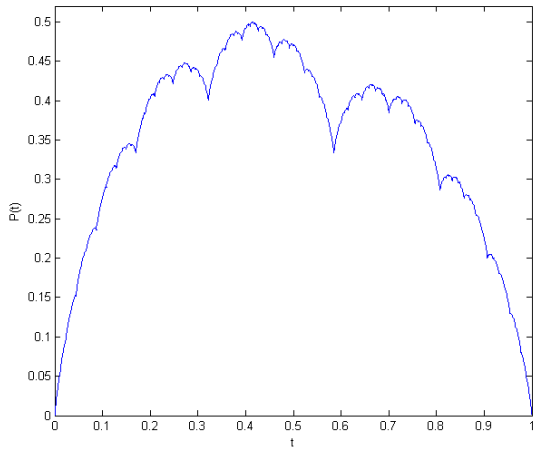
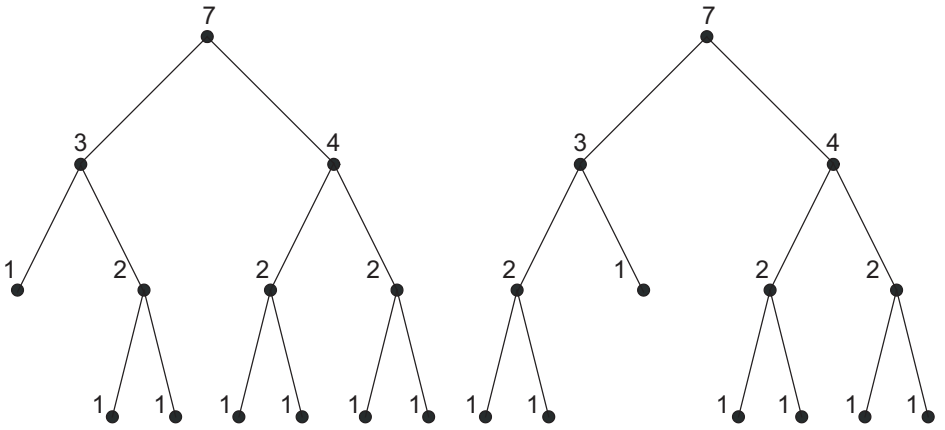
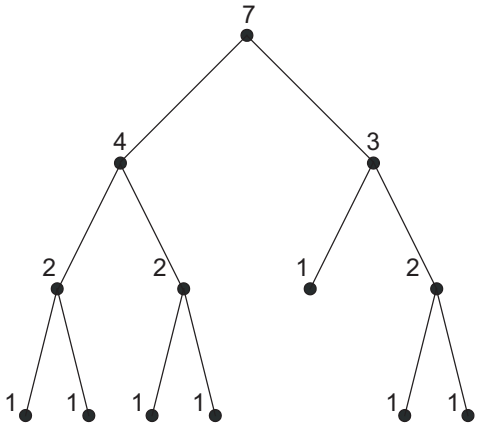


Figure 1.6: One period of $P_2(t)$ function such that $H_2(n) = nP_2(\log_2 n)$.

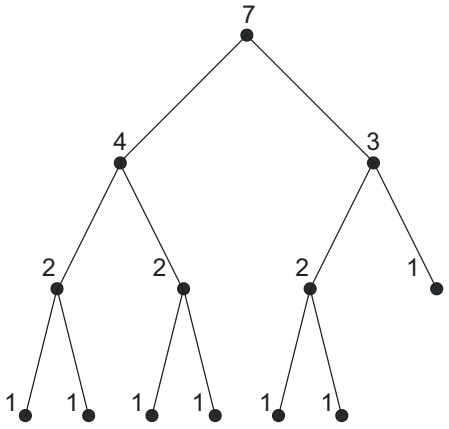


(a) Tree represented by $BAAA$

(b) Tree represented by $ABAA$



(c) Tree represented by $AABA$



(d) Tree represented by $AAAB$

Figure 1.7: Four bisection trees with hierarchical type class strings

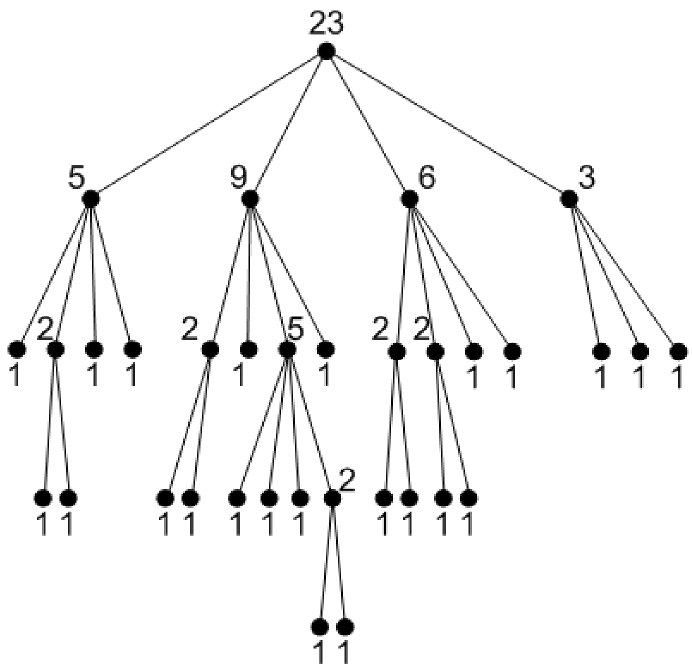


Figure 1.8: A random tree realization $T(G_{23})$ induced by SCFG G_{23} (See Example 1.7.1). The label on each vertex indicates the number of leaves descending from the vertex.

Chapter 2

Balanced Trees via SCFG's

Balanced Trees. Let $\alpha \geq 2$ be an integer. In Chapter 1, we defined the concept of α-balanced tree strictly in terms of the structure of the tree. In the present chapter, we show that this class of trees is the ensemble of trees generated by a particular type of SCFG, such that the SCFG-generated probability distribution on the class of trees is equiprobable (that is, every tree in the class has the same probability). The equiprobable property will imply later on that we can use fixed-length binary codewords when we losslessly compress the trees in the class of α-balanced trees having n leaves.

What should an SCFG be so that the ensemble of trees generated by it is the class of α-balanced trees having n leaves? To answer this question, let us see what a production rule of this SCFG would have to look like. Suppose we have a vertex v of an α-balanced tree having α children, such that there is a total of k leaves descending from v. Let $(k_1, k_2, \cdots, k_\alpha)$ be the vector whose i-th component k_i is the number of leaves descending from the i-th child of v (remember that our trees are always ordered!). Then k_i must either be $\lfloor k/\alpha \rfloor$ or $\lceil k/\alpha \rceil$. The k_i's must also sum to k. The reader can check that the only way these things can happen is if r of k_i's are equal to $\lfloor k/\alpha \rfloor + 1$ and the remaining $\alpha - r$ of the k_i's are equal to $\lfloor k/\alpha \rfloor$, where $r \in \{0, 1, \cdots, \alpha - 1\}$ is the remainder upon division of k by α, that is,

$$k = \alpha \lfloor k/\alpha \rfloor + r.$$

From now on, let us denote the remainder r via the Matlab notation $\mathrm{mod}(k, \alpha)$. To

generate v and its children, the SCFG must thus possess a production rule of the form

$$k \to (k_1, k_2, \cdots, k_\alpha) \tag{2.1}$$

where $\text{mod}(k, \alpha)$ of the right side entries are equal to $\lfloor k/\alpha \rfloor + 1$ and the remaining $\alpha - \text{mod}(k, \alpha)$ entries are equal to $\lfloor k/\alpha \rfloor$. Because the k_i's can be permuted without these properties being affected, and there are $\binom{\alpha}{\text{mod}(k,\alpha)}$ such permutations with $\text{mod}(k, \alpha)$ meaning $\text{mod}(k, \alpha)$, we therefore have this many production rules of the form (2.1), and we require them to be equiprobable. Suppose we now have a nonleaf vertex v of an α-balanced tree which has $k < \alpha$ children. As these children are all leaves, we must have a production rule of the SCFG of form

$$k \to (1, 1, \cdots, 1),$$

where the right side consists of k entries all equal to 1. This rule would have probability 1 assigned to it. We are now ready for a formal definition.

Definition. Let $n \geq \alpha$. The SCFG generating the set of α-balanced trees with n leaves is denoted SCFG(α, n). SCFG(α, n) has start symbol n, set of non-terminal symbols $\{2, 3, \cdots, n\}$, and terminal symbol 1. There are two types of production rules:

- For each k satisfying $\alpha \leq k \leq n$, there are $\binom{\alpha}{\text{mod}(k,\alpha)}$ equally likely production rules of the form

  $$k \to (k_1, k_2, \cdots, k_\alpha)$$

  in which $\text{mod}(k, \alpha)$ of the k_i's are equal to $\lfloor k/\alpha \rfloor + 1$ and the remaining $\alpha - \text{mod}(k, \alpha)$of the k_i's are equal to $\lfloor k/\alpha \rfloor$.

- For each k satisfying $2 \leq k < \alpha$, there is a unique production rule of the form

  $$k \to (1, 1, \cdots, 1),$$

  where the right side has k entries, and this rule is assigned probability 1.

Theorem 2.0.2. *The ensemble of trees generated by the grammar SCFG(α, n) is the set of all α-balanced trees having n leaves, with each tree assigned equal probability.*

Remarks. It is clear that the ensemble of the Theorem is the set of all α-balanced trees having n leaves. What needs to be proved is that these trees have equal probability.

This will follow from the fact that rules having the same left side are assigned equal probability. We also remark that in forming the α-balanced trees with n leaves some of the rules in SCFG(α, n) may not be used. Throwing away these rules, we obtain a smaller grammar which shall be denoted SCFG$^*(\alpha, n)$, which is the smallest SCFG generating the set of α-balanced trees having n leaves.

We present several examples illustrating the concept of α-balanced tree.

Example 2.0.3. *The tree in Figure 1.8 is not a* 4*-balanced tree, since to meet the balance condition the labels on the four depth* 1 *vertices would have to be some permutation of* $5, 6, 6, 6$. *However, all the trees in Figure 2.1 are* 3*-balanced trees. In fact, these constitute the set of all* 3*-balanced trees having* 5 *leaves.*

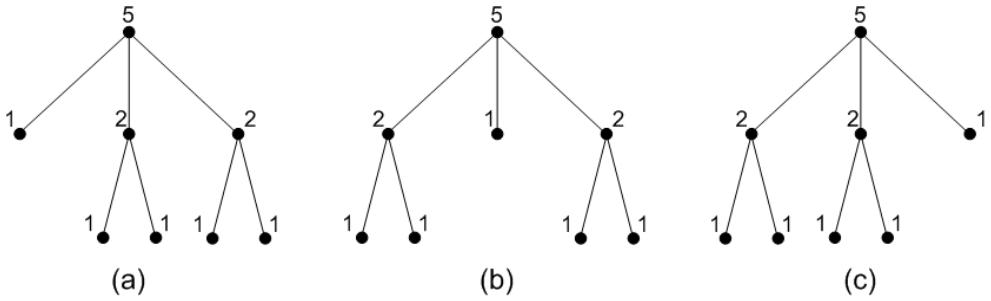


Figure 2.1: 3-balanced trees with 5 leaf vertices

Example 2.0.4. *As has been pointed out in Section 1.5, the set of* 2*-balanced trees coincides with the set of bisection trees. For each* $2 \leq k \leq n$*, the production rules of SCFG*$(2, n)$ *with left side* k *would be*

$$k \xrightarrow{1} (k/2, k/2)$$

if k *is even and*

$$k \xrightarrow{1/2} (\lfloor k/2 \rfloor, \lceil k/2 \rceil)$$
$$k \xrightarrow{1/2} (\lceil k/2 \rceil, \lfloor k/2 \rfloor)$$

if k *is odd, where we have written the probability assigned to a rule as a label over the arrow.*

Example 2.0.5. *Let us look at the grammars generating the* 3*-balanced trees. We will call* 3*-balanced trees* 3*-balanced trees. For general* $n \geq 3$*, the rules in SCFG*$(3, n)$ *with left side* k *(*$3 \leq k \leq n$*) are as follows:*

- *If* k *is divisible by* 3*, we have one rule*

$$k \xrightarrow{1} (k/3, k/3, k/3).$$

- *If* k *is of the form* $3j + 1$*, we have the three rules*

$$k \xrightarrow{1/3} (j + 1, j, j)$$

$$k \xrightarrow{1/3} (j, j + 1, j)$$

$$k \xrightarrow{1/3} (j, j, j + 1).$$

- *If* k *is of the form* $3j + 2$*, we have the three rules*

$$k \xrightarrow{1/3} (j + 1, j + 1, j)$$

$$k \xrightarrow{1/3} (j + 1, j, j + 1)$$

$$k \xrightarrow{1/3} (j, j + 1, j + 1).$$

The only remaining rule is $2 \to (1, 1)$ *(with probability one).*

Example 2.0.6. *Let us look further at the set of all* 3*-balanced trees having* 5 *leaves, which we see depicted in Figure 2.1. We see by looking at these trees that the production rules of the reduced grammar SCFG*$^*(3, 5)$ *are:*

$$5 \xrightarrow{1/3} (1, 2, 2)$$

$$5 \xrightarrow{1/3} (2, 1, 2)$$

$$5 \xrightarrow{1/3} (2, 2, 1)$$

$$2 \xrightarrow{1} (1, 1).$$

Each such tree is built by the application of three production rules in this grammar, one for each of the three nonleaf vertices. One of the three rules has probability $1/3$*,*

and the other two have probability 1. *Thus, each of the three trees has probability* $(1/3)(1)(1) = 1/3$ *of being generated. This illustrates the equiprobable character of the probability distribution on the set of* α*-balanced trees having* n *leaves induced by grammar* $SCFG(\alpha, n)$.

Example 2.0.7. *We now look at the grammars generating the* 4*-balanced trees. For general* $n \geq 4$*, the rules in* $SCFG(4, n)$ *with left side* k $(4 \leq k \leq n)$ *are as follows:*

- *If* k *is divisible by* 4*, we have one rule*

$$k \xrightarrow{1} (k/4, k/4, k/4, k/4).$$

- *If* k *is of the form* $4j + 1$*, we have the four rules*

$$k \xrightarrow{1/4} (j+1, j, j, j)$$
$$k \xrightarrow{1/4} (j, j+1, j, j)$$
$$k \xrightarrow{1/4} (j, j, j+1, j)$$
$$k \xrightarrow{1/4} (j, j, j, j+1).$$

- *If* k *is of the form* $4j + 2$*, we have the* $\binom{4}{2} = 6$ *rules*

$$k \xrightarrow{1/6} (j+1, j+1, j, j)$$
$$k \xrightarrow{1/6} (j+1, j, j+1, j)$$
$$k \xrightarrow{1/6} (j+1, j, j, j+1)$$
$$k \xrightarrow{1/6} (j, j+1, j+1, j)$$
$$k \xrightarrow{1/6} (j, j+1, j, j+1)$$
$$k \xrightarrow{1/6} (j, j, j+1, j+1).$$

- *If* k *is of the form* $4j + 3$*, we have the four rules*

$$k \xrightarrow{1/4} (j+1, j+1, j+1, j)$$
$$k \xrightarrow{1/4} (j+1, j+1, j, j+1)$$
$$k \xrightarrow{1/4} (j+1, j, j+1, j+1)$$
$$k \xrightarrow{1/4} (j, j+1, j+1, j+1).$$

The remaining rules are $3 \to (1,1,1)$ *(with probability one), and* $2 \to (1,1)$ *(with probability one).*

Example 2.0.8. *Let us count how many* 3*-balanced trees there are having* 13 *leaves. Each such tree is built via the application of eight production rules. The first production rule that is applied has left side* 13 *and right side some permutation of* $(4,4,5)$*. This rule has probability* $1/3$*. The next two rules that are applied each have left side* 4 *and right side a permutation of* $(1,1,2)$*; each of these two rules has probability* $1/3$*. The fourth rule that is applied has left side* 5 *and right side a permutation of* $(1,2,2)$*; this rule has probability* $1/3$*. The remaining four rules that are applied are all* $2 \to (1,1)$*, having probability* 1*. Thus, the probability of each* 3*-balanced tree having* 13 *leaves is*

$$(1/3)(1/3)(1/3)(1/3)(1)(1)(1)(1) = 1/81.$$

Since the induced probability distribution on these trees is equiprobable, this means there are 81 3*-balanced trees having* 13 *leaves.*

Example 2.0.9. *In Example 2.0.8, we pointed out that the set* $\mathcal{T}_{13}$ *of* 3*-balanced trees with* 13*leaves is in natural one-to-one correspondenced with the hierarchical type class* $\mathcal{S}_3(BBAAABABA)$*, and we showed in particular how the tree* $T^* \in \mathcal{T}_{13}$ *in Figure 1 of [3] corresponds to string* $BBAAABABA \in S$*. There are a total of* $3^4 = 81$ *trees in* $\mathcal{T}_{13}$*, are therefore* 81 *strings in* S*. We can losslessly encode each of these* 81 *trees into a binary codeword of fixed length* $\lceil \log_2(81) \rceil = 7$ *by instead assigning codewords to the corresponding strings in* S*. We illustrate how this is done for the string* $BBAAABABA$*. Visualize the* $T_2(3)$ *tree in which the* 9 *leaves are labeled left to right with the entries of the sting, namely,*

$$B, B, A, A, A, B, A, B, A.$$

The three left-to-right depth one vertices of the $T_2(3)$ *tree then have ordered sets of children labels which are the respective triples*

$$(B,B,A), (A,A,B), (A,B,A). \tag{2.2}$$

The relations

$$(B,B,A) \in \{(A,B,B) = 0,\ (B,A,B) = 1,\ (B,B,A) = 2\},$$

$$(A,A,B),(A,B,A) \in \{(A,A,B) = 0\,(A,B,A) = 1,\ (B,A,A) = 2\}$$

indicate how each triple in (2.2) is positioned in the set of all possible re-orderings of the three entries of the triple, with each set in lexicographical order, and each member in each set assigned an index $0, 1, 2$ *to indicate its position within the set. The three depth one vertices in the* $T_2(3)$ *tree are then assigned the respective label pairs* $(ABB, 2)$, $(AAB, 0)$, $(AAB, 1)$, *where the reader will note that the first entry of each pair indicates the first triple in a set of permuted triples, and the second entry gives the position number of a particular triple within this set. The three children of the root vertex of* $T_2(3)$ *yield triple of string labels* (ABB, AAB, AAB) *occupying position* 2 *in the set of permuted triples*

$$\{(AAB,AAB,ABB) = 0,\ (AAB,ABB,AAB) = 1,\ (ABB,AAB,AAB) = 2\},$$

and so the root vertex is assigned label $(AABAABABB, 2)$. *In this way, each nonleaf vertex* v *of* $T_2(3)$ *has been assigned a label pair* (i, j), *where*

- i *is a string which generates the set of triples* $S(i_1, i_2, i_3)$ *consisting of all distinct permutations of* (i_1, i_2, i_3), *where* i_1, i_2, i_3 *are the three strings of equal length that* i *is partitioned into left-to-right.*

- j *is position number of a triple in* $S(i_1, i_2, i_3)$ *and the successive entries of this triple will be the string labels on the left-to-right children of* v.

