Entropy and Optimal Compression of Some General Plane Trees

ZBIGNIEW GOŁĘBIEWSKI, Wrocław University of Science and Technology
ABRAM MAGNER, University of Illinois at Urbana-Champaign
WOJCIECH SZPANKOWSKI, Purdue University

We continue developing the information theory of structured data. In this paper, we take up d-ary trees ($d \geq 2$) and trees with unrestricted degree. We first compute the entropy which gives us the fundamental lower bound on compression of such trees. Then we present efficient compression algorithms based on arithmetic encoding that achieve the entropy within a constant number of bits. A naïve implementation of these algorithms has a prohibitive time complexity of $O(n^d)$ elementary operations, but our efficient algorithms run in $O(n^2)$ of these operations, where n is the number of nodes. It turns out that extending source coding (i.e., compression) from sequences to advanced data structures such as general trees is mathematically quite challenging and leads to new recurrences that find ample applications in the information theory of general structures (e.g., to analyze the information content of general non-plane trees).

ACM Reference format:
Zbigniew Gołębiewski, Abram Magner, and Wojciech Szpankowski. 2016. Entropy and Optimal Compression of Some General Plane Trees. 1, 1, Article 1 (January 2016), 22 pages.
DOI: 10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Advances in sensing, communication, and storage technologies have created a state of the art in which our ability to collect data from richly instrumented environments has far outpaced our ability to process, understand, and analyze this data in a (provably) rigorous manner. A significant component of this complexity arises from the highly structured nature of data. This poses significant challenges for theoretical characterization of limits of information storage, content, and transmission and methods that achieve these limits. While heuristic approaches are often currently used, critical issues regarding their performance remain. These challenges have motivated our recent research program [6, 7, 15] and others [2, 12, 20], which aims to characterize limits on information content of non-sequential data, such as trees and graphs. Quite generally, these problems consist of two complementary parts: fundamental achievable limits on data compression (where the Shannon entropy of the probability distribution from which the data arises gives the minimum possible expected code length), and, conversely, efficient encoding/decoding algorithms that achieve those limits.

As a start to understanding structured data in an information-theoretic setting, we focused on graphs [6] and trees with vertex (correlated) names [15]. In 1990, Naor proposed an efficiently computable representation for unlabeled graphs (solving Turán's open question) that is optimal up to the first two leading terms of the entropy when all unlabeled graphs are equally likely. Naor's result is asymptotically a special case of recent work of Choi and Szpankowski [6], who extended

Naor's result to general Erdős-Rényi graphs. In particular, in [6] the entropy and an optimal compression algorithm (up to two leading terms of the entropy) for Erdős-Rényi graph structures were presented. Recently, these results were extended to preferential attachment graphs in [14] and to uniform random intersection graphs in [11]. Furthermore, in [16] an automata approach was used to design an optimal graph compression scheme. For binary plane-oriented trees rigorous information-theoretic results were obtained in [12], complemented by a universal grammar-based lossless coding scheme [20]. Finally, in recent work [15] (see also [7]) the authors study binary trees with structure-correlated vertex names, as well as non-plane binary trees, and design optimal compression schemes based on arithmetic encoding.

In this paper, we study general plane-oriented d-ary trees without correlated vertex names (i.e., trees with degree $d \geq 2$) and general trees without any restriction on vertex degrees, giving both precise entropy calculations and information theoretically optimal and efficient compression algorithms. It turns out that moving from binary trees to d-ary (general) trees is mathematically quite challenging. In [15] for binary trees an equivalence was proved between two models: the *binary search tree model* and a model in which leaves are selected randomly to expand the tree by adding two additional nodes (new leaves). This equivalence allowed to analyze the entropy of such trees by solving a relatively simple recurrence, and then to construct an optimal compression algorithm. Unfortunately, this equivalence does not work for $d \geq 3$, and we develop here a different methodology to overcome this problem. Furthermore, there are various ways to generalize binary tree models to larger degrees, and we discuss here three different models.

In this paper, we shall show that for d-ary trees T_n on n internal nodes the entropy $H(T_n)$ satisfies

$$H(T_n) = H(\text{root}) + d \sum_{k=0}^{n-1} H(T_k) p_{n,k}$$

where $H(\text{root})$ is the entropy of the split probability distribution at the root (that is, the distribution of the d-tuple of root subtree sizes), and $p_{n,k}$ is the probability of one specified subtree being of size k. We consider three models. For the m-ary *search tree model* discussed in Section 2, this recurrence can be handled by results from [5, 9]. Then we analyze a more interesting (and possibly more relevant) d-ary tree model in which we randomly select a leaf and add exactly d leaves to it as children. In [15] $d = 2$ was studied, but the analysis is more complicated when $d > 2$. For example, for $d = 2$ we have $p_{n,k} = 1/n$ while for $d = 3$ we can prove that

$$p_{n,k} = \frac{1}{2n} \frac{\binom{2k}{k} 2^{2n}}{\binom{2n}{n} 2^{2k}}.$$

Note the dependence on k. We complete our analysis by studying general trees in which there is no restriction on the node degree.

In Section 3 we show that for d-ary trees we need to analyze a new type of recurrence of the following form (see Lemma 3.6):

$$x_n = a_n + \frac{\alpha}{n} \frac{n!}{\Gamma(n+\alpha-1)} \sum_{k=0}^{n-1} \frac{\Gamma(k+\alpha-1)}{k!} x_k \tag{1}$$

where $\alpha = d/(d-1)$, a_n is a given sequence, and Γ is the Euler gamma function. This is a recurrence with full history and is amenable to what is called the *method of differences*, by which such a recurrence can be converted to a linear one of the form $x_n = A_n x_{n-1} + B_n$, for some sequences A_n, B_n. In the case of $d \geq 3$, this step is rather complicated and, furthermore, the specific form of A_n and B_n leads to a "closed-form" solution which makes subsequent asymptotic analysis nontrivial (this is reflected in the forms of the singularities of the respective generating functions of $\{x_n\}$

for different values of d: when $d = 2$, the generating function is meromorphic, while for $d \geq 3$, algebraic singularities are introduced). The situation is even more involved when we consider general trees in Section 3.3 where no restrictions on degrees are imposed.

After describing in Section 2 three possible generalizations of the binary tree model from [15], we present our main results in Section 3. We first provide in Corollary 3.2 the entropy rate for m-ary search trees. Then we consider d-ary increasing trees and in Theorem 3.7 give our expression for the entropy of such trees. We extend it to general increasing trees in Theorem 3.11. After establishing these fundamental lower bounds on tree compression, we provide efficient compression algorithms, which are optimal in expected code length to within a constant number of bits.

2 MODELS

In this section we describe the concepts of unlabeled plane trees with and without restrictions on the nodes' out-degrees. This will allow us to introduce three models of tree generation.

We call a rooted tree a plane tree when we distinguish left-to-right order of the successors of each node; i.e., we distinguish all different embeddings of a tree into the plane (see [8]). General unlabeled plane trees are rooted plane trees with no restriction on the number of children of the nodes. On the other hand, unlabeled d-ary plane trees are rooted plane trees where each internal (i.e., non-leaf) node has exactly d children (either internal of external). Since they are unlabeled, they can be seen as objects that encode the structures of a possible labeled (in the graph-theoretic sense) plane tree. We define the size of the unlabeled plane tree and also unlabeled d-ary plane tree by the number of internal nodes.

2.1 Unlabeled m-Ary Search Tree Model

Search trees are plane trees built from a set of n distinct keys taken from some totally ordered set, for instance a random permutation of the numbers $\{1, 2, \dots, n\}$. An m-ary *search tree* is a tree in which each node has at most m children; moreover, each node stores up to $m - 1$ keys. We define the size of a search tree as the number of keys n. The construction of m-ary search trees can be described as follows [8].