In similar fashion, this vertex labeling of the $T_2(3)$ *tree can be done for each of the* 81 *strings in* S. *In each of these cases, let the vertex labels on the nonleaf vertices be ordered as*

$$(i_1, j_1), (i_2, j_2), (i_3, j_3), (i_4, j_4),$$

with the (i, j) *label on the root vertex listed first, followed by the* (i, j) *labels on the three depth one vertices ordered left to right. For each of the strings in* S, *the* i_1 *label will always be* $AABAABABB$, *the first string in* S *in lexicographical order, and the decoder will know this string because the encoder and decoder both know* S. *The decoder will therefore be able to reconstruct every label on* $T_2(3)$ *from the* 4*-tuple of indices* (j_1, j_2, j_3, j_4). *(The decoder puts label* $(AABAABABB, j_1)$ *on the root, determines the labels on its three children (the depth one vertices) from the index* j_1, *and then*

determines the labels on the children of the depth one vertices (the leaves) from j_2, j_3, j_4; the leaf labels yield the string in S that was encoded.) There are $3^4 = 81$ possible 4-tuples (j_1, j_2, j_3, j_4), since each entry j_i belongs to the set $\{0, 1, 2, 3\}$ of size 4. The encoder can easily provide a fixed-length binary codeword for each of these 4-tuples. One way to do this is to regard (j_1, j_2, j_3, j_4) as the base three expansion of an integer belonging to the set $\{01, 1, \cdots, 80\}$; the codeword is then the 7-bit binary expansion of this integer. For example, for the string $BBAAABABA$ processed in the preceding, the decoder will receive $j_2, j_3, j_4 = (2, 2, 0, 1)$. The integer for which this is the base three expansion is $2 * 27 + 2 * 9 + 0 * 3 + 1 = 73$. The codeword is then the binary expansion of 73, which is 1001001.

Discussion. Let $H_\alpha(n)$ be the logarithm to base two of the number of α-balanced trees having n leaves. If we encode each such tree into a unique fixed-length binary codeword of minimal length, then this length will be $\lceil H_\alpha(n) \rceil$. The quantity $H_\alpha(n)/n$ thus represents the approximate compression rate in code bits per leaf of this compression scheme if the number of leaves n is large. Later on in this thesis, we want to examine what happens to $H_\alpha(n)/n$ as n becomes large. This problem will be solved by finding a periodic function P_α on the real line, of period one, such that

$$H_\alpha(n) = nP_\alpha(\log_\alpha n), \quad n \geq \alpha.$$

We have already seen such a result in the 2-balanced tree case (bisection trees). As a consequence of this thesis, we will have such a result for general $\alpha \geq 2$; this is the main contribution of our work.

Our machinery for finding the P_α function involves certain concepts that will be covered next, the first of which is the concept of an iterated function system.

2.1 Iterated Function Systems

Definition of Contraction Mapping. Let X be a complete metric space. A contraction mapping on X is a function $F : X \to X$ for which there is a non-negative real number $c < 1$ such that

$$d\,(F(x), F(y)) \leq c\, d(x, y), \quad x, y \in X, \tag{2.3}$$

where d is the metric on X. According to the Banach fixed point theorem, there exists a unique fixed point of a contraction mapping F on X, that is, a unique point $x^* \in X$ satisfying $F(x^*) = x^*$. Moreover, x^* can be obtained as follows. Let x_0 be any initial point in X. Define sequence $\{x_n : n \geq 1\}$ recursively by iteratively applying the function F:

$$x_n = F(x_{n-1}),\ n \geq 1. \tag{2.4}$$

Then, the sequence $\{x_n\}$ converges to x^*.

Definition of Iterated Function System. Let D be a non-empty closed subset of a finite-dimensional Euclidean space, which we regard as a metric space equipped with the Euclidean metric. An iterated function system (IFS) on D is a finite non-empty set of contraction mappings on D. Given an IFS $\{T_0, T_1, \cdots, T_{n-1}\}$ on D, there exists (Theorem 9.1, [10]) a unique non-empty compact subset S of D such that

$$S = \bigcup_{i=0}^{n-1} T_i(S).$$

The set S is called the attractor of the IFS $\{T_0, T_1, \cdots, T_{n-1}\}$. We remark that one can obtain S as the fixed point of a contraction mapping F on a metric space X consisting of the non-empty compact subsets of D, with S computed as the limit of a sequence of sets generated by recursion under iteration by F, as in (2.4); for more details, see [10]. We will call an IFS on D an affine IFS if D is a convex set and if the mappings in the IFS are affine mappings on D.

In the set of mappings $\{T_0, T_1, \cdots, T_{n-1}\}$ of the preceding paragraph, suppose we drop the assumption that the mappings T_i are contraction mappings, but we still obtain an attractor S defined uniquely by $S = \bigcup_i T_i(S)$. Then, we will call $\{T_0, T_1, \cdots, T_{n-1}\}$ a *generalized IFS*. The concept of generalized IFS will be useful to us because sometimes we will have an attractor of a set of mappings without being able to show that the mappings are contractions.

Theorem 2.1.1. *Let T_0, T_1 be the mappings on $\mathbb{R}^2$ given by*

$$\begin{aligned} T_0(x,y) &= (x, x+y)/2 \\ T_1(x,y) &= (x+1, 1-x+y)/2. \end{aligned} \tag{2.5}$$

Then $\{T_0, T_1\}$ is an affine IFS, and there is a unique continuous function $h_2 : [0,1] \to [0,1]$ such that the graph of h_2 is the attractor of this IFS. Furthermore, letting P_2 be the unique continuous periodic function with period one on $\mathbb{R}$ such that

$$h_2(x) = (x+1)P_2(\log_2(x+1)), \quad 0 \le x \le 1, \tag{2.6}$$

we have

$$H_2(n) = nP_2(\log_2 n), \quad n \ge 2, \tag{2.7}$$

where $H_2(n)$ is the logarithm to base two of the number of bisection trees having n leaves.

Plotting P_2. Earlier in the thesis, we gave a plot of the function P_2 in Figure 1.6. We now explain how we did this, using the results of the preceding theorem. Let S_0 be the single-point set $S_0 = \{(0,0)\}$. Then, for the affine IFS defined in (2.5), recursively define finite sets of points $S_1, S_2, \cdots$ as follows:

$$S_n = T_0(S_{n-1}) \cup T_1(S_{n-1}), \quad n \ge 1.$$

By the preceding theorem, set S_n consists of 2^n points on the graph of h_2. Then, using the change of variable (2.6), we converted the points in S_n to 2^n points on the graph of P_2.

Another way to obtain the plot of P_2 is to obtain the plot directly as the attractor of a generalized IFS. Let C_2 be the graph of P_2. One can apply the change of variable of Theorem 2.1.1 to the IFS $\{T_0, T_1\}$ to obtain the generalized IFS $\{T_0^*, T_1^*\}$ on C_2 in which

$$\begin{aligned} T_0^*(t,y) &= \left(\log_2\left[\tfrac{2^t+1}{2}\right], \tfrac{2^t y+(2^t-1)}{2^t+1}\right), \quad (t,y) \in C_2 \\ T_1^*(t,y) &= \left(\log_2\left[\tfrac{2^t+2}{2}\right], \tfrac{2^t y+(2-2^t)}{2^t+2}\right), \quad (t,y) \in C_2. \end{aligned} \tag{2.8}$$

Starting with the set $S_0^* = \{(0,0)\}$, one then obtains for each $n \ge 1$ a set S_n^* of points in C_2 by the recursion

$$S_n^* = T_0^*(S_{n-1}) \cup T_1^*(S_{n-1}), \ n \ge 1.$$

For n at least 15, plotting the points in S_n^* gives an accurate plot of P_2.

Preview of Coming Attractions. One of the principal focuses of this thesis is to obtain a result like the preceding theorem for computing the entropies of α-balanced trees for any $\alpha \ge 3$. For example, for $\alpha = 3$, we will exhibit an IFS $\{T_0, T_1, T_2\}$ on $\mathbb{R}^2$

whose attractor is the graph of a function $h_3 : [0,1] \to \mathbb{R}$ which converts via change of variable into a continuous function P_3 with period 1 for which

$$H_3(n) = nP_3(\log_2 n), \quad n \geq 3,$$

where $H_3(n)$ is the entropy of the ensemble of 3-balanced trees having n leaves. More generally, for general $\alpha \geq 3$, the IFS we will derive will consist of α contractions on $\mathbb{R}$, and we will be able to use this IFS to characterize the asymptotic behavior of α-balanced tree entropy as the number of leaves grows without bound.

Chapter 3

Compression Rate Curves Associated with Balanced Trees

This chapter describes the generalization of information theory of 2-balanced tree structures to the case for α-balanced trees, where $\alpha \geq 2$ is fixed throughout this chapter.

3.1 Entropy and Compression Rate Curve of α-balanced Trees

Entropy of an α-balanced Tree.

Theorem 3.1.1. *The entropy sequence $\{H_\alpha(n)\}_{n=1}^{\infty}$ for a general α-balanced tree is generated by the following recursion,*

$$H_\alpha(n) = \begin{cases} 0, & 1 \leq n \leq \alpha - 1 \\ \log_2 \binom{\alpha}{\mathrm{mod}(n,\alpha)} + \mathrm{mod}(n,\alpha) \cdot H_\alpha\left(\lceil \frac{n}{\alpha} \rceil\right) + (\alpha - \mathrm{mod}(n,\alpha)) \cdot H_\alpha\left(\lfloor \frac{n}{\alpha} \rfloor\right), & n \geq \alpha. \end{cases} \tag{3.1}$$

Example 3.1.2. *Let $\alpha = 3$. We have $H_3(1) = H_3(2) = 0$. For $n \geq 3$, the recursion for $H_3(n)$ is as follows.*

$$H_3(n) = \begin{cases} 3H_3(n/3), & n \equiv 0 \bmod 3 \\ \log_2 3 + 2H_3(\lfloor n/3 \rfloor) + H_3(\lceil n/3 \rceil), & n \equiv 1 \bmod 3 \\ \log_2 3 + H_3(\lfloor n/3 \rfloor) + 2H_3(\lceil n/3 \rceil), & n \equiv 2 \bmod 3. \end{cases}$$

Example 3.1.3. *Let $\alpha = 4$. We have $H_4(1) = H_4(2) = H_4(3) = 0$. For $n \geq 4$:*

$$H_4(n) = \begin{cases} 4H_4(n/4), & n \equiv 0 \bmod 4 \\ 2+3H_4(\lfloor n/4 \rfloor)+H_4(\lceil n/4 \rceil), & n \equiv 1 \bmod 4 \\ \log_2 6+2H_4(\lfloor n/4 \rfloor)+2H_4(\lceil n/4 \rceil), & n \equiv 2 \bmod 4 \\ 2+H_4(\lfloor n/4 \rfloor)+3H_4(\lceil n/4 \rceil), & n \equiv 3 \bmod 4. \end{cases}$$

Example 3.1.4. *Let $\alpha = 5$. We have $H_5(i) = 0$, $1 \leq i \leq 4$. For $n \geq 5$:*

$$H_5(n) = \begin{cases} 5H_5(n/5), & n \equiv 0 \bmod 5 \\ \log_2 5+4H_5(\lfloor n/5 \rfloor)+H_5(\lceil n/5 \rceil), & n \equiv 1 \bmod 5 \\ \log_2 10+3H_5(\lfloor n/5 \rfloor)+2H_5(\lceil n/5 \rceil), & n \equiv 2 \bmod 5 \\ \log_2 10+2H_5(\lfloor n/5 \rfloor)+3H_5(\lceil n/5 \rceil), & n \equiv 3 \bmod 5 \\ \log_2 5+H_5(\lfloor n/5 \rfloor)+4H_5(\lceil n/5 \rceil), & n \equiv 4 \bmod 5. \end{cases}$$

Theorem 3.1.5. *Fix $\alpha \geq 2$. There is a unique periodic continuous function $P_\alpha : \mathbb{R} \to \mathbb{R}$, with period 1, such that*

$$\frac{H_\alpha(n)}{n} = P_\alpha\left(\log_\alpha n\right), \qquad n \geq 1. \tag{3.2}$$

Compression Rate Curve C_α. In the special case $\alpha = 2$, we earlier discussed the compression rate curve C_2 (which is the graph of P_2) and discussed its significance with regard to the compression of bisection trees. Similarly, for each $\alpha > 2$, the compression rate curve C_α, the graph of P_α, has significance with regard to the compression of α-balanced trees. The compression rate curve C_α is defined as

$$C_\alpha = \{(t, y) : 0 \leq t \leq 1,\ y = P_\alpha(t)\}. \tag{3.3}$$

A point on $(t, P_\alpha(t))$ on the curve C_α has the following data compression interpretation. One can find infinitely many n such that the fractional part of $\log_\alpha n$ is approximately t and $H_\alpha(n)/n$ is approximately $P_\alpha(t)$. For each such n, the length of the binary codeword in compressing the α-balanced trees with n leaves will be roughly $P_\alpha(t)n$ (in other words, compression rate in code bits per leaf is roughly $P_\alpha(t)$).

The compression rate curve C_α consists of $\alpha - 1$ pieces of continuous and non-overlapping curves, $C_\alpha(j)$.

$$C_\alpha = \bigcup_{j=1}^{\alpha-1} C_\alpha(j). \tag{3.4}$$

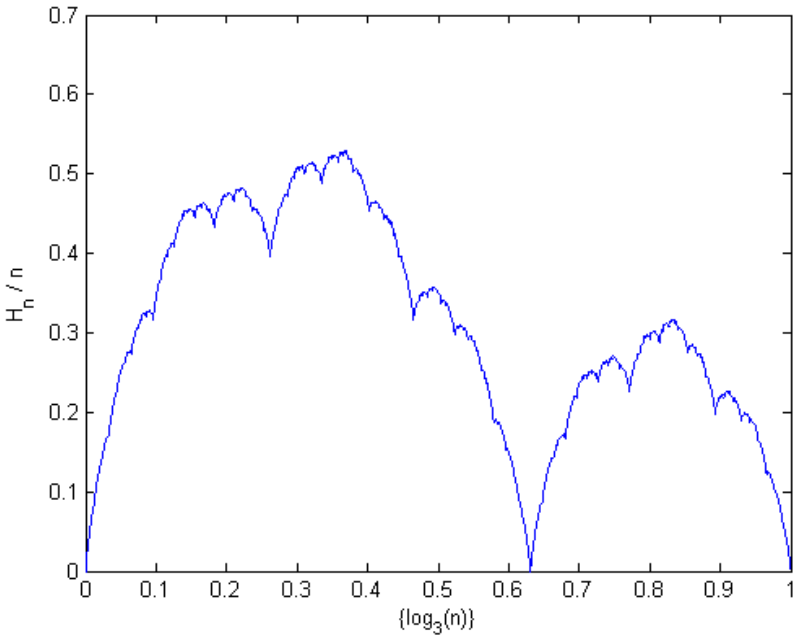


Figure 3.1: Compression rate curve C_3 for a 3-balanced tree

The j-th piece of the C_α is defined in the interval between the logarithms to base α of two consecutive integers as

$$C_\alpha(j) = \{(t, P_\alpha(t)) \in C_\alpha : \log_\alpha j \leq t \leq \log_\alpha(j+1)\}, \;\; j = 1, 2, \cdots, \alpha - 1. \tag{3.5}$$

Thus, there are $(\alpha - 2)$ points with zero ordinate $\{(t, y) : 0 < t < 1, y = 0\}$ in C_α and two consecutive pieces meet only at the point with zero ordinate.

$$C_\alpha(j) \cap C_\alpha(j+1) = \{(\log_\alpha(j+1, 0)\}, \;\; j = 1, 2, \cdots, \alpha - 2.$$

For example, the reader can see in Figure 3.1 that the curve C_3 consists of two pieces meeting at the point $(0, \log_2 3)$.

Generalized IFS Whose Attractor is C_α. Theorem 3.1.6 which follows is the main result of this thesis, and will be proved in the appendix. For each $\alpha \geq 2$, it lays out a specific generalized IFS whose attractor is C_α.

Theorem 3.1.6. *Fix $\alpha \geq 2$. For each $i = 0, 1, \cdots, \alpha - 1$, let T_i^* be the mapping of $[0,1] \times \mathbb{R}$ into itself defined piecewise for $j = 1, 2, \cdots, \alpha - 1$ as*

$$T_i^*(t,y) = \left(\log_\alpha \left[\frac{\alpha^t+(\alpha-1)(i+1)}{\alpha}\right], \frac{\alpha^t y+(\alpha^t-j)\log_2 z_{i-j}+(j+1-\alpha^t)\log_2 z_{i-j+1}}{\alpha^t+(\alpha-1)(i+1)}\right), \quad \log_\alpha j \leq t \leq \log_\alpha(j+1),\ y \in \mathbb{R}, \tag{3.6}$$

where $z_n \triangleq \binom{\alpha}{\mathrm{mod}(n,\alpha)}$. Then each T_i^* maps C_α into itself and $\{T_0^*, T_1^*, \cdots, T_{\alpha-1}^*,\}$ is a generalized IFS whose attractor is C_α.

Discussion.

- Change of variable $\alpha^t = t + 1$ allows us to re-express the right side of (3.6) for each $j = 1, 2, \cdots, \alpha - 1$ as

$$\left(\log_\alpha \left[\frac{x+(\alpha-1)(i+1)+1}{\alpha}\right], \frac{(x+1)y+(1-j+x)\log_2 z_{i-j}+(j-x)\log_2 z_{i-j+1}}{x+(\alpha-1)(i+1)+1}\right), \quad j-1 \leq x \leq j;\ y \in \mathbb{R}. \tag{3.7}$$

  This simpler expression will aid us in the later proof of Theorem 3.1.6.

- As a consequence of our method of proof of Theorem 3.1.6, we will also be able to conclude the following interesting property. For each T_i^* in the generalized IFS of Theorem 3.1.6, and each $j \in \{0, 1, \cdots, \alpha - 1\}$, there exists $j' \in \{0, 1, \cdots, \alpha - 1\}$ such that

$$T_i^*\left(C_\alpha(j)\right) \subset C_\alpha(j').$$

  This property will imply that T_i^* maps C_α into itself.

- In expression (3.7), we see the following relationships between the two coordinates. The numerator of the logarithm of the first coordinate coincides with the denominator of the second coordinate. Also, as i increases by one, this common part increases by $\alpha - 1$. For example, for $\alpha = 3$, this common part increments as $x + 3 \to x + 5$ and $x + 5 \to x + 7$.

- In (3.7), let x_l denote the fractional part of x and let $x_u = 1 - x_l$. Then $x_l = 1 - j + x$ and $x_u = j - x$. Then the numerator of the second coordinate of (3.7) is $(x + 1)y$ plus a linear combination of x_l and x_u. For example if $\alpha = 3$, j is either 1 or 2. When $j = 1$, the $x \in [0, 1]$ and so $x_l = x$ and $x_u = 1 - x$. If $j = 2$, then $x \in [1, 2]$ and so $x_l = x - 1$ and $x_u = 2 - x$.

3.1.1 3-Balanced Trees

Let $\alpha = 3$. Then Theorem 3.6 gives a generalized IFS $\{T_0^*, T_1^*, T_2^*\}$ of mappings from $[0,1] \times \mathbb{R}$ into itself, such that the attractor of this IFS is compression rate curve C_3. According to compression rate curve properties discussed earlier, C_3 consists of two pieces which overlap each other only at the point $(\log_3 2, 0)$ on C_3.

$$\begin{array}{l} C_3 = C_3(1) \cup C_3(2) \\ : \left\{ \begin{array}{lcl} C_3(1) & = & \{(t, P_3(t)) : 0 \le t \le \log_3 2\} \\ C_3(2) & = & \{(t, P_3(t)) : \log_3 2 \le t \le 1\}. \end{array} \right. \end{array} \tag{3.8}$$

For a point (t, y) on C_3, the generalized IFS $\{T_0^*, T_1^*, T_2^*\}$ is defined depending on which piece of curve ($C_3(1)$ or $C_3(2)$) the point belongs to.

$$\left.\begin{array}{lcl} T_0^*(t,y) & = & \left(\log_3 \frac{3^t+2}{3}, \frac{3^t y + (3^t-1)\log_2 3}{3^t+2}\right) \\ T_1^*(t,y) & = & \left(\log_3 \frac{3^t+4}{3}, \frac{3^t y + (2-3^t)\log_2 3}{3^t+4}\right) \\ T_2^*(t,y) & = & \left(\log_3 \frac{3^t+6}{3}, \frac{3^t y + \log_2 3}{3^t+6}\right) \end{array}\right\} \ (t,y) \in C_3(1),$$

$$\left.\begin{array}{lcl} T_0^*(t,y) & = & \left(\log_3 \frac{3^t+2}{3}, \frac{3^t y + \log_2 3}{3^t+2}\right) \\ T_1^*(t,y) & = & \left(\log_3 \frac{3^t+4}{3}, \frac{3^t y + (3^t-1)\log_2 3}{3^t+4}\right) \\ T_2^*(t,y) & = & \left(\log_3 \frac{3^t+6}{3}, \frac{3^t y + (2-3^t)\log_2 3}{3^t+6}\right) \end{array}\right\} \ (t,y) \in C_3(2). \tag{3.9}$$

The mappings (3.9) can be simplified by substituting $x + 1$ for 3^t, as follows. (This change of variable will be exploited at the end of this section in our proof of Theorem 3.1.6.)

$$\left.\begin{array}{lcl} U_0^*(x,y) & = & \left(\log_3 \frac{x+3}{3}, \frac{(x+1)y + x\log_2 3}{x+3}\right) \\ U_1^*(x,y) & = & \left(\log_3 \frac{x+5}{3}, \frac{(x+1)y + (1-x)\log_2 3}{x+5}\right) \\ U_2^*(x,y) & = & \left(\log_3 \frac{x+7}{3}, \frac{(x+1)y + \log_2 3}{x+7}\right) \end{array}\right\} \ (x,y) \in C_3(1),$$

$$\left.\begin{array}{lcl} U_0^*(x,y) & = & \left(\log_3 \frac{x+3}{3}, \frac{(x+1)y + \log_2 3}{x+3}\right) \\ U_1^*(x,y) & = & \left(\log_3 \frac{x+5}{3}, \frac{(x+1)y + x\log_2 3}{x+5}\right) \\ U_2^*(x,y) & = & \left(\log_3 \frac{x+7}{3}, \frac{(x+1)y + (1-x)\log_2 3}{x+7}\right) \end{array}\right\} \ (x,y) \in C_3(2).$$

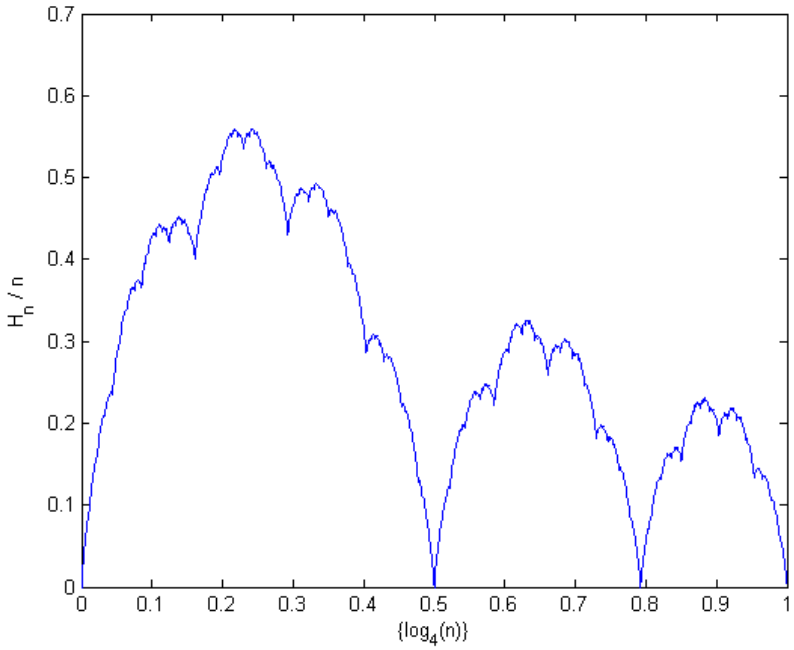


Figure 3.2: Compression rate curve C_4 for a 4-balanced tree

3.1.2 Compression Rate Curve C_4 for 4-balanced Trees

When $\alpha = 4$, the compression rate curve C_4 consists of 3 pieces $\{C_4(1), C_4(2), C_4(3)\}$ and two consecutive pieces join at $(\log_4 2, 0)$ and $(\log_4 3, 0)$ on C_4.

$$\begin{aligned} & C_4 = C_4(1) \cup C_4(2) \cup C_4(3) \\ & : \begin{cases} C_4(1) &= \{(t, P_4(t)) : 0 \le t \le \log_4 2\} \\ C_4(2) &= \{(t, P_4(t)) : \log_4 2 \le t \le \log_4 3\} \\ C_4(3) &= \{(t, P_4(t)) : \log_4 3 \le t \le 1\}. \end{cases} \end{aligned} \tag{3.10}$$

The IFS of C_4 is defined separately on each of the $\{C_4(1), C_4(2), C_4(3)\}$.

$$\left.\begin{aligned} U_0^*(x,y) &= \left(\log_4 \tfrac{x+4}{4}, \tfrac{(x+1)y + x\log_2 4}{x+4}\right) \\ U_1^*(x,y) &= \left(\log_4 \tfrac{x+7}{4}, \tfrac{(x+1)y + (1-x)\log_2 4}{x+7}\right) \\ U_2^*(x,y) &= \left(\log_4 \tfrac{x+10}{4}, \tfrac{(x+1)y + x\log_2 4 + (1-x)\log_2 6}{x+10}\right) \\ U_3^*(x,y) &= \left(\log_4 \tfrac{x+13}{4}, \tfrac{(x+1)y + (1-x)\log_2 4 + x\log_2 6}{x+13}\right) \end{aligned}\right\} (x,y) \in C_4(1), \tag{3.11}$$

$$\left.\begin{array}{rcl} U_0^*(x,y) & = & \left(\log_4 \frac{x+4}{4}, \frac{(x+1)y+(2-x)\log_2 4+(x-1)\log_2 6}{x+4}\right) \\ U_1^*(x,y) & = & \left(\log_4 \frac{x+7}{4}, \frac{(x+1)y+(x-1)\log_2 4}{x+7}\right) \\ U_2^*(x,y) & = & \left(\log_4 \frac{x+10}{4}, \frac{(x+1)y+(2-x)\log_2 4}{x+10}\right) \\ U_3^*(x,y) & = & \left(\log_4 \frac{x+13}{4}, \frac{(x+1)y+(x-1)\log_2 4+(2-x)\log_2 6}{x+13}\right). \end{array}\right\} (x,y) \in C_4(2), \quad (3.12)$$

$$\left.\begin{array}{rcl} U_0^*(x,y) & = & \left(\log_4 \frac{x+4}{4}, \frac{(x+1)y+(x-2)\log_2 4+(3-x)\log_2 6}{x+4}\right) \\ U_1^*(x,y) & = & \left(\log_4 \frac{x+7}{4}, \frac{(x+1)y+(3-x)\log_2 4+(x-2)\log_2 6}{x+7}\right) \\ U_2^*(x,y) & = & \left(\log_4 \frac{x+10}{4}, \frac{(x+1)y+(x-2)\log_2 4}{x+10}\right) \\ U_3^*(x,y) & = & \left(\log_4 \frac{x+13}{4}, \frac{(x+1)y+(3-x)\log_2 4}{x+13}\right). \end{array}\right\} (x,y) \in C_4(3), \quad (3.13)$$

Now the numerators of the second coordinate have the terms the linear combination of a logarithm and a fractional part of x, which is either x_l or x_u.

3.2 Change of Variable Approach

Theorem 3.1.6 is proved via a change of variable approach. The change of variable that shall be employed allows one to convert results of Kieffer [1] on hierarchical types classes to results concerning ensembles of balanced trees. In this section, we explain how this change of variable approach works.

Definition. In Examples 1.6.1-1.6.2, we indicated how an ensemble of balanced trees is in one-to-one correspondence with a hierarchical type class. We now need to formally define how one obtains this correspondence. Let $\alpha \geq 2$, and let $n \geq \alpha$ be a positive integer, and let $T_n(\alpha)$ be the ensemble of all -balanced trees having n leaves. Label each vertex of each tree in $T_n(\alpha)$ with the number of leaves descending from that vertex. Let α^k be the largest power of such that $\alpha^k \leq n$. Let $j \in \{1, 2, \cdots, \alpha - 1\}$ be the unique integer such that $j\alpha^k \leq n < (j+1)\alpha^k$. (The pair (k, j) has the following interpretation: The expansion of integer n to base α consists of $k+1$ entries, and the first of these entries is j.) For any tree in $T_n(\alpha)$, if we throw away all edges which descend below the depth k vertices, then we will obtain the tree $T_n(\alpha)$ having α^k leaves, and each of these leaves carries a label from the set $\{j, j+1\}$. Let $\{A, B\}$ be the binary abstract alphabet consisting of the two letters A,B. Consider the following mapping ϕ from $T_n(\alpha)$ into

the set of all strings over $\{A, B\}$ of length α^k. If $T \in \mathcal{T}_n(\alpha)$, let

$$L_1, L_2, \cdots, L_{\alpha^k}$$

be the left-to-right labels from the set $\{j, j+1\}$ on the leaves of the $T_n(\alpha)$ tree assigned to T as in Examples 1.6.1, 1.6.2. Mapping each $L_i = j$ into A and each $L_i = j+1$ into B, the resulting sequence is $\phi(T)$. The mapping ϕ is one-to-one, and $\mathcal{S}_n(\alpha) = \phi(\mathcal{T}_n(\alpha))$ is a hierarchical type class. In this way, we have defined a one-to-one correspondence between the ensemble $T_n(\alpha)$ and the hierarchical type class $\mathcal{S}_n(\alpha)$.

Example 3.2.1. *Consider the ensemble $\mathcal{T}_7(2)$ consisting of the bisection trees having 7 leaves. 7 in binary is 111. The (k, j) pair is then $(2, 1)$. ($k = 2$, one less than the number of entries in 111 and $j = 1$, the leading entry of 111.) By pruning edges below vertices of depth $k = 2$, one obtains a correspondence between the trees in $\mathcal{T}_7(2)$ and labelings of the tree $\mathcal{T}_2(2)$, in which the labels on the $2^2 = 4$ leaves belong to the set $\{j, j+1\} = \{1, 2\}$. It can be checked that one of these edge labelings is 1, 2, 2, 2. The correspondences $1 \leftrightarrow A$, $2 \leftrightarrow B$ then tell us that $\mathcal{T}_7(2)$ is in one-to-one correspondence with the hierarchical type class $\mathcal{S}_2(ABBB)$. This is the same result we obtained in Example 1.6.1 with more effort.*

Example 3.2.2. *Consider the ensemble $\mathcal{T}_{14}(3)$ consisting of the 3-balanced trees having 14 leaves. 14 to base 3 is 112. The (k, j) pair is then $(2, 1)$. ($k = 2$, one less than the number of entries in 112 and $j = 1$, the leading entry of 112.) By pruning edges below vertices of depth $k = 2$, one obtains a correspondence between the trees in $\mathcal{T}_{14}(3)$ and labelings of the tree $\mathcal{T}_2(3)$, in which the labels on the $3^2 = 9$ leaves belong to the set $\{j, j+1\} = \{1, 2\}$. It can be checked that one of these edge labelings is 1, 1, 2, 2, 2, 1, 2, 1, 2. The correspondences $1 \leftrightarrow A$, $2 \leftrightarrow B$ then tell us that $\mathcal{T}_{14}(3)$ is in one-to-one correspondence with the hierarchical type class $\mathcal{S}_3(AABBBABAB)$. This is the same result we obtained in Example 1.6.2.*

Lemma 3.2.3. *Let $\alpha \geq 2$ be fixed. Let $n \geq \alpha$, and let (k, j) be the pair defined earlier for which $j\alpha^k \leq n < (j+1)\alpha^k$. Then each string in hierarchical type class $\mathcal{S}_n(\alpha)$ consists of $n - j\alpha^k$ A's and $(j+1)\alpha^k + n$ B's.*

For the following result, see [1].

Theorem 3.2.4. *Let $\alpha \geq 2$. There is a unique continuous function $h_\alpha : [0,1] \to \mathbb{R}$ such that for each $n \geq \alpha$,*

$$h_\alpha(x) = \log_2 \left(\text{card}\left(\mathcal{S}_n(\alpha)\right)\right)/\alpha^k,$$

where α^k is the largest power of α which is $\leq n$, and $\alpha^k x$ is the number of entries of each string $\mathcal{S}_n(\alpha)$ which are equal to A. Furthermore, the graph $\{(x, h_\alpha(x)) : 0 \leq x \leq 1\}$ of h_α is the attractor of the IFS $\{T_0, T_1, \cdots, T_{\alpha-1}\}$ in which each T_i is the contraction mapping on $[0,1] \times \mathbb{R}$ defined by

$$T_i(x,y) \triangleq \alpha^{-1}\left(x + i,\ x\log_2 \tbinom{\alpha}{i+1} + (1-x)\log_2 \tbinom{\alpha}{i} + y\right), \quad (x,y) \in [0,1] \times \mathbb{R}.$$

Corollary 3.2.5. *Fix $\alpha \geq 2$. Let $P_\alpha : \mathbb{R} \to \mathbb{R}$ be the unique continuous periodic function, with period 1, such that*

$$h_\alpha(x) = (x+j)P_\alpha\left(\log_\alpha(x+j)\right), \quad x \in [0,1],\ j = 1, 2, \cdots, \alpha - 1. \tag{3.14}$$

Then

$$H_\alpha(n)/n = P_\alpha(\log_\alpha n),\ n \geq 1. \tag{3.15}$$

Remark. In the Appendix, we give an alternate proof of the change of variable formula (3.14).

3.3 Generating Compression Rate Curve C_α

This section presents two methods generating the plot of compression rate curve C_α. The first one exploits the result yielding the IFS to get the graph of a self-affine hierarchical function. And then by a change of variable, the compression rate curve for an α-balanced tree C_α is obtained from the graph. The second method generating C_α directly apply the IFS of C_α to an initial set of points on the compression rate curve.

We have the sequence $\{H_\alpha(n) : n \geq 1\}$ such that for $l = 0, 1, \cdots, \alpha - 1$,

$$H_\alpha(1) = H_\alpha(2) = \cdots = H_\alpha(\alpha - 1) = 0,$$

$$H_\alpha(\alpha k + l) = (\alpha - l)H_\alpha(k) + lH_\alpha(k+1) + \log_2 \binom{\alpha}{l},\ \ k \geq 1.$$

The compression rate curve P_α is a periodic function on the real line, with period 1, such that

$$H_\alpha(n)/n = P_\alpha(\log_\alpha n),\ \ n \geq 1. \tag{3.16}$$

3.3.1 Method 1: Via Change of Variable

In [1], a function $H(n_1, n_2)$ was introduced, defined for each pair of non-negative integers (n_1, n_2) such that $n_1 + n_2$ is of the form α^j for some $j \geq 0$. This function is the unique function satisfying:

$$H(0,1) = H(1,0) = 0,$$

$$H(\alpha k_1, \alpha k_2) = \alpha H(k_1, k_2),$$

$$H(\alpha k_1 + l, \alpha k_2 + \alpha - l) = lH(k_1 + 1, k_2) + (\alpha - l)H(k_1, k_2 + 1) + \log_2 \binom{\alpha}{l}, \ {\scriptstyle l=1,2,\cdots,\alpha-1.}$$

It was shown in [1] that there exists a unique continuous function h_α defined on $[0,1]$ such that

$$h_\alpha(i/\alpha^j) = H(i, \alpha^j - i)/\alpha^j, \ \ i = 0, 1, \cdots, \alpha^j; \ j = 0, 1, 2, \cdots. \tag{3.17}$$

Lemma 3.3.1. *Let P be the unique continuous periodic function on the real line, with period 1, such that for $j = 1, 2, \cdots, \alpha - 1$,*

$$h_\alpha(x) = (x + j)P(\log_\alpha(x + j)), \ \ 0 \leq x \leq 1.$$

Then,

$$H_\alpha(n)/n = P(\log_\alpha n), \ \ n \geq 1, \tag{3.18}$$

and therefore we may take P as the function P_α in (3.16).