If $n = 0$ the tree is empty. If $1 \leq n \leq m - 1$ the tree consists of a root only, with all keys stored in the root. If $n \geq m$ we select $m - 1$ keys that are called pivots. The pivots are stored in the root. The $m - 1$ pivots split the set of remaining $n - m + 1$ keys into m sublists $I_1, \dots, I_m$: if the pivots are $p_1 < p_2 < \cdots < p_{m-1}$, then $I_1 := (p_i : p_i < p_1), I_2 := (p_i : p_1 < p_i < p_2), \dots, I_m := (p_i : p_{m-1} < p_i)$. We then recursively construct a search tree for each of the sets I_i of keys. In order to obtain an unlabeled search tree of size n we remove the keys from a search tree of size n (see Fig. 1).

The standard probability model assumes that every permutation of the keys $\{1, \dots, n\}$ is equally likely. The choice of pivots can then be deterministic. For example, one always chooses the first $m - 1$ keys. After removing node labels from a m-ary search tree generated by a uniformly random permutation, we obtain a random unlabeled search tree, where the probability distribution of the resulting tree on the set of unlabeled search trees is non-uniform.

2.2 Unlabeled d-ary Plane Increasing Tree Model

We consider the following generation model of an unlabeled random d-ary plane increasing tree [3, 8]. The process starts with an empty tree, that is with just an external node (leaf). The first step in the growth process is to replace this external node by an internal one with d successors that are external nodes (see Figure 2). Then with probability $\frac{1}{d}$ each, one of these d external nodes is selected and again replaced by an internal node with d successors. In each subsequent step one of

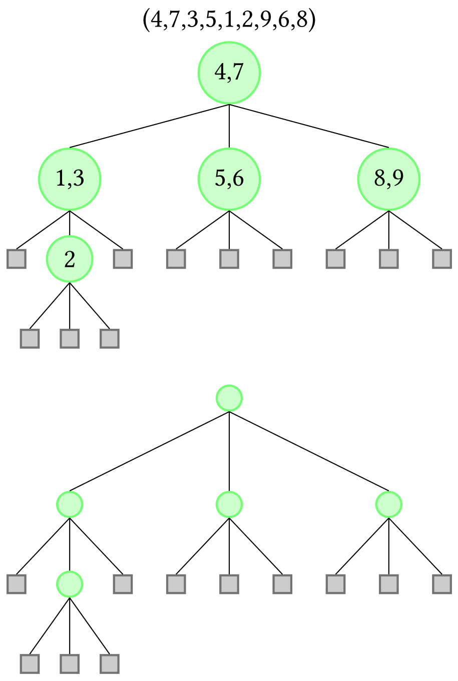


Fig. 1. Example of a 3-ary search tree of size 9 built from the permutation $(4, 7, 3, 5, 1, 2, 9, 6, 8)$ and its unlabeled counterpart.

the external nodes is replaced (with equal probability) by an internal node with d successors. At the end, we remove the labels from internal nodes of the tree.[1]

It is known that this evolution process (without removing the labels at the end) produces random d-ary *plane increasing trees* (see [3, 8]). By definition, d-ary plane increasing trees are rooted plane trees with labels on internal nodes and where each node has exactly d successors (either internal or external). The root is labeled by 1 and the labels of all internal successors of any node v are larger than the label of v. We define the size of the d-ary plane increasing tree by the number of internal nodes.

Let $\mathcal{F}$ denote the set of all unlabeled d-ary plane trees and, for each integer $n \geq 0$, let $\mathcal{F}_n$ be the subset of $\mathcal{F}$ consisting of all trees that contains exactly n internal nodes. Let F_n be a random variable supported on $\mathcal{F}_n$, constructed by the generation process just described.

Similarly, let $\mathcal{G}$ and $\mathcal{G}_n$ be the analogous sets of labeled d-ary plane increasing trees. Throughout the paper, we will denote a given unlabeled d-ary plane tree of size n as $\mathtt{f}_n$ and a given labeled d-ary plane increasing tree of size n as $\mathtt{g}_n$.

Observe that the above described evolution process (choosing an external node to replace with uniform distribution among all external nodes) generates every d-ary plane increasing tree of size n in a unique way and with uniform distribution. However, after removing labels from a d-ary plane increasing tree $\mathtt{g}_n$ we obtain an unlabeled d-ary plane tree $\mathtt{f}_n$ with a non-uniform distribution. For example, there are $\binom{3}{2,0,1}\binom{1}{0,1,0} = 3$ ways to obtain the resulting tree from Figure 2, but there is only

[1]Observe that labels describe the history of the evolution process.

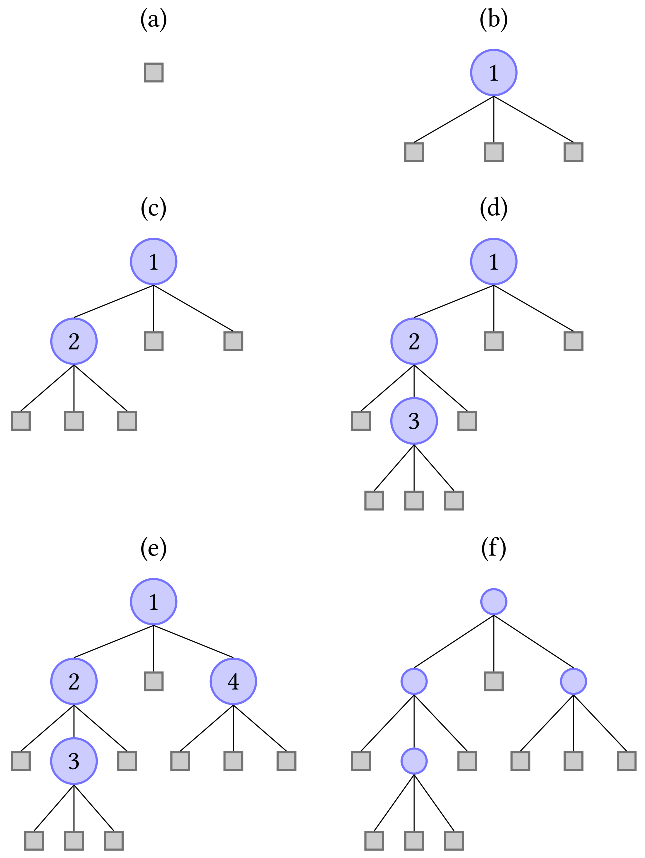


Fig. 2. Example of the generation process that produces 3-ary plane tree of size 4 and its unlabeled counterpart.

one way (exactly $\binom{3}{3,0,0}\binom{2}{2,0,0}\binom{1}{1,0,0} = 1$) to obtain a tree where every internal node is a leftmost child of a parent node.

2.3 Unlabeled General Plane Increasing Tree Model

We consider the following generation model of unlabeled plane trees. Suppose that the process starts with the root node carrying a label 1. Then we add a child node with label 2 to the root. The next step is to attach a node with label 3. However, there are three possibilities: either to add it to the root (as a left or right sibling of 2) or to the node with label 2. One proceeds further in the same way, as shown in Figure 3. At the end, we remove the labels from internal nodes of the tree. Observe that if a node already has out-degree k (where the descendants are ordered), then there are $k + 1$ possible ways to add a new node (this time we do not distinguish between external and internal nodes). Hence, if a plane tree already has $j - 1$ nodes then there are precisely $2j - 3$ possibilities to attach the jth node (see Figure 3). More precisely, the probability of choosing a node of out-degree k equals $(k + 1)/(2j - 3)$.