Change of Variable Method. From the Lemma, the following change of variable method for obtaining $P_\alpha(t)$ from $h_\alpha(t)$ is now clear: for $j = 1, 2, \cdots, \alpha - 1$,

$$P_\alpha(t) = \alpha^{-t} h_\alpha(\alpha^t - j), \quad \log_\alpha j \leq t \leq \log_\alpha(j + 1). \tag{3.19}$$

3.3.2 Method 2: Via IFS Iteration

Let T_i be the affine contraction mapping from $\mathbb{R}^2$ to itself which is defined as

$$T_i(x, y) = \left(\frac{x + i}{k}, \frac{y + (1 - x)\log_2 \binom{k}{i} + x \log_2 \binom{k}{i+1}}{k} \right), \ i = 0, 1, \cdots, \alpha - 1.$$

In the paper [1], it is shown that the α mappings above are contraction mappings. It is also shown that the graph of the function $h_\alpha(x)$ is the attractor of the iterated function system $\{T_0, T_1, \cdots, T_\alpha\}$. Using the change of variable presented in the previous section, one transforms the mappings $T_0, T_1, \cdots, T_{\alpha-1}$ into mappings $T_0^*, T_1^*, \cdots, T_{\alpha-1}^*$ on $[0,1] \times \mathbb{R}$ such that the graph of P_α is the attractor of the IFS $\{T_0^*, T_1^*, \cdots, T_{\alpha-1}^*\}$. The transformed mappings are given in (3.6) and the attractor of the mappings is C_α which is in (3.4) with (3.5).

Remarks. We point out that the mappings T_i^* are the same mappings we reported in [3] without proof. We do not know whether the T_i^* mappings are contractions. We will nevertheless call $\{T_0^*, T_1^*, \cdots, T_{\alpha-1}^*\}$ an iterated function system (IFS) because it has an attractor. (It has an attractor because the IFS $\{T_0, T_1, \cdots, T_{\alpha-1}\}$ has an attractor.)

Method 2: Finding $P_\alpha(t)$ by IFS Iteration. Start with the set $S_0 = \{(0,0)\}$. Form sets $S_1, S_2, \cdots$ as follows by iteration:

$$S_n = T_0^*(S_{n-1}) \cup T_1^*(S_{n-1}) \cup \cdots \cup T_{\alpha-1}^*(S_{n-1}), \quad n \geq 1.$$

For each n, S_n consists of α^n points on

$$\{(t, P_\alpha(t)) : 0 \leq t \leq 1\}.$$

3.3.3 Case Studies

1) Generating Compression Rate Curve C_3.
We have the sequence $\{H_3(n) : n \geq 1\}$ such that

$$H_3(1) = H_3(2) = 0,$$

$$H_3(3k) = 3H_3(k), \quad k \geq 1,$$

$$H_3(3k+1) = 2H_3(k) + H_3(k+1) + \log_2 3, \quad k \geq 1,$$

$$H_3(3k+2) = 2H_3(k+1) + H_3(k) + \log_2 3, \quad k \geq 1.$$

The compression rate curve P_3 is a periodic function on the real line, with period 1, such that

$$H_3(n)/n = P_3(\log_3 n), \quad n \geq 1. \tag{3.20}$$

Method 1: Via Change of Variable

In [1], a function $H(n_1, n_2)$ was introduced, defined for each pair of non-negative integers (n_1, n_2) such that $n_1 + n_2$ is of the form 3^j for some $j \geq 0$. This function is the unique function satisfying:

$$H(0,1) = H(1,0) = 0,$$

$$H(3k_1, 3k_2) = 3H(k_1, k_2),$$

$$H(3k_1+1, 3k_2+2) = H(k_1+1, k_2) + 2H(k_1, k_2+1) + \log_2 3,$$

$$H(3k_1+2, 3k_2+1) = 2H(k_1+1, k_2) + H(k_1, k_2+1) + \log_2 3.$$

It was shown in [1] that there exists a unique continuous function h_3 defined on $[0,1]$ such that

$$h_3(i/3^j) = H(i, 3^j - i)/3^j, \;\; i = 0, 1, \cdots, 3^j; j = 0, 1, 2, \cdots. \tag{3.21}$$

Lemma 3.3.2. *Let P be the unique continuous periodic function on the real line, with period 1, such that*

$$h_3(x) = (x+1)P(\log_3(x+1)), \;\; 0 \leq x \leq 1. \tag{3.22}$$

$$h_3(x) = (x+2)P(\log_3(x+2)), \;\; 0 \leq x \leq 1. \tag{3.23}$$

Then

$$H_3(n)/n = P(\log_3 n), \;\; n \geq 1, \tag{3.24}$$

and therefore we may take P as the function P_3 in (3.20).

Change of Variable Method. From Lemma 3.3.2, the following change of variable method for obtaining $P_3(t)$ from $h_3(t)$ is now clear:

$$\begin{aligned} P_3(t) &= 3^{-t} h_3(3^t - 1), \;\; 0 \leq t \leq \log_3 2. \\ P_3(t) &= 3^{-t} h_3(3^t - 2), \;\; \log_3 2 \leq t \leq 1. \end{aligned} \tag{3.25}$$

Method 2: Via IFS Iteration

In the paper [1], it is shown that the following three mappings on $[0,1] \times \mathbb{R}$ are

contraction mappings:

$$\begin{aligned} T_0(x,y) &= (x,y)\begin{pmatrix} 1/3 & (\log_2 3)/3 \\ 0 & 1/3 \end{pmatrix} \\ T_1(x,y) &= (x,y)\begin{pmatrix} 1/3 & 0 \\ 0 & 1/3 \end{pmatrix} + (1/3, (\log_2 3)/3) \\ T_2(x,y) &= (x,y)\begin{pmatrix} 1/3 & -(\log_2 3)/3 \\ 0 & 1/3 \end{pmatrix} + (2/3, (\log_2 3)/3) \end{aligned} \tag{3.26}$$

It is also shown that the graph of the function $h_3(x)$ is the attractor of the iterated function system $\{T_0, T_1, T_2\}$. Using substitution $x = 3^t + 1$ and the change of variable (3.25), one transforms the mappings T_0, T_1, T_2 into mappings T_0^*, T_1^*, T_2^* on $[0,1]\times\mathbb{R}$ such that the graph of P_3 is the attractor of the IFS $\{T_0^*, T_1^*, T_2^*\}$. The mappings T_0^*, T_1^*, T_2^* are given in (3.9) with $C_3(1)$ and $C_3(2)$ defined in (3.8).

Start with the set $S_0 = \{(0,0)\}$. Form sets $S_1, S_2, \cdots$ as follows by iteration:

$$S_n = T_0^*(S_{n-1}) \cup T_1^*(S_{n-1}) \cup T_2^*(S_{n-1}), \quad n \geq 1.$$

For each n, S_n consists of 3^n points on

$$\{(t, P_3(t)) : 0 \leq t \leq 1\}.$$

Example 3.3.3. *In Figure 3.3(a), we have plotted the graph of h_3. We used Method 1 to transform $h_3(x)$ into $P_3(t)$ and we give the resulting plot of $P_3(t)$ in Figure 3.3(b). Then, we used Method 2 to obtain the set S_{12}, consisting of 3^{12} points on the curve $P_3(t)$, and we plotted these points to obtain the plot of the graph of $P_3(t)$ in Figure 3.1. As the reader can clearly see, the Figure 3.3(b) and Figure 3.1 plots are the same.*

2) Generating Compression Rate Curve C_4.

We have the sequence $\{H_4(n) : n \geq 1\}$ such that

$$H_4(1) = H_4(2) = H_4(3) = 0,$$

$$H_4(4k) = 4H_4(k), \quad k \geq 1,$$

$$H_4(4k+1) = 3H_4(k) + H_4(k+1) + 2, \quad k \geq 1,$$

$$H_4(4k+2) = 2H_4(k) + 2H_4(k+1) + \log_2 6, \quad k \geq 1,$$

$$H_4(4k+3) = H_4(k) + 3H_4(k+1) + 2, \quad k \geq 1.$$

The compression rate curve P_4 is a periodic function on the real line, with period 1, such that

$$H_4(n)/n = P_4(\log_4 n), \quad n \geq 1. \tag{3.27}$$

Method 1: Via Change of Variable

In [1], a function $H(n_1, n_2)$ was introduced, defined for each pair of non-negative integers (n_1, n_2) such that $n_1 + n_2$ is of the form 4^j for some $j \geq 0$. This function is the unique function satisfying:

$$H(0,1) = H(1,0) = 0,$$

$$H(4k_1, 4k_2) = 4H(k_1, k_2),$$

$$H(4k_1+1, 4k_2+3) = H(k_1+1, k_2) + 3H(k_1, k_2+1) + 2,$$

$$H(4k_1+2, 4k_2+2) = 2H(k_1+1, k_2) + 2H(k_1, k_2+1) + \log_2 6,$$

$$H(4k_1+3, 4k_2+1) = 3H(k_1+1, k_2) + H(k_1, k_2+1) + 2.$$

It was shown in [1] that there exists a unique continuous function h_4 defined on $[0,1]$ such that

$$h_4(i/4^j) = H(i, 4^j - i)/4^j, \quad i = 0, 1, \cdots, 4^j; j = 0, 1, 2, \cdots. \tag{3.28}$$

Lemma 3.3.4. *Let P be the unique continuous periodic function on the real line, with period 1, such that*

$$\begin{aligned} h_4(x) &= (x+1)P(\log_4(x+1)), \quad 0 \leq x \leq 1. \\ h_4(x) &= (x+2)P(\log_4(x+2)), \quad 0 \leq x \leq 1. \\ h_4(x) &= (x+3)P(\log_4(x+3)), \quad 0 \leq x \leq 1. \end{aligned}$$

Then

$$H_4(n)/n = P(\log_4 n), \quad n \geq 1,$$

and therefore we may take P as the function P_4 in (3.27).

Proof. Proof is omitted.

Change of Variable Method. From the Lemma, the following change of variable method for obtaining $P_4(t)$ from $h_4(t)$ is now clear:

$$\begin{aligned} P_4(t) &= 4^{-t}h_4(4^t-1), \ \ 0 \le t \le \log_4 2, \\ P_4(t) &= 4^{-t}h_4(4^t-2), \ \ \log_4 2 \le t \le \log_4 3, \\ P_4(t) &= 4^{-t}h_4(4^t-3), \ \ \log_4 3 \le t \le 1. \end{aligned} \tag{3.29}$$

Method 2: Via IFS Iteration

In the paper [1], it is shown that the following three mappings on $[0,1]\times\mathbb{R}$ are contraction mappings:

$$\begin{aligned} T_0(x,y) &= (x,y)\begin{pmatrix} 1/4 & 1/2 \\ 0 & 1/4 \end{pmatrix} \\ T_1(x,y) &= (x,y)\begin{pmatrix} 1/4 & (\log_2 6)/4 - 1/2 \\ 0 & 1/4 \end{pmatrix} + (1/4, 1/2) \\ T_2(x,y) &= (x,y)\begin{pmatrix} 1/4 & 1/2 - (\log_2 6)/4 \\ 0 & 1/4 \end{pmatrix} + (1/2, (\log_2 6)/4) \\ T_3(x,y) &= (x,y)\begin{pmatrix} 1/4 & -1/2 \\ 0 & 1/4 \end{pmatrix} + (3/4, 1/2) \end{aligned}$$

It is also shown that the graph of the function $h_4(x)$ is the attractor of the iterated function system $\{T_0, T_1, T_2, T_3\}$. Using the change of variable (3.29), one transforms the mappings T_0, T_1, T_2, T_3 into mappings $T_0^*, T_1^*, T_2^*, T_3^*$ on $[0,1]\times\mathbb{R}$ such that the graph of P_4 is the attractor of the IFS $\{T_0^*, T_1^*, T_2^*, T_3^*\}$ in (3.11)-(3.13) which is defined on C_4 as in (3.10).

Method 2: Finding $P_4(t)$ by IFS Iteration. Start with the set $S_0 = \{(0,0)\}$. Form sets $S_1, S_2, \cdots$ as follows by iteration:

$$S_n = T_0^*(S_{n-1}) \cup T_1^*(S_{n-1}) \cup T_2^*(S_{n-1}) \cup T_3^*(S_{n-1}), \ \ n \ge 1.$$

For each n, S_n consists of 4^n points on

$$\{(t, P_4(t)) : 0 \le t \le 1\}.$$

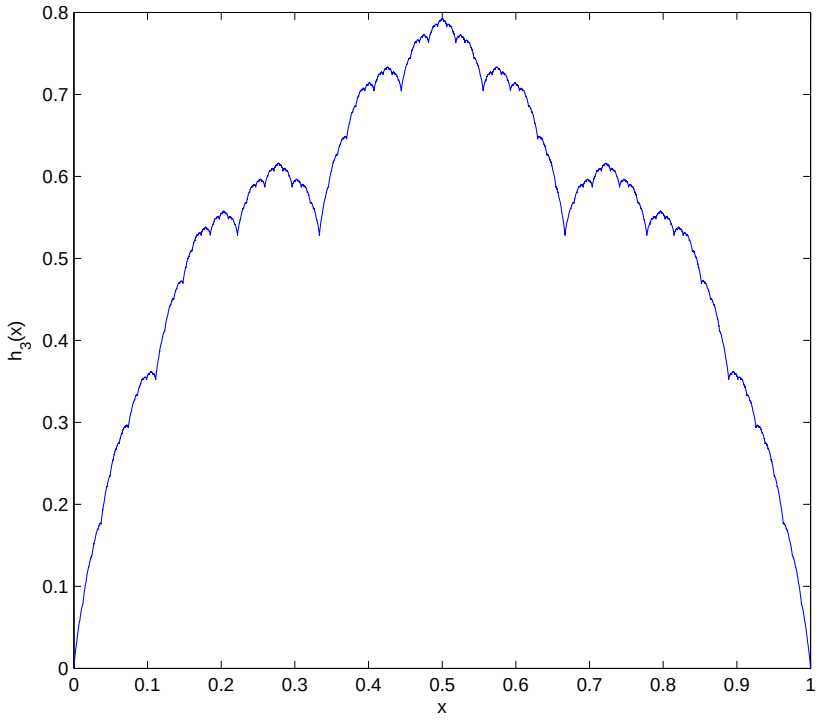
3.3.4 Execution Time

Table 3.1 specifies the execution time to get the N points on the curve C_3 and C_4 after k iterations of processes. The time unit is in seconds. The points for the second columns were obtained by direct entropy rate calculation of (3.1). For the third columns, the set of points was generated by k times of applying IFS to an arbitrary point on the curve C_3 and C_4. From Table 3.1, we make sure of the obvious fact that the point generation via IFS is much faster than that via the direct calculation of the entropy rate equation, (3.1). In other words, with IFS we can check out the behavior of the relative size of a data structure rather faster.

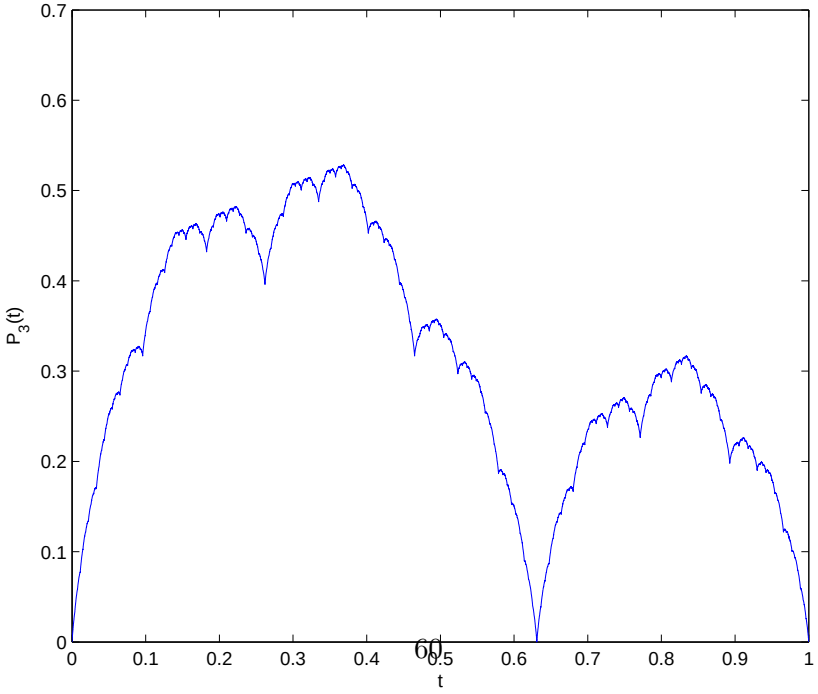
$\alpha = 3$			
n	$N = \sum_{k=1}^{n} 3^k$	$H(n)$ (sec)	IFS (sec)
9	29,524	88.97	0.0072
10	88,573	477.24	0.0146
11	265,720	2,835.05	0.0409
12	797,161	17,247.47	0.1221
13	2,391,484	114,580.03	0.3812

$\alpha = 4$			
n	$N = \sum_{k=1}^{n} 4^k$	$H(n)$ (sec)	IFS (sec)
8	87,381	211.40	0.0147
9	349,525	2,520.26	0.0524
10	1,398,101	30,172.91	0.2155
11	5,592,405	406,256.27	0.9003

Table 3.1: Execution time for generating the curves C_3 and C_4



(a) The graph of h_3



(b) Application of the change of variable method to the graph of h_3

Figure 3.3: Figures obtained before (a) and after (b) applying (3.25)

Chapter 4

Compression Algorithms for SCFG Ensembles

This chapter is about lossless compression algorithms for SCFG ensembles. For a compression algorithm to work properly, both encoder and decoder know which SCFG (along with the number of leaves of a tree) is used to construct trees in a tree ensemble. Then encoder and decoder use the information to optimally compress and decompress the tree.

4.1 A Lossless Compression Algorithm for a Nonequiprobable Tree Ensemble

In this section we describe how some SCFG ensemble in which the trees in the ensemble are not equiprobable is encoded using a lossless compression algorithm. An example of such SCFG ensembles is binary search tree model introduced in Section 1.5.1.