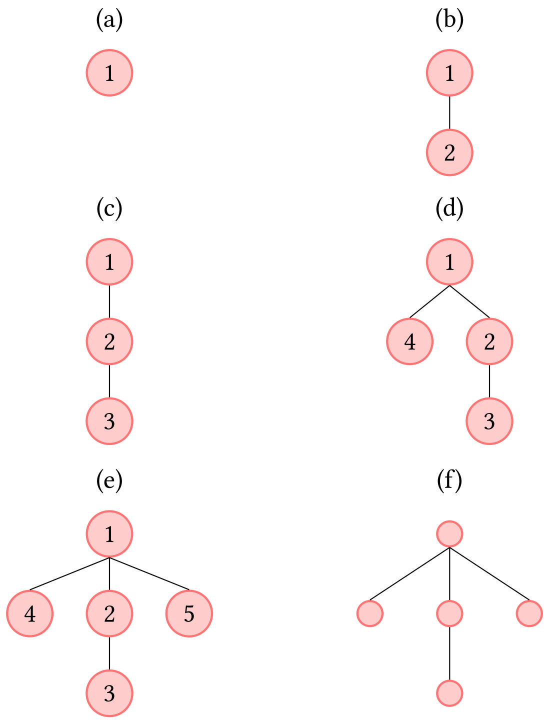


Fig. 3. Example of the generation process that produces a labeled general plane tree of size 5 and its unlabeled counterpart.

3 MAIN RESULTS

In this section we present our main results. In particular, we briefly address the entropy of m-ary search trees. Then we present our derivation of the recurrence for the entropy of unlabeled d-ary plane increasing trees that leads us to a formula for the entropy rate. Finally, we derive the entropy rate for general trees. In the next section, we give efficient and nearly optimal compression algorithms, which achieve an expected code length within a small constant number of bits of the entropies derived here.

In all our models, the probability distribution on the relevant set of trees is non-uniform, and subtrees of the root are conditionally independent given their respective sizes. Indeed, let T_n be a random variable representing a tree $\mathtt{t}_n$ on n internal nodes. Assume now that at the root we split $\mathtt{t}_n$ into d subtrees of size $k_1, \ldots, k_d$, respectively, where $k_1 + \cdots + k_d = n - 1$. Then the probability $\mathbb{P}(T_n = \mathtt{t}_n)$ of generating tree $\mathtt{t}_n$ in all our models satisfies

$$\mathbb{P}(T_n = \mathtt{t}_n) = \mathbb{P}(k_1, \ldots, k_d) \prod_{i=1}^{d} \mathbb{P}\left(T_{k_i} = \mathtt{t}_{k_i}\right) \tag{2}$$

where $\mathbb{P}(k_1, \ldots, k_d)$ is the probability of a split at the root of n internal nodes into subtrees of sizes $k_1, \ldots, k_d$, respectively. This split probability is different for m-ary search trees, d-ary trees, and general trees, as we shall see in this section.

We shall use the following notation. Let $\mathbf{k}^{(d)} = (k_1, \ldots, k_d)$ denote an d-dimensional vector of non-negative integers and $\|\mathbf{k}^{(d)}\| = |k_1| + \ldots + |k_d|$ be its L^1 norm. Let $(k, \mathbf{k}^{(d-1)}) = (k, k_2 \ldots, k_d)$

denote a d-dimensional vector with the first coordinate equal to k. We often write $\mathbf{k}$ for $\mathbf{k}^{(d)}$ when the vector dimension is obvious.

3.1 The Entropy of the Unlabeled m-ary Search Trees

Let U_n denote a random unlabeled m-ary search tree with n keys, generated according to the process described earlier. We write u_n for an arbitrary fixed m-ary (unlabeled) search tree with n keys.

We describe the splitting of keys at the root of the search tree by the random vector $\mathbf{Y}_n^{(m)} = (Y_{n,1}, \dots, Y_{n,m})$, where $Y_{n,j} = |I_j|$ is the number of keys that go into the jth subtree of the root. If $n \geq m-1$ we have $Y_{n,1} + \cdots Y_{n,m} = n - m + 1$ and

$$\mathbb{P}\left(\mathbf{Y}_n^{(m)} = \mathbf{k}^{(m)}\right) = \frac{1}{\binom{n}{m-1}}. \tag{3}$$

Notice that $\mathbb{P}\left(\mathbf{Y}_n^{(m)} = \mathbf{k}^{(m)}\right)$ does not depend on the vector $\mathbf{k}^{(m)}$ coordinates $k_1, \dots, k_m$, which simplifies calculations.