For the random binary search model, the probability distributions $\{p_n\}_{n=2}^{\infty}$ which agree with the notation in (1.3) are given by

$$p_n(i, n-i) = \frac{1}{n-1}, \quad i = 1, 2, \cdots, n-1, \tag{4.1}$$

([16]). As seen in Figure 1.4-1.5, $p_n(i, n-i)$ is equal for all i for a fixed n. Also the

entropy equation of a random tree in the model in Theorem 1.5.5 is rephrased as

$$H(T_n) = \log_2(n-1) + \frac{2}{n-1}\sum_{i=1}^{n-1} H(T_i),\ n \geq 3, \tag{4.2}$$

with $H(T_1) = H(T_2) = 0$ [16]. Exploiting that the entropy is represented in a recursion from (4.2) and that the probability of p_n is known and is uniform over all i from (4.1), one can losslessly compress a tree in binary search tree model. Trees in random binary search tree ensemble are not equiprobable. For example, the tree generated by the keys $3124, 3142, 3412$ has probability $1/8$ whereas the tree generated by the key 4312 has probability $1/24$ as seen in Figure 1.4. Because of the non-equiprobable characteristic of the tree ensemble, variable-length codewords are employed to compress a tree. If the known *a priori* and recursive entropy of a source are considered in the category of compression algorithms with variable-length codewords, arithmetic coding is one of the possible coding methods for compressing a binary search tree. The basic idea of arithmetic coding is to successively subdivide the interval between 0 and 1 according to the known probabilities of a source.

Here is how to encode a binary search tree T_n with $n \geq 3$ leaves using arithmetic coding. (The tree T_2 does not need a compression because it is trivial.) For a fixed $n \geq 3$, the probability that the left child v_1 of the root has i_1 leaves is $1/(n-1)$ $(1 \leq i_1 \leq n-1)$. Thus the interval $[0, 1)$ is divided into $n-1$ subintervals which are all equal in size and the i_1-th interval is chosen. If the subtree rooted at the left child v_2 of the v_1 has $i_2 \geq 3$ leaves, the i_1-th interval is again divided into $(i_1 - 1)$ subintervals each with size $1/(i_1 - 1)$. Repeat this dividing procedure until the number of leaves of the leftmost subtree reaches down 2. Then move upward until a not-yet-processed subtree is found which is rooted at the right child of some vertex. Traverse the tree from left to right keeping dividing the interval. Decoder also follows the same rules of traversing of a tree encoder has. For the arithmetic coding here, there is no interval for letting decoder know the time of termination. However, decoder can know when to terminate the process of decompressing by checking out whether all the subtrees in T_n have ≤ 2 leaves or not. If the first (or the second) interval is selected, that means the left subtree of a nonleaf vertex of T_n has one leaf (or two leaves).

Example 4.1.1. *Let $\mathcal{T}_{15}$ be an ensemble of binary search trees with* 15 *leaves. Figure 4.1 shows a binary search tree $t^* \in \mathcal{T}_{15}$. The letters labeling some of the vertices of t^**

(not the root) are i_j^L and i_j^R with $j = 1, 2, \cdots, 9$. Each of the letters denotes the number of leaves of a subtree rooted at a nonleaf vertex, and the sum of the labels of two children is the label of their parent vertex. For instance, $n = i_1^L + i_1^R$. The probability that two children from a parent with n leaves have i and $n - i$ leaves is $p_n(i, n-i)$ in (4.1). For the root and its two children in Figure 4.1, it is $p_n(i_1^L, i_1^R) = 1/(n-1)$.

The first step is to divide the probability space between 0 and 1 into $n - 1$ subintervals according to the probability of the two children of the root, $p_n(i_1^L, i_1^R)$. Then the fifth interval is chosen out of 14 intervals because $i_1^L = 5$ means that the left subtree of the root has 5 leaves. So the selected interval is $I_1 = [4/14, 5/14)$. Next, the label $i_1^L = 5$ splits into i_2^L and i_2^R. There are 4 possible subintervals of I_1 for the subtree rooted at the vertex with label i_1^L and the second one is selected because $i_2^L = 2$ leading to the resultant interval, I_2 is the portion $[1/4, 2/4)$ of the interval $[4/14, 5/14)$ which is $[4/14 + 1/56, 4/14 + 2/56) = [17/56, 18/56)$. In the same way, keep dividing the interval until all subtrees with > 2 leaves are covered. The traverse across the tree t^* is given in the following direction.

$$n \to i_1^L \to i_2^L \to i_2^R \to i_3^L \to i_1^R \to i_4^L \to i_5^R \to i_6^R \to i_4^R \to i_7^R \to i_8^L \to i_9^L \to i_9^R.$$

After the successive dividing intervals one can get I_9 at last. The final task is to find any number with the shortest binary representation in the interval I_9, which is the resultant codeword of the binary search tree t^*.

Remark. *The probability of the tree t^* in Figure 4.1 is*

$$P_{15}(t^*) = \frac{1}{n-1} \cdot \frac{1}{i_1^L - 1} \cdot \frac{1}{i_2^R - 1} \cdot \frac{1}{i_3^R - 1} \cdot \frac{1}{i_4^L - 1} \cdot \frac{1}{i_5^R - 1} \cdot \frac{1}{i_4^R - 1} \cdot \frac{1}{i_7^R - 1} \cdot \frac{1}{i_8^L - 1}.$$

4.2 Lossless Compression Algorithms for Equiprobable Tree Ensembles

Now we consider random tree ensembles with equiprobability. As opposed to the nonequiprobable tree ensemble in the previous section, fixed-length codewords are employed to losslessly compress a random tree in an ensemble where trees are equiprobable. In this section we illustrate two simple lossless compression algorithms. First, the structure part of a balanced tree will be encoded and decoded. Then the description of

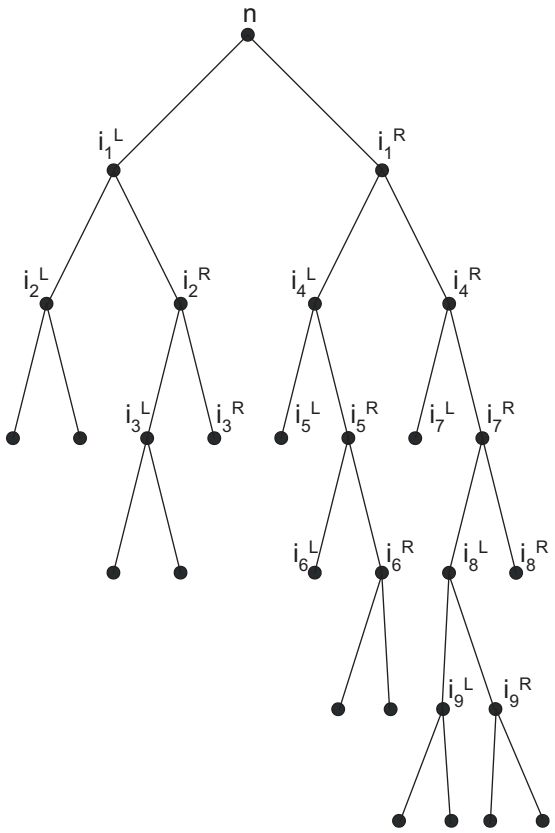


Figure 4.1: An encoding example of a binary search tree with $n = 15$.

compression method for a random tree associated with a string in a hierarchical type class will follow.

4.2.1 Compression Method for an α-balanced Tree

Let us consider a bisection tree (i.e., 2-balanced tree) first. A nonleaf vertex v in the tree has two children at which there are two subtrees rooted. Whether a binary bit is assigned or not is determined according to the numbers of leaves of the two subtrees. If they are equal or both of them are one (i.e., the two children are leaves), no bit is assigned. When they differ (the difference is always one for a 2-balanced tree) and the left subtree has one more leaves then v is assigned bit 0. If the left subtree has

less leaves, v is assigned bit 1. Repeat this procedure until all nonleaf vertices in the tree get assigned each of their own bit. The distributed binary bits over the tree are then collected in a breadth-first order and the resulting sequence is the codeword of the 2-balanced tree.

Example 4.2.1. *Let $\alpha = 2$ and $n = 11$. Figure 4.2 shows a 2-balanced tree with the α and n. The root labeled with 11 splits into the left child with 6 and the right child with 5, so the root is assigned bit 1 which is denoted as a circled number. No bit is assigned to the next vertex, which is the left child of the root labeled with 6 because both of its two children are labeled with 3. The right child of the root labeled with 5 has two children each of which is labeled with 2 and 3. Since the label 2 on the left is smaller than the label 3 on the right, bit 0 is assigned to the right child of the root. The rest of the vertices in the tree are either leaves or the vertices labeled with 2 which do not need to be encoded. The binary codeword 10011 is obtained by visiting the tree in a breadth-first order.*

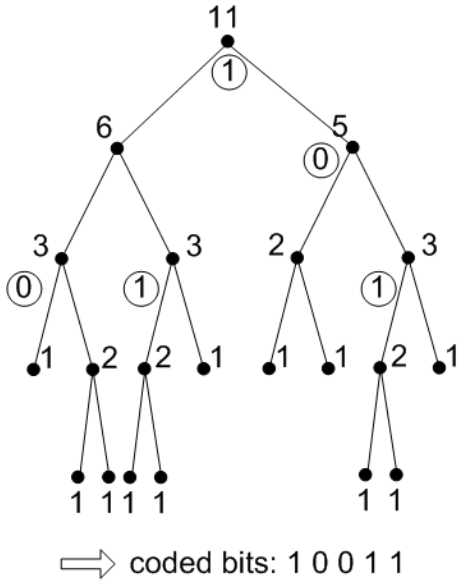


Figure 4.2: Encoded binary sequence for a 2-balanced tree with $n = 11$.

For an α-balanced tree, a nonleaf vertex v in the tree has their children such that

the number of leaves down from v is split into the numbers of leaves from the children of v most evenly. If the number of leaves k down from v is divisible by α, or k is less than α, v need not to be assigned a bit. If k is not divisible by α, $\binom{\alpha}{\mathrm{mod}(k,\alpha)}$ permutations are possible to distribute the number k over α children as the number of leaves descending from each of them. We enumerate the permutations in a lexicographic order to make a list and get codewords from the indices of the list as the enumerative coding scheme in [13] suggested. Let $i \geq 0$ be an index of the ordered list and it is assigned to the vertex v. So every nonleaf vertex in the tree is assigned an integer in the range of $[0, 1, \cdots, \binom{\alpha}{\mathrm{mod}(k,\alpha)} - 1]$. All the nonleaf vertices are assigned such integers as long as each number of leaves associated with the vertices is not divisible by α and is greater than α. Then an integer sequence can be obtained by collecting all the integers as in the way that one does for a 2-balanced tree. The sequence of α-numbers is converted into a sequence of binary numbers, which is a resulting codeword.

Example 4.2.2. *Let us consider a* 4*-balanced tree* T_{27} *with* 27 *leaves illustrated in Figure 4.3. Because* $\mathrm{mod}(27, 4) = 3$*, there are* $\binom{4}{3} = 4$ *ways that the number* 27 *splits into three* 7*'s and one* 6*, which is ordered with indices from* 0 *to* 3 *in the leftmost table below. The labels (i.e., the number of leaves of the* 4 *subtrees) at level* 1 *are* $7, 7, 6, 7$ *from left to right and the index* 2 *is assigned to level* 1*. At level* 2 *of the tree, there are three subtrees having* 7 *leaves and one subtree having* 6 *leaves. The first, second and fourth subtrees with* 7 *leaves have labels on their children* $(1, 2, 2, 2), (2, 2, 2, 1)$*, and* $(2, 2, 1, 2)$*, respectively. According to the second table,* 0*,* 3*,* 2 *are assigned to each of the subtrees. The third subtree with* 6 *leaves at level* 2 *has labels on its children* $(2, 1, 1, 2)$ *and the index* 3 *is assigned to the subtree according to the rightmost table below.*

ordered label	*index*
6 7 7 7	*0*
7 6 7 7	*1*
7 7 6 7	*2*
7 7 7 6	*3*

ordered label	*index*
1 2 2 2	*0*
2 1 2 2	*1*
2 2 1 2	*2*
2 2 2 1	*3*

ordered label	*index*
1 1 2 2	*0*
1 2 1 2	*1*
1 2 2 1	*2*
2 1 1 2	*3*
2 1 2 1	*4*
2 2 1 1	*5*

Table 4.1: Three sets of labels and indices for a 4*-balanced tree*

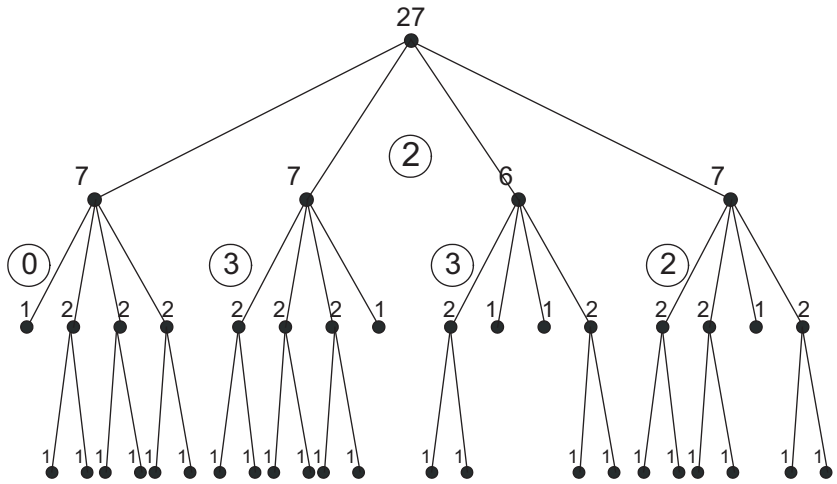


Figure 4.3: Encoded binary sequence for a 4*-balanced tree with* 27 *leaves.*

Each index assigned to the five subtrees in T_{27} *is converted into a binary number. Suppose an index is chosen out of* k *possible indices in a table. Then* $\log_2 k$ *binary bits are spared to represent the index. Thus,* $2 = 10, 0 = 00, 3 = 11, 3 = 011$*, and* $2 = 10$*. The five indices in binary numbers are collected in a breath-first order being ended up with the resultant sequence* $x = 10001101110$*.*

According to the equation in Example 3.1.3, the entropy of the 4*-balanced tree* T_{27} *is* $H_4(27) = 8 + \log_2 6$ *and the minimum length of fixed-length binary codeword spent to represent the* T_{27} *is* $\lceil H_4(27) \rceil = 11$ *bits. The sequence* x *obtained via the compression algorithm is of length* 11*, which is equal to the minimum.*

4.2.2 Compression Method for a Hierarchical Type Tree

Here is a method of encoding and decoding of a string in a hierarchical type class $S_{m,\alpha}$ with $m \geq 2$ and $\alpha \geq 2$. Assume $S_{m,\alpha}$ be of order $J \geq 1$. Then the strings in $S_{m,\alpha}$ are of length α^J and their associated α-ary trees $T_J(\alpha)$ are of depth J.

Consider a tree $T_J(2)$. Let x_R be the string placed on the right child and x_L be the string placed on the left child of a nonleaf vertex v of a tree induced by a hierarchical type class $S_{m,2}$. The string x labeling the v is split into twofolds such that x_R and x_L are of the same length and x is in the form of either $x = x_L * x_R$ or $x = x_R * x_L$ where

$*$ means concatenation. When x_R and x_L are equal, there is no bit assigned to v. If x_R precedes x_L lexicographically, v is assigned bit 1. If x_L precedes x_R, it is assigned bit 0. And the rest of the codeword-forming procedure is same as that for the 2-balanced trees.

The implementation of encoder for a general case ($\alpha > 2$) is as follows. If a J-string x in a hierarchical type class is considered with $J \geq 1$, each entry of x labels each of the α^J leaves of a rooted α-tree left to right. At the j-th level of the tree ($0 \leq j \leq J$), there are α^j strings of length α^{J-j}. Let us look at the bottom level (level J) first. From left to right every α children (descending from a nonleaf vertex at the $(J-1)$-th level) is labeled with α strings $x^{(1)}, x^{(2)}, \cdots, x^{(\alpha)}$ of length 1 each. The α strings are enumerated in a lexicographical order with q indices. The index of the strings ranges from 0 to $q-1$ and encoder represents the α strings as a pair of two numbers, of which the first entry is the index of the ordering and the second entry is q. Then the parent vertex of the α children is assigned a string which is the concatenations of the ordered version of the α strings. In this fashion, the rest of the $\alpha^{J-1} - 1$ sets of α children are assigned such pairs of integers. Encoder processes upwards from the leaves to the root, so what it does next is to compare the first α strings of length α each at the $(J-1)$-th level (which are obtained from the J-th level) and assign a pair of index and the total number of possible indices. Also the concatenation of the α strings are lexicographically re-ordered and be assigned to the vertex at level $J-2$ which is the parent of the leftmost α vertices at level $J-1$. Repeat this procedure until pairs of integers are assigned to all the α branches in the tree that are derived from a parent. The pairs are gathered in a row in top-down order, and then re-ordered according to the second coordinate of the pairs. Because decoder also knows about the second entries of the pairs, only the first entries are extracted and converted into a sequence of binary bits. This is the resultant encoded sequence and decoder interprets the sequence backwards.

Example 4.2.3. *Let $\alpha = 3$. Each entry of string $x = ACBBBACAC$ in hierarchical type class $S_{3,3}$ labels each leaf of a 3-ary tree which is represented by the type class. The leftmost three branches with labels A, C, and B are indexed first. The ordered numbers of A, C, B is $(0, 2, 1)$ which is located on the second from the top on Table 4.2 and represented as $(1, 6)$. Their parent vertex is labeled with ABC, which is a re-ordered string. The next three branches which are labeled with B, B, A each, are assigned the*

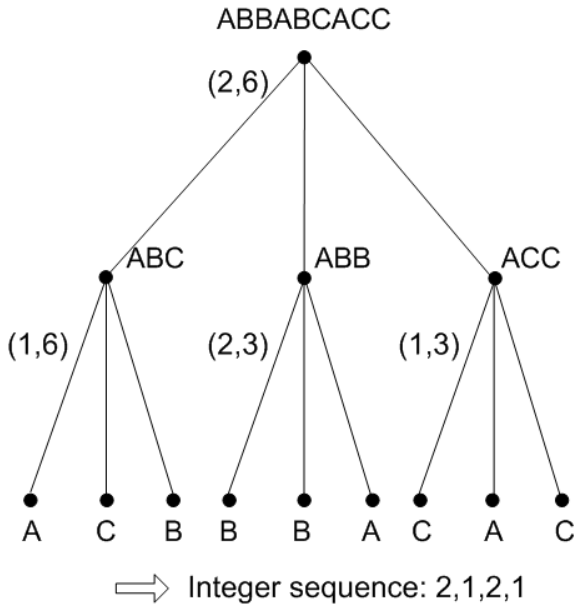


Figure 4.4: Encoded sequence of a string in a hierarchical type class with $m = 3$ and $\alpha = 3$.

pair, $(2, 3)$ *because there are only three possibilities in the ordering of permutations of* A, B, B*. Also the new label of the vertex from which they descend becomes* ABB*. The last three strings* C, A, C *are represented as* $(1, 3)$ *and the string* ACC *is placed on their parent vertex. So the new three strings at level 1 are* ABC*,* ABB*,* ACC *from left to right. The ordering numbers of the three is* $(1, 0, 2)$ *and the corresponding index 2 along with the number 6 is assigned to level 1 as seen in Figure 4.4.*

Visiting the tree in breadth-first order, one obtains the 4 pairs of numbers $(1, 6)$*,* $(2, 3)$*,*$(1, 3)$*,* $(2, 6)$*. They are arranged again along with the second coordinates in increasing order, which now would be* $(2, 3), (1, 3), (1, 6), (2, 6)$*. Then only the first coordinates* $2, 1, 1, 2$ *is extracted from the pairs and the encoded binary codeword will be* 1001001010.