Suppose that the tree u_n has subtrees $\mathsf{u}_{k_1}, \dots, \mathsf{u}_{k_m}$ of sizes $k_1, \dots, k_m$. Then by (2)

$$\mathbb{P}\left(U_n = \mathsf{u}_n\right) = \mathbb{P}\left(\mathbf{Y}_n^{(m)} = \mathbf{k}^{(m)}\right) \prod_{j=1}^{m} \mathbb{P}\left(U_{k_j} = \mathsf{u}_{k_j}\right). \tag{4}$$

Let us establish the initial conditions of the entropy of m-ary search trees. If $n = 0$ we have an empty tree, and $H(U_0) = 0$. Moreover, if $1 \leq n \leq m-1$, all keys are stored in one node, and $H(U_n) = 0$. For $n > m-1$ there is a bijection between a tree U_n and a tuple $(\mathbf{Y}_n^{(m)}, U_{Y_{n,1}}, \dots, U_{Y_{n,m}})$, which is an immediate consequence of (4). Therefore, for $n > m-1$, we have

$$\begin{aligned} H(U_n) &= H\left(\mathbf{Y}_n^{(m)}, U_{Y_{n,1}}, \dots, U_{Y_{n,m}}\right) = H\left(\mathbf{Y}_n^{(m)}\right) + H\left(U_{Y_{n,1}}, \dots, U_{Y_{n,m}} | \mathbf{Y}_n^{(m)}\right) \\ &= H\left(\mathbf{Y}_n^{(m)}\right) + m \sum_{k=0}^{n-m+1} H(U_k)\, \mathbb{P}\left(Y_{n,1} = k\right), \end{aligned}$$

where the second equality is by conditional independence of the root subtrees given their sizes, where two subtrees having the same size have the same distribution.

For $n \geq m-1$ and $1 \leq j \leq m$, the random variables $Y_{n,j}$ are identically distributed, and for $0 \leq k \leq n-1$,

$$\mathbb{P}\left(Y_{n,1} = k\right) = \sum_{\|\mathbf{k}^{(m-1)}\| = n-m+1-k} \mathbb{P}\left(\mathbf{Y}_n^{(m)} = \left(k, \mathbf{k}^{(m-1)}\right)\right) = \frac{\binom{n-k-1}{m-2}}{\binom{n}{m-1}}. \tag{5}$$

The last equation comes from the fact that there are $\binom{n-l-1}{m-2}$ ways of split $n-l-1$ keys into $m-1$ sublists (i.e. choosing $m-2$ pivots from $n-l-1$ keys); it also can be found, e.g., in [8].

To finish the calculation of $H(U_n)$, we need to calculate $H(\mathbf{Y}_n^{(m)})$. >From (3) we have

$$\begin{aligned} H\left(\mathbf{Y}_n^{(m)}\right) &= - \sum_{\|\mathbf{k}\| = n-m+1} \mathbb{P}\left(\mathbf{Y}_n^{(m)} = \mathbf{k}^{(m)}\right) \log \mathbb{P}\left(\mathbf{Y}_n^{(m)} = \mathbf{k}^{(m)}\right) \\ &= \frac{1}{\binom{n}{m-1}} \log \binom{n}{m-1} \sum_{\|\mathbf{k}\| = n-m+1} 1 = \log \binom{n}{m-1}. \end{aligned}$$

The last equality comes from the fact that the sum $\sum_{\|\mathbf{k}\| = n-m+1} 1$ equals to the number of choices of $m-1$ pivots from n keys, which is $\binom{n}{m-1}$.