The two compression methods are basically similar in the sense that they employ fixed-length codewords due to the equal probabilities of the objects to be compressed in a set they belong. Also both encoder and decoder have knowledge (of the total number

Table 4.2: 3-tuple orderings with 6 possible indices

ordering	*index*	*codeword*
(0,1,2)	*0*	*000*
(0,2,1)	*1*	*001*
(1,0,2)	*2*	*010*
(1,2,0)	*3*	*011*
(2,0,1)	*4*	*100*
(2,1,0)	*5*	*101*

of leaves of a tree or of the string labeled on the root vertex) in advance.

Note that the difference between the two compression methods above is as follows. In the compression of a balanced tree, the comparison is made between the numbers of leaves from α children of a nonleaf vertex. On the other hand, the compression of a string in a hierarchical type class involves comparison of between the strings labeled on siblings.

Chapter 5

Fractal Properties of Compression Rate Curves

5.1 Notation of Fractal

Concept of Fractality. Fractal in this thesis refers to a geometric object that can be subdivided in parts, each of which is a copy of (or at least similar to) the whole of reduced size. The followings are the characteristics of a fractal.

- (a) Irregular and fragmented shape: a fractal has an irregular shape and it can be broken up into multiple smaller pieces.
- (b) Self-similarity: each of the split pieces is similar to the whole object.
- (c) Scale invariance: property (b) applies to all fragments in any scale. In other words, one can see a fragment and the whole are of a similar shape no matter how much the object is magnified.

Compression rate curves C_α for an α-balanced tree is a fractal in the sense that it has self-similar patterns in any scale.

5.2 Fractal Dimensions

Definition of Hausdorff Dimensions [10]. Suppose C be a geometric object which

is a subset of $\mathbb{R}^n$ for any positive integer n and let s be a non-negative real number. Let δ be any positive real number. Let U_i be a non-empty subset of $\mathbb{R}^n$ for $i = 1, 2, \cdots$. Hausdorff measure on C is defined as

$$H^s = \liminf_{\delta \to 0} \left\{ \sum_{i \geq 1} |U_i|^s : \ C \subset \bigcup_{i \geq 1} U_i, \ 0 \leq |U_i| \leq \delta \right\}, \tag{5.1}$$

where $|U_i|$ means the diameter of each U_i. Also it is known that $H^s \geq 0$.

Hausdorff dimension is defined using Hausdorff measure. The value s giving the minimum Hausdorff measure is referred to as Hausdorff dimension $\dim_{\mathrm{H}}$ and is written as

$$\dim_{\mathrm{H}} = \inf\{s \geq 0 : \ H^s = 0\}.$$

Box Dimension. Hausdorff dimension is well-defined but it is hard to obtain practically. It is defined based on a measure, which employs a unit with flexible size. The $\{U_i\}$'s are not guaranteed to have the diameters of the same size and even they are possible to have shapes different to each other. So practically determining their sizes and shapes imposes difficulty in calculating the Hausdorff dimension. A fractal dimension which is easier to calculate can be consider *box dimension*. The box dimension is measured using sets of equal size. (One can roughly regard the box dimension as being another version of (5.1) with $U_i = U$ which is a square of side-length δ.) Most of cases each of the sets consists of squares of the same size but some adopt other geometric shapes of the same size as elements of the sets.

Definition. Let C be a non-empty subset of $\mathbb{R}^n$. Consider a real number δ such that $0 < \delta < 1$. For fixed δ, lay a grid of square cells with side of length δ over C and count the number N_δ of squares that contains the contour of C. Box dimension $\dim_B$ is defined as the ratio of logarithms of N_δ and $1/\delta$ for small enough δ.

$$\dim_B = \lim_{\delta \to 0} \frac{\log N_\delta}{-\log \delta}. \tag{5.2}$$

For any $C \subset \mathbb{R}^n$, the relation of box dimension $\dim_{\mathrm{B}}$ and Hausdorff dimension $\dim_{\mathrm{H}}$ of C is as follows.

$$\dim_{\mathrm{H}} \leq \dim_{\mathrm{B}} \tag{5.3}$$

5.3 Box Dimension of Compression Rate Curves

In this section we consider the box dimension (5.2) for the compression rate curve for α-balanced trees C_α for $\alpha \geq 2$.

Box Dimension of Compression Rate Curve C_2.

Theorem 5.3.1. *The graph of $P_2(t)$ which is compression rate curve C_2 for 2-balanced trees is of box dimension 1.*

Lemma 5.3.2. *Let $f : [0,1] \to R$ be a continuous function whose graph is of box dimension 1. Let ϕ be a differentiable, strictly increasing function from $[0,1]$ onto itself such that for some $\epsilon > 0$,*

$$d\phi(x)/dx \geq \epsilon, \quad 0 \leq x \leq 1.$$

Let $g : [0,1] \to R$ be the function

$$g(x) = f(\phi(x)), \quad 0 \leq x \leq 1.$$

Then the graph of g is of box dimension 1.

Lemma 5.3.3. *Let $f : [0,1] \to R$ be a continuous function whose graph has box dimension 1. Let $\phi : [0,1] \to R$ be any Lipschitz function. Let $g : [0,1] \to R$ be defined by*

$$g(x) = \phi(x)f(x), \quad 0 \leq x \leq 1.$$

Then the graph of g has box dimension 1.

Box Dimension of Compression Rate Curve C_α.

Theorem 5.3.4. *Fix $\alpha \geq 2$. Compression rate curve C_α for an α-balanced tree is of box dimension 1.*

5.4 Estimation of Box Dimension

In this section we discuss estimations of the box dimension of compression rate curve C_2 for a 2-balanced tree.

5.4.1 Box Counting Estimation

In Section 5.3 the exact box dimension of the curve C_2 was given by an indirect method using two different IFS's. It was obtained by transformations of the box dimension of the attractor of the affine IFS defined in (2.5). In this section, however, the box dimension is calculated only with the curve C_2 graphically without using the information of its IFS. The value of the box dimension which is obtained is not exact and we have to rely on the estimation of it. A method of box dimension estimation is studied [10].

According to Chapter 11 of [10], the upper and lower bounds of the box dimension of C are given as

$$\frac{1}{\delta}\sum_{i=0}^{\lfloor\frac{1}{\delta}\rfloor-1} R[i\delta,(i+1)\delta] \;\le\; N_\delta(C) \;\le\; 2\lfloor\frac{1}{\delta}\rfloor + \frac{1}{\delta}\sum_{i=0}^{\lfloor\frac{1}{\delta}\rfloor-1} R[i\delta,(i+1)\delta] \tag{5.4}$$

for small δ, where R is the maximum range of C_2 over an interval $[t_1,t_2]$,

$$R[t_1,t_2] \triangleq \sup_{t_1\le k,l\le t_2} |C_2(k)-C_2(l)|\,.$$

For the curve C_2, $\dim_B$ is estimated by computing (5.4) with Matlab program. The estimation is

$$1.063 \;\le\; \dim_B(C_2) \;\le\; 1.103$$

with the size of $\delta = 1.91e-6$.

The estimate of the box dimension of C_2 is off by more than 0.063 considering the box dimension $\mathrm{d_{box}} = 1$ of C_2 obtained in Section 5.3.

5.4.2 Linear Approximation

The mappings (3.7) in case of $\alpha = 2$ can be simplified by a transformation of f : $(\log_2 x, y) \to (x, y)$ on $(x, y) \in [0,1]\times[0,1]$ such as

$$\begin{aligned} U_0(x,y) &= \left(\tfrac{x+2}{2}, \tfrac{(x+1)y+x}{x+2}\right) \\ U_1(x,y) &= \left(\tfrac{x+3}{2}, \tfrac{(x+1)y+1-x}{x+3}\right). \end{aligned} \tag{5.5}$$

Each $U_i,\ (i=0,1)$ can be approximated as a affine combination as

$$U_i(x,y) \approx (x-x_0, y-y_0)J_{U_i}(x_0,y_0) + U_i(x_0,y_0),$$

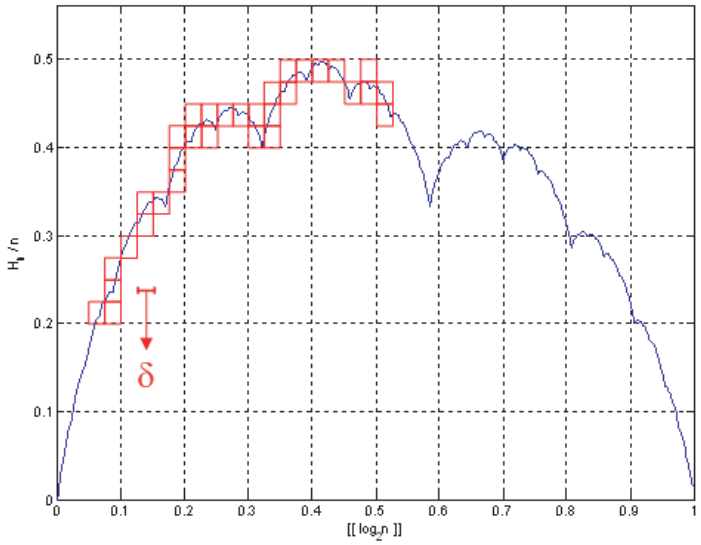


Figure 5.1: Unit δ for getting the estimation of $\dim_{\mathrm{B}}(C_2)$

where the function J_{U_i} is the Jacobian matrix of U_i at a point (x_0, y_0) in $[0,1] \times [0,1]$.

$$
\begin{aligned}
J_{U_i}(x_0, y_0) &= \begin{bmatrix} \frac{\partial J_{U_{i,1}}}{\partial x}\Big|_{(x_0,y_0)} & \frac{\partial J_{U_{i,2}}}{\partial x}\Big|_{(x_0,y_0)} \\ \frac{\partial J_{U_{i,1}}}{\partial y}\Big|_{(x_0,y_0)} & \frac{\partial J_{U_{i,2}}}{\partial y}\Big|_{(x_0,y_0)} \end{bmatrix} \\
&= \begin{cases} \begin{bmatrix} \frac{1}{2} & \frac{y_0+2}{(x_0+2)^2} \\ 0 & \frac{x_0+1}{x_0+2} \end{bmatrix}, & i = 0 \\ \begin{bmatrix} \frac{1}{2} & \frac{2y_0-4}{(x_0+3)^2} \\ 0 & \frac{x_0+1}{x_0+3} \end{bmatrix}, & i = 1 \end{cases} \\
&= \begin{bmatrix} a_i & b_i \\ 0 & d_i \end{bmatrix}.
\end{aligned}
\tag{5.6}
$$

According to Theorem 3.1 of [11], for a positive integer N, the attractor of the IFS

$$
U_n(x, y) \approx (x, y) \begin{pmatrix} a_i & b_i \\ 0 & d_i \end{pmatrix} + (e_i, f_i), \quad i = 0, 1 \cdots, N-1
$$

has box dimension $\mathrm{d_{box}}$ which is the solution d of

$$\sum_{i=0}^{N-1} |d_i|\, a_i^{d-1} = 1, \tag{5.7}$$

Plugging (5.6) in (5.7) leads to

$$\begin{aligned} \frac{1}{2^{d-1}} \left(\left| \frac{x_0+1}{x_0+2} \right| + \left| \frac{x_0+1}{x_0+3} \right| \right) &= 1 \\ d = \log_2 \left(\left| \frac{x_0+1}{x_0+2} \right| + \left| \frac{x_0+1}{x_0+3} \right| \right) &+ 1. \end{aligned} \tag{5.8}$$

Because the right-hand side of (5.8) is a monotonically increasing function with respect to the variable x_0. The box dimension is

$$0.7370 \leq \mathrm{d_{box}} \leq 1.2224.$$

Discussion. According to the definition of fractal put by Benoit Mandelbrot [19], a fractal is a set for which its Hausdorff dimension strictly exceeds its topological dimension. In Section 5.3, however, it turned out that the box dimension (then of course, the Hausdorff dimension as well by (5.3)) of a C_α for $\alpha \geq 2$ does not exceed the topological dimension: they are equal. Although the curve C_α does not satisfy the condition imposed by Mandelbrot, it can be regarded as being a fractal in the sense that it satisfies the fractal properties mentioned earlier in this chapter.

Chapter 6

Conclusions and Future Work

In this thesis we discussed the notion of structured data that consists of two parts: the structure part and the data part. The structure part refers to a finite random graph and the data part is a collection of data labels that are assigned to parts of the structure. Because of the divided character of the structured data into two parts, information conveyed by structured data also is in the form of two types. The information of structured data can be regarded as the approximate minimal length of a fixed-length binary codeword when we encode the structured data with a lossless compression algorithm. One of the key motivations behind the study of structured data is an increased demand for storage systems and data structures that can hold large amount of information. As part of a new data structure, "structured data" was dealt with and we focused on the structure part of the structured data in the thesis.

As the structure part, we surveyed several types of graphs including Erdös-Rényi graphs, random rooted trees, and grammar-based structured random trees. Among these, the class of random rooted trees induced by a context-free stochastic grammar was our main concern. The information theory of balanced trees, generated by such a grammar having a special characteristic, was investigated. The entropy of balanced trees was expressed in a recursive form. Also the asymptotic behavior of the entropy rate of balanced trees was analyzed along with its properties. The concepts of iterated function system (IFS) and its attractor were introduced for more accurate and brief description of the asymptotic behavior.

Our main contribution in this thesis is to generalize the results obtained in the

binary balanced trees to the results for the case of α-ary trees. The entropy and the asymptotic entropy rate of an α-ary balanced tree were given as the number of leaves of the tree grows without bound. Also an explicit form of equation was given that links an affine IFS of a graph of hierarchical entropy function to the IFS of compression rate curve for an α-ary balanced tree. With the specification of the IFS, the compression rate curve can be generated faster and the amount of time for the generating was shown. Lastly, an lossless compression algorithm was described for an α-ary balanced tree. The algorithm exploits the characteristics of the balanced grammar with which the tree is built and assigns a fixed-length codeword to some of nonleaf vertices of the tree. The algorithm is optimal in the sense that the resultant codeword length reaches down the entropy of the balanced tree compressed.

The coding algorithm for trees corresponding to strings in a hierarchical type class in Section 4.2.2 compresses both a structure and data strings in a structured data. Up to this point, there have not been much degree of freedom in forms of random trees in the category of structured data. So far, lossless compressing of both label and structure has covered only a specific form of tree structure, α-ary trees on which string labels are divided into α substrings as the direction goes from the root to leaves. The α-balanced trees has more degree of freedom than the α-ary random trees in that the trees can have any number of leaves. However coding of the α-balanced trees is for compressing of just the structure part of structured data, and the data part is not included as the process of the coding.

There would be demands for some time ahead for the lossless compression of more wide range of random tree forms as well as data labels on them. Thus, it is pertinent to develop a compression algorithm that deals with a labeled random tree which is not under strict restriction of the tree form. For example, a random rooted tree does not have to be α-ary and the numbers of leaves of subtrees from a nonleaf vertex in the tree do not have to be in as equal as possible. A tree in a labeled random rooted tree ensemble can be compressed via an algorithm that exploits the SCFG of the SCFG's own, which defines the rules of both the structure and data label.

In addition to development of the compressing algorithm, calculation of the entropy of such a random tree ensemble would be also needed. The entropy of the α-balanced tree ensemble serves as an indicator of how well the coding algorithm compresses a tree

in the ensemble. Likewise, entropy of the generalized random tree ensemble should be preceded to estimate the performance of an lossless algorithm for the ensemble.

References

[1] J. Kieffer, "A Catalog of Self-Affine Hierarchical Entropy Functions," submitted to *Algorithms*, 2011.

[2] J. Kieffer, "Hierarchical Type Classes and Thier Entropy Functions," In *Proceedings First Internatinoal Conference on Data Compression, Communication and Processing (CCP 2011)*; Editor, Carpentieri, B.; IEEE Press: Piscataway, NJ, 2011.

[3] S. Oh and J. Kieffer, "Fractal Compression Rate Curves In Lossless Compression of Balanced Trees," *Proc. IEEE Intern. Symp. Inform. Theory, Austin, Texas, June, 2010.*

[4] R. Cameron, "Source encoding using syntactic information source models," *IEEE Trans. Inform. Theory*, Vol. 34, pp. 84350, July 1988.

[5] E. Kawaguchi and T. Endo, "On a method of binary-picture representation and its application to data compression," *IEEE Trans. Pattern Anal. Machine Intell.* , Vol. PAMI-2, pp. 275, 1980.

[6] E. Kourapova and B. Ryabko, "Application of formal grammars for encoding information sources," *Probl. Inform. Transm.* , Vol. 31, pp. 236, 1995.

[7] J. Storer and T. Szymanski, "Data compression via textual substitution," *J. Assoc. Comput. Mach.* , Vol. 29, pp. 92851, 1982.

[8] J. Kieffer and E.H. Yang, "Grammar-based codes: a new class of universal lossless source codes," *IEEE Trans. Inform. Theory*, Vol. 46, pp. 73754, 2000.

[9] J. Kieffer and E.H. Yang, "Structured Grammar-Based Codes for Universal Lossless Data Compression," *Communications in Information and Systems*, Vol. 2, pp. 292, 2002.

[10] K. Falconer, *Fractal Geometry.* John Wiley & Sons, Chichester, England, 2003.

[11] M. Barnsley, *Fractals Everywhere.* Academic Press, Boston, 1993.