Finally, we arrive at the following recurrence for the entropy of unlabeled m-ary search trees:

$$H(U_n) = \log\binom{n}{m-1} + \frac{m}{\binom{n}{m-1}} \sum_{k=0}^{n-m+1} \binom{n-k-1}{m-2} H(U_k).$$

The asymptotics of a recurrence like this one have been studied before; see Proposition 7 in [5] and Theorem 2.4 in [9], which we quote below.

Theorem 3.1 ([9], Theorem 2.4, Asymptotic Transfer Theorem). *Let*

$$a_n = b_n + \frac{m}{\binom{n}{m-1}} \sum_{j=0}^{n-(m-1)} \binom{n-1-j}{m-2} a_j, \quad n \geq m-1,$$

with specified initial conditions (say) $a_j := b_j$, $0 \leq j \leq m-2$. *If*

$$b_n = o(n) \quad \text{and} \quad \sum_{n\geq 0} \frac{b_n}{(n+1)(n+2)} \text{ converges,}$$

then

$$a_n = \frac{K_1}{\mathcal{H}_m - 1} n + o(n), \quad \text{where} \quad K_1 := \sum_{j\geq 0} \frac{b_j}{(j+1)(j+2)}.$$

Here, $\mathcal{H}_m$ *is the mth harmonic number.*

Hence, the entropy of the m-ary search tree becomes

$$H(U_n) = c_m n + o(n),$$

where

$$c_m = 2\phi_m \sum_{k\geq 0} \frac{\log\binom{k}{m-1}}{(k+1)(k+2)}$$

and $\phi_m = \frac{1}{2\mathcal{H}_m - 2}$ is called occupancy constant.

Observe that if $m = 2$ the number of nodes of an m-ary search tree equals n, the number of inserted keys; but for $m > 2$ the number of nodes is a random variable $S_{n,m}$. Knuth [13] was the first to show that $\mathbb{E}(S_{n,m}) \sim \phi_m n$.

In order to compare the constant c_m of m-ary search trees with that for d-ary increasing trees (discussed next), we note that m-ary search trees of size n have on average $\sim \phi_m n$ internal nodes. Thus, it makes sense to normalize the constant c_m as $\hat{c}_m = c_m/\phi_m$. Then numerically $\hat{c}_2 \approx 1.73638$, $\hat{c}_3 \approx 2.5014$, $\hat{c}_4 \approx 2.93994$, $\hat{c}_5 \approx 3.22688$.

Using Theorem 3.1 and the fact that $H\left(\mathrm{Y}_n^{(m)}\right) = \log\binom{n}{m-1} = o(n)$, we conclude this section with the following result.

Corollary 3.2. *The entropy rate* $h_{m,u} = \lim_{n\to\infty} H(U_n)/n$ *of the unlabeled m-ary trees, generated according to the m-ary search tree model, is given by*

$$h_{m,u} = 2\phi_m \sum_{k\geq 0} \frac{\log\binom{k}{m-1}}{(k+1)(k+2)}. \tag{6}$$

Remark 1. It can be checked numerically that the entropy rate for unlabeled m-ary search trees for $m = 2, 3, 4, 5$ is $h_{2,u} \approx 1.73638$, $h_{3,u} \approx 1.50084$, $h_{4,u} \approx 1.3569$, $h_{5,u} \approx 1.25723$ respectively. Notice that the entropy rate decreases as we increase m. Intuitively, this is to be expected: consider the extreme case, with $m = n$ (though, in the rest of the paper, we always consider constant m). In this case, every pattern of insertions into an m-ary search tree results in the same unlabeled tree (since the m slots of the root are never filled), so the entropy is 0.

3.2 The Entropy of the Unlabeled d-ary Plane Increasing Trees

Let $g_n = |\mathcal{G}_n|$ be the number of d-ary plane increasing trees with n internal nodes. From [8] we know that for $d = 2$ we have $g_n = n!$. Moreover, for $d > 2$ we have

$$g_n = (-1)^n (d-1)^n \frac{\Gamma(2 - \frac{d}{d-1})}{\Gamma(2 - \frac{d}{d-1} - n)} \tag{7}$$

which is a consequence of the *hook length formula* [13].