[12] L. Peshkin, "Structure induction by lossless grpha compression," *Proceedings DCC*, Snowbird, 2007.

[13] T. Cover, "Enumerative Source Encoding," *IEEE Trans. Inform. Theory*, Vol. 19, No. 1, pp. 73-77, Jan. 1973.

[14] T. Cover and J. Thomas, *Elements of Information Theory*, 2nd Ed. New York, Wiley, 2006.

[15] R. Sedgewick and P. Flajolet, *An introduction to the analysis of algorithms*, Addison-Wesley, Reading, MA, 1996.

[16] J. Kieffer, E.H. Yang, and W. Szpankowski, "Structural Complexity of Random Binary Trees," *Proc. IEEE Intern. Symp. Inform. Theory, Seoul, Korea, June-July*, 2009.

[17] W. Szpankowski, *Average Case Analysis of Algorithms on Sequences.* Wiley, New York, 2001.

[18] Y. Choi and W. Szpankowski, "Compression of Graphical Structures," *Proc. IEEE Intern. Symp. Inform. Theory, Seoul, Korea, June-July, 2009.*

[19] B. Mandelbrot, *The fractal geometry of nature,* W. H. Freeman and Company, San Francisco, 1977.

Appendix A

Proofs of Theorems

Proof of Theorem 1.5.5. For $n \geq 3$, let $q_n = H(T_n) - \log_2(n-1)$. Then we have to show

$$q_n = 2n \sum_{i=1}^{n-2} \frac{\log_2 i}{(i+1)(i+2)}, \quad n \geq 3. \tag{A.1}$$

The formula is true for $n = 3$ because $q_3 = 0$. Fix $N > 3$. As our induction hypothesis, we assume formula (A.1) holds for all n satisfying $3 \leq n \leq N-1$. Our proof by induction will be complete once we show that the formula holds for $n = N$. By formula (1.5), we have

$$H(T_N) = \log_2(N-1) + \frac{2}{N-1} \sum_{k=1}^{N-1} H(T_k),$$

which yields

$$\begin{aligned} q_N &= \frac{2}{N-1} \sum_{k=3}^{N-1} [q_k + \log_2(k-1)] \\ &= \frac{2}{N-1} \sum_{k=3}^{N-1} \log_2(k-1) + \frac{2}{N-1} \sum_{k=3}^{N-1} q_k, \end{aligned}$$

using the fact that $H(T_1) = H(T_2) = 0$. Substituting

$$q_k = 2k \sum_{i=1}^{k-2} \frac{\log_2 i}{(i+1)(i+2)},$$

and reversing the order of the two summations, we obtain

$$q_N = \frac{2}{N-1}\sum_{k=3}^{N-1}\log_2(k-1) + \frac{2}{N-1}\left[\sum_{i=1}^{N-3}\sum_{k=i+2}^{N-1}\frac{(2k)\log_2 i}{(i+1)(i+2)}\right].$$

In this formula, we can substitute

$$\sum_{k=i+2}^{N}(2k) = -(i+1)(i+2) + (N-1)N$$

and then re-group the terms as

$$\begin{aligned} q_N &= \frac{2}{N-1}\left[\sum_{k=3}^{N-1}\log_2(k-1) - \sum_{i=1}^{N-3}\log_2 i\right] + 2N\sum_{i=1}^{N-3}\frac{\log_2 i}{(i+1)(i+2)} \\ &= \frac{2\log_2(N-2)}{N-1} + 2N\sum_{i=1}^{N-3}\frac{\log_2 i}{(i+1)(i+2)} \\ &= 2N\sum_{i=1}^{N-2}\frac{\log_2 i}{(i+1)(i+2)}, \end{aligned}$$

completing the proof by induction.

□

Proof of Theorem 1.5.6. For each $n \geq 1$, pick 2^k, 2^{k+1}, the unique consecutive powers of two such that

$$2^k \leq n < 2^{k+1}.$$

It is easy to show by mathematical induction that

$$H(n) = H^*(n - 2^k, 2^{k+1} - n).$$

(One exploits the fact that the numbers $\{H(n)\}$ and the numbers $\{H^*(i,j)\}$ are generated by a similar "divide-and-conquer" recurrence.) Because of the relationship between H^* and h, we now have

$$H(n) = n(2^k/n)h\left(\frac{n}{2^k} - 1\right).$$

To complete the proof, we just have to show that

$$(2^k/n)h\left(\frac{n}{2^k} - 1\right) = P(\log_2 n). \tag{A.2}$$

Setting $x = (n/2^k) - 1$ in equation (1.7), we obtain

$$h\left(\frac{n}{2^k} - 1\right) = (n/2^k)P(\log_2(n/2^k)) = (n/2^k)P(\log_2 n),$$

and then multiplying both sides by $2^k/n$ we obtain (A.2), completing the proof. □

Proof of Theorem 2.1.1. Proof. Let 2×2 matrix A_i be the matrix of the linear part of the affine transformation T_i $(i = 0, 1)$ such that $T_i(x_1) = x_1 A_i + b$ with a constant real vector $b \in \mathbb{R}^2$. That is,

$$A_0 = \begin{bmatrix} 1/2 & 0 \\ 1/2 & 1/2 \end{bmatrix}, \quad A_1 = \begin{bmatrix} 1/2 & 0 \\ -1/2 & 1/2 \end{bmatrix}.$$

Then the condition of a contraction (2.3) can be written as

$$(x_1 - x_2)A_i A_i^T (x_1 - x_2)^T \leq c^2 (x_1 - x_2)(x_1 - x_2)^T$$

for all $x_1, x_2 \in \mathbb{R}^2$.

$$\lambda = \frac{(x_1 - x_2)A_i A_i^T (x_1 - x_2)^T}{(x_1 - x_2)(x_1 - x_2)^T} \leq c^2$$

When λ, the maximum eigenvalue of the positive semi-definite matrix $A_i A_i^T$ is less than one, it is known that under this circumstance T_i is a contraction mapping.

In the paper [2], it is shown that there is a unique continuous function $h_2 : [0, 1] \to [0, 1]$ whose graph $\{(x, h_2(x)) : 0 \leq x \leq 1\}$ is the attractor of the IFS $\{T_0, T_1\}$. Furthermore, it is shown in that paper that h_2 may also be characterized as the unique continuous function on $[0, 1]$ such that

$$h_2(i/2^k) = H_2(i, 2^k - i)/2^k, \quad i = 0, 1, \cdots, 2^k; k = 0, 1, \cdots,$$

where

$$H_2(0, 1) = H_2(1, 0) = 0$$

and $H_2(i_1, i_2)$ is defined recursively for each pair (i_1, i_2) of non-negative integers summing to a positive power of two by

$$H_2(i_1, i_2) = H_2(i_1/2, i_2/2), \quad i_1, i_2 \text{ even},$$

$$H_2(i_1, i_2) = 1 + H_2((i_1 - 1)/2, (i_2 + 1)/2) + H_2((i_1 + 1)/2, (i_2 - 1)/2), \quad i_1, i_2 \text{ odd}.$$

We showed earlier in the thesis (in our coverage of bisection trees) that the change of variable (2.6) yields function P_2 satisfying (2.7). This completes the proof.

□

Proof of Theorem 3.1.1. For a fixed n such as $2 \leq n \leq \alpha - 1$, an α-balanced tree ensemble with n leaves consists of one tree. It is a depth-one tree with all n leaves. Thus, the entropy $H_\alpha(n)$ for $2 \leq n \leq \alpha - 1$ is 0. (The case $n = 1$ is excluded because it is trivial.)

For $n \geq \alpha$ and some $k \geq 1$, consider an α-ary random rooted tree $T_k(\alpha)$ which is either an α-balanced tree $T(n)$ having α^k leaves or a tree obtained by removing all edges of maximal depth of $T(n)$ which was mentioned early in Section 1.5. The equation (1.2) can be generalized for an α-ary tree such as

$$Q_n = \{(i_1, i_2, \cdots, i_\alpha)\},$$

where i_j is either $\lceil \frac{n}{\alpha} \rceil$ or $\lfloor \frac{n}{\alpha} \rfloor$ $(j = 1, 2, \cdots, \alpha)$ and $\sum_{j=1}^{\alpha} i_j = n$. Then (1.5) can be generalized as the following.

$$H(n) = h(P_n) = h(p_n) + \sum_{\substack{(i_1, i_2, \cdots, i_\alpha) \\ \in Q_n}} q_n(i_1, i_2, \cdots, i_\alpha) \sum_{j=1}^{\alpha} h(P_{i_j}), \quad n \geq 2. \tag{A.3}$$

For an α-balanced tree ensemble, $\text{card}(Q_n) = \binom{\alpha}{\text{mod}(n,\alpha)}$ and each $q_n(i_1, i_2, \cdots, i_\alpha)$ for $(i_1, i_2, \cdots, i_\alpha) \in Q_n$ is equiprobable, $h(q_n) = \log_2(\text{card}(Q_n))$.Thus (A.3) can be re-written as

$$H(n) = \log_2 \binom{\alpha}{\text{mod}(n, \alpha)} + \sum_{j=1}^{\alpha} H(i_j), \quad n \geq 2.$$

There are $\text{mod}(n, \alpha)$ subtrees having $\lceil \frac{n}{\alpha} \rceil$ leaves and $(\alpha - \text{mod}(n, \alpha))$ subtrees having $\lfloor \frac{n}{\alpha} \rfloor$ leaves rooted at the α children of an α-balanced tree with n leaves. Therefore, (3.1) for $n \geq \alpha$ is obvious. □

Proof of Theorem 3.1.6. Let X be a metric space, ($\mathbb{R}^2$, Euclidean). According to Theorem 2 and the following Corollary in [1], the graph of h_α, $\{(p, h_\alpha(p)) : 0 \leq p \leq 1\}$ is the

attractor of the affine IFS, $T = \{T_0, T_1, \cdots, T_{\alpha-1}\}$. For $i = 0, 1, \cdots, \alpha - 1$, T_i is given as

$$T_i\left(p, h_\alpha(p)\right) = \left(\tilde{T}_i(p), h_\alpha(\tilde{T}_i(p))\right), \quad p \in P, \tag{A.4}$$

where P is a subset of X consisting of all 2-dimensional probability vectors and

$$\tilde{T}_i(p) = \alpha^{-1} p \begin{bmatrix} i+1 & \alpha - i - 1 \\ i & \alpha - i \end{bmatrix}.$$

Let θ denote the invertible function defined in (3.19). If A^* represents C_α, the graphs of P_α and A represents the graph of h_α,

$$A^* = \theta\left(A\right).$$

because of the relation, $P_\alpha = \theta(h_\alpha)$. Also, $T^* = \theta(T)$ by the following reason. The equation obtained by substituting x for $\alpha^t - j$ in (3.6) is equal to the transformed version of (A.4) by the function θ which can be re-written as

$$\theta(x, y) = \left(\log_\alpha(x + j),\ \frac{y}{x + j}\right).$$

Because A is the attractor of the IFS T,

$$A = T(A).$$

By Theorem 5.1 in [11],

$$\begin{aligned} T^*(A^*) &= \left(\theta \circ T \circ \theta^{-1}\right)(A^*) \\ &= \left(\theta \circ T\right)(A) \\ &= \theta(A) \\ &= A^*. \end{aligned}$$

In other words, A^* is the attractor of the IFS T^*, which proves the theorem. □

Proof of Lemma 3.2.3. We have a labeled $T_k(\alpha)$ tree corresponding to each tree in $\mathcal{T}_n(\alpha)$, the ensemble of all α-balanced trees having n leaves. For each of these labeled $T_k(\alpha)$ trees, the following properties hold:

- (p.1): The root label is n.

- (p.2): Each leaf label belongs to $\{j, j+1\}$.
- (p.3): The labels are additive, meaning that the label on each non-leaf vertex is the sum of the labels on its children.
- (p.4): For each depth level $i = 0, 1, \cdots, k$, each vertex label L at depth i will satisfy
$$j\alpha^{k-i} \leq L \leq (j+1)\alpha^{k-i}.$$

For each of the labeled $T_k(\alpha)$ trees, at each depth level $i = 0, 1, \cdots, k$, replace each depth i vertex label L with the new label consisting of the pair of non-negative integers
$$(m_1, m_2) = \left(L - j\alpha^{k-i},\ (j+1)\alpha^{k-i} - L\right).$$

Then, each of the re-labeled $T_k(\alpha)$ trees obeys the following properties, each corresponding to one of the properties (p.1)-(p.4).

- (q.1): The root label is $(n - j\alpha^k, (j+1)\alpha^k - n)$.
- (q.2): Each leaf label belongs to $\{(0,1), (1,0)\}$.
- (q.3): The labels are additive.
- (q.4): The label (m_1, m_2) on each depth i vertex has sum $m_1 + m_2$ of its entries equal to $\alpha^{ki} (i = 0, 1, \cdots, k)$.

The hierarchical type class $\mathcal{S}_n(\alpha)$ results when the new leaf labels on the labeled $T_k(\alpha)$ trees are converted according to $(0,1) \leftrightarrow A$, $(1,0) \leftrightarrow B$. The numbers of As and Bs in each string in hierarchical type class $\mathcal{S}_n(\alpha)$ are thus the numbers of $(0,1)$s and $(1,0)$s, respectively, appearing as new leaf labels in the labeled $T_k(\alpha)$ tree from which the string arises. But, by the additivity property (q.3), these numbers are the entries of $\big(n - j\alpha^k, (j+1)\alpha^k - n\big)$, the root label. □

Proof of Theorem 3.2.5. The function h_α satisfies $h_\alpha(0) = h_\alpha(1) = 0$, from which it follows that
$$P_\alpha(\log_\alpha n) = 0,\ 1 \leq n < \alpha.$$

Therefore, in proving (3.15), we henceforth assume that $n \geq \alpha$. Assigning (k, j) to n as previously (so that $j\alpha^k \leq n < (j+1)\alpha^k$), we have

$$H_\alpha(n)/n = \log_2 \left(\text{card}\left(\mathcal{S}_n(\alpha)\right)\right)/n = (\alpha^k/n) h_\alpha\left(\frac{n - j\alpha^k}{\alpha^k}\right), \tag{A.5}$$

because by the Lemma 3.2.3, the argument of h_α in the preceding equation, when multiplied by α^k, is the number of A s appearing in the strings of $\mathcal{S}_n(\alpha)$. Letting

$$x = (n/\alpha^k - j,$$

we have $0 \leq x \leq 1$, and so $h_\alpha(x)$ is defined. Re-writing (A.5), we have

$$h_\alpha(x) = (x + j)[H + \alpha(n)/n],$$

and by definition of P_α, we have

$$h_\alpha(x) = (x + j) P_\alpha\left(\log_\alpha(x + j)\right).$$

Therefore,

$$H_\alpha(n)/n = P_\alpha\left(\log_\alpha(x + j)\right) = P_\alpha\left(\log_\alpha(n/\alpha^k)\right) = P_\alpha(\log_\alpha n),$$

by periodicity of P_α. □

Proof of Lemma 3.3.1. Suppose we can prove the formulas

$$H(m, \alpha^j - m) = H_\alpha(m + \eta(\alpha^j)), \;\; m = 0, 1, \cdots, \alpha^j;\; \eta = 1, 2, \cdots, \alpha - 1, \tag{A.6}$$

for all $j = 0, 1, 2, \cdots$. Then (3.16) will follow. To see this, note by (3.17) that (3.18) implies

$$h_\alpha(m/\alpha^j) = [(m/\alpha^j) + 1][H_\alpha(m + \eta(\alpha^j))/(m + \alpha^j)],$$
$$m = 0, 1, \cdots, 4^j;\; j = 0, 1, 2, \cdots;\; \eta = 1, 2, \cdots, \alpha - 1.$$

Applying (3.16) with $x = m/\alpha^j$, we then have

$$P\left(\log_\alpha((m/\alpha^j) + \eta)\right) = H_\alpha(m + \eta(\alpha^j))/(m + \alpha^j),$$

which, using the periodicity of P, simplifies to

$$P\left(\log_\alpha(m + \eta(\alpha^j))\right) = H_\alpha(m + \eta(\alpha^j))/(m + \alpha^j), \;\; m = 0, 1, \cdots, 4^j; j = 0, 1, 2, \cdots. \tag{A.7}$$

We can write any integer $n \geq 1$ as being of the form $n = m + \eta(\alpha^j)$ in (A.7). Thus we'd obtain (3.16) for all $n \geq 1$. To complete our proof, we show (A.6). Let us prove the statement

$$H(m, \alpha^j - m) = H_\alpha(m + \eta(\alpha^j)), \;\; m = 0, 1, \cdots, \alpha^j; \; \eta = 1, 2, \cdots, \alpha - 1, \tag{A.8}$$

for all $j \geq 0$. We prove this by induction on j, starting with $j = 0$. For $j = 0$, we'd have to prove

$$H(m, 1 - m) = H_\alpha(m + \eta), \;\; m = 0, 1.$$

This is clearly true (both sides are zero). Fix $J \geq 1$. We want to prove (A.8) for $j = J$, assuming by induction hypothesis that it is true for each j in the range $0 \leq j < J$. Suppose m in the range $0, 1, \cdots, \alpha^J$ is of the form $m = \alpha m_1$. By recursion, we have

$$H(m, \alpha^J - m) = \alpha H(m_1, \alpha^{J-1} - m_1)$$

$$H_\alpha(m + \alpha^J) = \alpha H_\alpha(m_1 + \eta(\alpha^{J-1}))$$

The two right hand sides are equal by the induction hypothesis, and therefore the two left hand sides are equal. Now suppose m in the range $0, 1, \cdots, \alpha^J$ is of the form $m = \alpha m_1 + l$. By recursion,

$$H(m, \alpha^J - m) = (\alpha - l)H(m_1, \alpha^{J-1} - m_1) + lH(m_1 + 1, \alpha^{J-1} - m_1 - 1) + \log_2 \binom{\alpha}{l}$$

$$H_\alpha(m + \eta(\alpha^J)) = (\alpha - l)H_\alpha(\alpha^{J-1} + m_1) + lH_\alpha(\eta(\alpha^{J-1}) + m_1 + 1) + \log_2 \binom{\alpha}{l}.$$

Since m_1 is in the range $0, 1, \cdots, \alpha^{J-1}$, the first terms on the right hand sides of the preceding two equations are equal by the induction hypothesis. Since $m_1 + 1$ is also in this range, the second terms are also equal by induction. We conclude that the left sides are equal. Our proof of (A.8) is now complete. □

Proof of Lemma 3.3.2. Suppose we can prove the formulas

$$H(m, 3^j - m) = H_3(m + 3^j), \;\; m = 0, 1, \cdots, 3^j; j = 0, 1, 2, \cdots \tag{A.9}$$

$$H(m, 3^j - m) = H_3(m + 2(3^j)), \;\; m = 0, 1, \cdots, 3^j; j = 0, 1, 2, \cdots. \tag{A.10}$$

Then (3.24) will follow. To see this, note by (3.21) that (A.9) implies

$$h_3(m/3^j) = [(m/3^j) + 1][H_3(m + 3^j)/(m + 3^j)], \;\; m = 0, 1, \cdots, 3^j; j = 0, 1, 2, \cdots.$$

Applying (3.22) with $x = m/3^j$, we then have

$$P(\log_3((m/3^j)+1)) = H_3(m+3^j)/(m+3^j),$$

which, using the periodicity of P, simplifies to

$$P(\log_3(m+3^j)) = H_3(m+3^j)/(m+3^j), \;\; m = 0, 1, \cdots, 3^j; j = 0, 1, 2, \cdots \tag{A.11}$$

Similarly, from (A.10) and (3.23) we'd obtain

$$P(\log_3(m+2(3^j))) = H_3(m+2(3^j))/(m+2(3^j)), \;\; m = 0, 1, \cdots, 3^j; j = 0, 1, 2, \cdots \tag{A.12}$$

We can write any integer $n \geq 1$ as being of the form $n = m + 3^j$ in (A.11) or of the form $n = m + 2(3^j)$ in (A.12). Thus we'd obtain (3.24) for all $n \geq 1$. To complete our proof, we show (A.9)-(A.10). Let us prove the statement

$$H(m, 3^j - m) = H_3(m+3^j), \;\; m = 0, 1, \cdots, 3^j \tag{A.13}$$

for all $j \geq 0$. We prove this by induction on j, starting with $j = 0$. For $j = 0$, we'd have to prove

$$H(m, 1-m) = H_3(m+1), \;\; m = 0, 1.$$

This is clearly true (both sides are zero). Fix $J \geq 1$. We want to prove (A.13) for $j = J$, assuming by induction hypothesis that it is true for each j in the range $0 \leq j < J$. Suppose m in the range $0, 1, \cdots, 3^J$ is of the form $m = 3m_1$. By recursion, we have

$$H(m, 3^J - m) = 3H(m_1, 3^{J-1} - m_1)$$

$$H_3(m+3^J) = 3H_3(m_1 + 3^{J-1})$$

The two right hand sides are equal by the induction hypothesis, and therefore the two left hand sides are equal. Now suppose m in the range $0, 1, \cdots, 3^J$ is of the form $m = 3m_1 + 1$. By recursion,

$$H(m, 3^J - m) = 2H(m_1, 3^{J-1} - m_1) + H(m_1 + 1, 3^{J-1} - m_1 - 1) + \log_2 3$$

$$H_3(m+3^J) = 2H_3(3^{J-1} + m_1) + H_3(3^{J-1} + m_1 + 1) + \log_2 3$$

Since m_1 is in the range $0, 1, \cdots, 3^{J-1}$, the first terms on the right hand sides of the preceding two equations are equal by the induction hypothesis. Since $m_1 + 1$ is also in this range, the second terms are also equal by induction. We conclude that the left sides are equal. Finally, suppose m in the range $0, 1, \cdots, 3^J$ is of the form $m = 3m_1 + 2$. By recursion

$$H(m, 3^J - m) = 2H(m_1 + 1, 3^{J-1} - m_1 - 1) + H(m_1, 3^{J-1} - m_1) + \log_2 3$$

$$H_3(m + 3^J) = 2H_3(3^{J-1} + m_1 + 1) + H_3(3^{J-1} + m_1) + \log_2 3$$

By induction, the two right hand sides are equal, and therefore the two left hand sides are equal. Our proof of (A.13) is now complete. Our final task is to prove the statement

$$H(m, 3^j - m) = H_3(m + 2(3^j)), \quad m = 0, 1, \cdots, 3^j \tag{A.14}$$

for all $j \geq 0$. We prove this by induction on j, starting with $j = 0$. For $j = 0$, the statement becomes

$$H(m, 1 - m) = H_3(m + 2), \quad m = 0, 1,$$

which is clearly true. Fix $J \geq 1$. We want to prove (A.14) for $j = J$, assuming by induction hypothesis that it is true for each j in the range $0 \leq j < J$. Suppose m in the range $0, 1, \cdots, 3^J$ is of the form $m = 3m_1$. By recursion, we have

$$H(m, 3^J - m) = 3H(m_1, 3^{J-1} - m_1),$$

$$H_3(m + 2(3^J)) = 3H_3(m_1 + 2(3^{J-1})).$$

The two right hand sides are equal by induction and therefore the two left hand sides are equal. Now suppose m in the range $0, 1, \cdots, 3^J$ is of the form $m = 3m_1 + 1$. By recursion,

$$H(m, 3^J - m) = 2H(m_1, 3^{J-1} - m_1) + H(m_1 + 1, 3^{J-1} - m_1 - 1) + \log_2 3$$

$$H_3(m + 2(3^J)) = 2H_3(2(3^{J-1}) + m_1) + H_3(2(3^{J-1}) + m_1 + 1) + \log_2 3$$

The two right hand sides are equal by induction and therefore the two left hand sides are equal. Finally, suppose m in the range $0, 1, \cdots, 3^J$ is of the form $m = 3m_1 + 2$. By recursion

$$H(m, 3^J - m) = 2H(m_1 + 1, 3^{J-1} - m_1 - 1) + H(m_1, 3^{J-1} - m_1) + \log_2 3$$

$$H_3(m + 2(3^J)) = 2H_3(2(3^{J-1}) + m_1 + 1) + H_3(2(3^{J-1}) + m_1) + \log_2 3$$

By induction, the two right hand sides are equal, and therefore the two left hand sides are equal. Our proof of (A.14) is now complete. □

Proof of Theorem 5.3.1. Let h_2 be the hierarchical entropy function defined in [2]. According to Example 11.4 of [10], the graph of h_2 has box dimension 1. The function h_2 is related to the function P_2 by (2.6) which is re-written as

$$P_2(t) = 2^{-t} h_2(2^t - 1),\ 0 \le t \le 1. \tag{A.15}$$

The statement of Theorem is obtained by successive application of Lemma 5.3.2 and Lemma 5.3.3 to (A.15). □

Proof of Lemma 5.3.2. Let M be a positive integer satisfying $M > 1/\epsilon$. For each positive integer m, let $m' = Mm$. Also, let $\delta_m = 1/m$ and $\delta_{m'} = 1/m'$. Let $\mathcal{P}_m$ be the set of intervals

$$\mathcal{P}_m = \{[i\delta_m, (i+1)\delta_m] : i = 0, 1, \cdots, m-1\}$$

and let $\mathcal{P}_{m'}$ be the set of intervals

$$\mathcal{P}_{m'} = \{[i\delta_{m'}, (i+1)\delta_{m'}] : i = 0, 1, \cdots, m'-1\}.$$

Consider the following $m+1$ points which we obtain by applying the function ϕ to the endpoints of intervals in $\mathcal{P}(m)$:

$$q_0 = 0 = \phi(0) < q_1 = \phi(\delta_m) < q_2 = \phi(2\delta_m) < \cdots < q_{m-2} = \phi((m-1)\delta_m) < q_m = \phi(1).$$

Pick intervals

$$I_0, I_1, \cdots, I_m \tag{A.16}$$

belonging to $\mathcal{P}_{m'}$ such that q_j belongs to I_j for each j. We show that these intervals are distinct. If not, suppose $I = I_{j_1} = I_{j_2}$, for $j_1 \ne j_2$. on the one hand, we have

$$|q_{j_1} - q_{j_2}| \le 1/m',$$

since the $\mathcal{P}_{m'}$ intervals are of width $1/m'$. On the other hand, we have

$$|q_{j_1} - q_{j_2}| = |\phi(j_1\delta_m) - \phi(j_2\delta_m)| \ge \epsilon|(j_1\delta_m) - (j_2\delta_m)| \ge \epsilon/m,$$

but $\epsilon/m \leq 1/m'$ is impossible by choice of M. Using the notation on pages 146-147 of [10], if I in an interval in $\mathcal{P}_m$, we have

$$R_g[I] = \sup\{|g(x) - g(y)| : x, y \in I\},$$

and if J is an interval in $\mathcal{P}_{m'}$, we have

$$R_f[I] = \sup\{|f(u) - f(v)| : u, v \in J\}.$$

For each I in $\mathcal{P}_m$, let $\mathcal{J}(I)$ be the list of intervals from $\mathcal{P}_{m'}$, such that the first and last members of the list are the two intervals in (A.16) containing the left and right endpoints of $\phi(I)$, respectively, and the intermediate members of the list $\mathcal{J}(I)$ are the intervals between these first and last members in the left-to-right list of intervals in $\mathcal{P}_{m'}$. It is not hard to see that

$$R_g[I] \leq \sum_{J \in \mathcal{J}} R_f[J].$$

Summing over I in $\mathcal{P}_{m'}$, we then have

$$\sum_I R_g[I] \leq 2 \sum_J R_f[J],$$

where from now on, $\sum_I$ denotes that we sum over all intervals I in $\mathcal{P}_m$, and $\sum_J$ denotes that we sum over all intervals J in $\mathcal{P}_{m'}$. We then have

$$\begin{aligned} N_{\delta_m}[g] &\leq 2\delta_m^{-1} + \delta_m^{-1} \sum_I R_g[I] \\ & \delta_m^{-1} + (2/M)\delta_{m'}^{-1} \sum_J R_f[J] \\ & \delta_m^{-1} + (2/M) N_{\delta_{m'}}[f], \end{aligned}$$

where the first and last inequality in the preceding employ formula (11.1) on page 147 of [10], first for the function g and then for the function f. This gives us

$$N_{\delta_m}[g] \leq 2\delta_m^{-1} + (2/M) N_{\delta_{m'}}[f],$$

valid for all m. It follows that

$$\limsup_{m \to \infty} \frac{\log \delta_m[g]}{-\log \delta_m} \leq \max(1, dim_B(graph[f])) = 1.$$

Given that

$$dim_B(graph[f]) = 1 + \lim_{\delta \to 0} \frac{\log \sum_{i=0}^{m-1} R_f[i\delta, i\delta + \delta]}{\log \delta^{-1}} = 1,$$

there exists $\delta_0 > 0$ such that for all $\delta < \delta_0$,

$$\frac{\log \sum_{i=0}^{m-1} R_f[i\delta, i\delta + \delta]}{\log \delta^{-1}} = 0,$$

where

$$R_f[t_1, t_2] = \sup_{t_1 \le t, u \le t_2} |f(t) - f(u)|.$$

Hence

$$\sum_{i=0}^{m-1} R_f[i\delta, i\delta + \delta] = c_0, \quad c_0 \in \mathbb{R},$$

which is

$$\sum_{i=0}^{m-1} \sup_{i\delta \le t_1, t_2 \le i\delta + \delta} |f(t_1) - f(t_2)| = c_0.$$

Without loss of generality, let us asume that $t_1 > t_2$.

$$t_1 - t_2 = \triangle \le \delta, \quad \triangle > 0$$

$$\sup_{i\delta \le t_1, t_2 \le i\delta + \delta} |f(t_2 + \triangle) - f(t_2)| = c_1\delta, \quad \text{for some } c_1 \in \mathbb{R}.$$

Because ϕ is a strictly increasing function, for some $c_2 > 0$,

$$\phi(t_2 + \triangle) - \phi(t_2) \ge c_2 \triangle .$$

$$\begin{aligned}
R_g[i\delta, i\delta + \delta] &= \sup_{i\delta \le t_2 \le i\delta + \delta} |g(t_2 + \triangle) - g(t_2)| \\
&= \sup_{i\delta \le t_1 \le i\delta + \delta} |f(\phi(t_2 + \triangle)) - f(\phi(t_2))| \\
&\ge \sup_{i\delta \le t_1 \le i\delta + \delta} |f(\phi(t_2) + c_2 \triangle) - f(\phi(t_2))| \\
&= c\delta, \quad \text{for some } c \in \mathbb{R} \\
\therefore dim_B(graph[g]) &\ge \lim_{\delta \to 0} \frac{\log \delta^{-1} \sum_{i=0}^{m-1} R_g[i\delta, i\delta + \delta]}{\log \delta^{-1}} \\
&\ge \lim_{\delta \to 0} \frac{\log c\delta^{-1}}{\log \delta^{-1}} \\
&= 1.
\end{aligned}$$

Thus,

$$dim_B(graph[g]) = 1.$$

□

Proof of Lemma 5.3.3. There exists a real constant $c_1 \geq 0$ such that

$$|\phi(t_1) - \phi(t_2)| \leq c_1|t_1 - t_2|.$$

For $0 \leq t_1, t_2 \leq 1$,

$$\begin{aligned}
|g(t_1) - g(t_2)| &= |\phi(t_1)f(t_1) - \phi(t_2)f(t_2)| \\
&= |\phi(t_1)f(t_1) - \phi(t_1)f(t_2) + \phi(t_1)f(t_2) - \phi(t_2)f(t_2)| \\
&\leq |\phi(t_1)||f(t_1) - f(t_2)| + |f(t_2)||\phi(t_1) - \phi(t_2)| \\
&\leq |\phi(t_1)||f(t_1) - f(t_2)| + c_1|f(t_2)||t_1 - t_2|.
\end{aligned}$$

For any subinterval I of $[0, 1]$,

$$\begin{aligned}
R_g[I] &= \sup_{0\leq t_1,t_2\leq 1} |g(t_1) - g(t_2)| \\
&\leq \sup_{0\leq t_1,t_2\leq 1} |\phi(t_1)||f(t_1) - f(t_2)| + c_1|f(t_2)||t_1 - t_2| \\
&\leq b_1 R_f[I] + b_2\delta. \qquad (b_1 = |\phi(t_1)|,\ \ b_2 = c_1|f(t_2)|)
\end{aligned}$$

Using the second inequality in (A.18),

$$\begin{aligned}
N_\delta(g) &\leq 2m + b_1 \left(\delta^{-1} \sum_{i=0}^{m-1} R_f[i\delta, i\delta + \delta] \right) + b_2 m \\
&\leq (2 + b_2)m + b_1 N_\delta(f) \\
&< (2 + b_2)(1 + \delta^{-1}) + b_1 N_\delta(f).
\end{aligned}$$

Thus,

$$\begin{aligned}
dim_B(g) &= \lim_{\delta\to 0} \tfrac{\ln N_\delta(g)}{\ln \delta^{-1}} \\
&\leq \lim_{\delta\to 0} \tfrac{\ln\{(2+b_2)(1+\delta^{-1})+b_1 N_\delta(f)\}}{\ln \delta^{-1}} \\
&= 1.
\end{aligned}$$

Therefore,

$$dim_B(g) \leq 1.$$

On the other hand, the lower bound of $dim_B(g)$ is as follows.

$$\begin{aligned}
|g(t_1) - g(t_2)| &= |\phi(t_1)f(t_1) - \phi(t_1)f(t_2) + \phi(t_1)f(t_2) - \phi(t_2)f(t_2)| \\
&\geq |\phi(t_1)|\,|f(t_1) - f(t_2)| - |f(t_2)|\,|\phi(t_1) - \phi(t_2)|
\end{aligned}$$

$$\begin{aligned}
R_g[I] &= \sup_{0\leq t_1,t_2\leq 1} |g(t_1) - g(t_2)| \\
&\geq \sup_{0\leq t_1,t_2\leq 1} \{b_1|f(t_1) - f(t_2)| - b_2c_1|t_1 - t_2|\} \qquad (b_1 = |\phi(t_1)|,\ \ b_2 = |f(t_2)|) \\
&\geq b_1R_f[I] - b_2c_1I
\end{aligned}$$

Using the first inequality in (A.18),

$$\begin{aligned}
N_\delta(g) &\geq \delta^{-1} \sum_{i=0}^{m-1} R_g[i\delta, (i+1)\delta] \\
&\geq \delta^{-1} \sum_{i=0}^{m-1} \{b_1R_f[i\delta, (i+1)\delta] - b_2c_1\delta\} \\
&\geq b_1\left(N_\delta(f) - 2m\right) - b_2c_1m
\end{aligned}$$

$$\begin{aligned}
dim_B(g) &\geq \lim_{\delta\to 0} \tfrac{\ln\{b_1N_\delta(f) - bm\}}{\ln \delta^{-1}} \qquad (b = 2b_1 - b_2c_1) \\
&\geq \lim_{\delta\to 0} \tfrac{\ln\{b_1N_\delta(f) - b(\delta^{-1}+1)\}}{\ln \delta^{-1}} \\
&= 1
\end{aligned}$$

$$\therefore dim_B(g) = 1.$$

□

Proof of Theorem 5.3.4. Let h_α be the hierarchical entropy function defined in [2]. The relation of functions P_α and h_α is given in (3.14), and it is re-written in terms of h_α as the following equation.

$$P_\alpha(t) = \alpha^{-t}h_\alpha(\alpha^t - j), \quad \log_\alpha j \leq t \leq \log_\alpha(j+1),\ j = 1, \cdots, \alpha - 1. \tag{A.17}$$

For each j, the $P_\alpha(t)$ is defined in the interval $[\log_\alpha j,\ \log_\alpha(j+1)]$, which is one of the pieces of the C_α curve, $C_\alpha(j)$. The j-th piece $C_\alpha(j)$ is a function of the h_α and it satisfies the conditions of Lemma 5.3.2 and Lemma 5.3.3. The graph of h_α is of box dimension of 1 according to Example 11.4 of [10] and the dimensions of the graph of h_α and the $C_\alpha(j)$ are the same by the Lemma 5.3.2 and Lemma 5.3.3. Because the curve

C_α is a union of disjoint Borel sets, $C_\alpha(j)$'s, the box dimension is preserved, which is 1. □

Proposition 11.1 of [10]. Let $f : [0,1] \to R$ be continuous. Suppose that $0 < \delta < 1$, and $m = \lfloor \frac{1}{\delta} \rfloor$. Then, if N_δ is the number of squares of the δ-mesh that intersect graph f,

$$\delta^{-1} \sum_{i=0}^{m-1} R_f[i\delta, (i+1)\delta] \leq N_\delta \leq 2m + \delta^{-1} \sum_{i=0}^{m-1} R_f[i\delta, (i+1)\delta] \tag{A.18}$$

,where

$$R_f[t_1, t_2] = \sup_{t_1 \leq t, u \leq t_2} |f(t) - f(u)|.$$