Let us briefly review the hook length formula and see how it is related to counting the number of increasing labelings for a given tree. The relation between the hook length formula and increasing trees belongs to folklore and is in fact an exercise of [13, p. 67].

Lemma 3.3. *(Hook length formula) The number $l_{\mathtt{t}}$ of increasing trees induced by an unlabeled plane rooted tree $\mathtt{t}$ is*

$$l_{\mathtt{t}} = \frac{|\mathtt{t}|!}{\prod_{\mathtt{s} \text{ subtree of } \mathtt{t}} |\mathtt{s}|}, \tag{8}$$

where $|\cdot|$ corresponds to the tree size measure.

From this we conclude the following corollary

Corollary 3.4. *The probability of an unlabeled plane rooted tree $\mathtt{t}$ obtained by removing labels from a plane increasing rooted tree is*

$$\mathbb{P}(T = \mathtt{t}) = \frac{|\mathtt{t}|!}{g_{|\mathtt{t}|} \prod_{\mathtt{s} \text{ subtree of } \mathtt{t}} |\mathtt{s}|}, \tag{9}$$

where $g_{|\mathtt{t}|}$ is the number of increasing trees of size $|\mathtt{t}|$.

Observe that (9) works for different kinds of increasing trees, we just have to put the right g_n in it. In the case of d-ary increasing trees it is (7).

Let $\mathcal{G}_{\mathtt{f}_n}$ denote the subset of $\mathcal{G}_n$ of trees that have the same structure as the unlabeled tree $\mathtt{f}_n \in \mathcal{F}_n$; that is, $\mathcal{G}_{\mathtt{f}_n}$ is the set of labeled representatives of $\mathtt{f}_n$. Moreover, let $g_{\mathtt{f}_n} = |\mathcal{G}_{\mathtt{f}_n}|$ be the number of d-ary plane increasing trees that have the same structure as a tree $\mathtt{f}_n$. Observe that the probability that the d-ary plane increasing tree source generates a given unlabeled tree $\mathtt{f}_n \in \mathcal{F}_n$ is

$$\mathbb{P}(F_n = \mathtt{f}_n) = \frac{g_{\mathtt{f}_n}}{g_n}. \tag{10}$$

Suppose that the tree $\mathtt{f}_n$ has subtrees $\mathtt{f}_{k_1}, \ldots, \mathtt{f}_{k_d}$ of sizes $k_1, \ldots, k_d$. Then

$$\mathbb{P}(F_n = \mathtt{f}_n) = \frac{1}{g_n} \binom{n-1}{k_1, \ldots, k_d} \prod_{j=1}^{d} g_{\mathtt{f}_{k_j}} = \binom{n-1}{k_1, \ldots, k_d} \frac{g_{k_1} \cdots g_{k_d}}{g_n} \prod_{j=1}^{d} \mathbb{P}\left(F_{k_j} = \mathtt{f}_{k_j}\right). \tag{11}$$

Observe that $\binom{n-1}{k_1,\ldots,k_d} \frac{g_{k_1} \cdots g_{k_d}}{g_n}$ is the probability that the subtrees of the root are of sizes $k_1, \ldots, k_d$ and thus (11) can be also derived from Equation (2). Let us define a random vector $\mathbf{V}_n^{(d)} : \mathcal{G}_n \to \{0, \ldots, n-1\}^d$ whose jth component $V_{n,j}$ denotes the size of the jth subtree. For $n \geq 1$ we have $V_{n,1} + \ldots + V_{n,d} = n - 1$ and

$$\mathbb{P}\left(\mathbf{V}_n^{(d)} = \mathbf{k}^{(d)}\right) = \binom{n-1}{k_1, \ldots, k_d} \frac{g_{k_1} \cdots g_{k_d}}{g_n}. \tag{12}$$

The entropy of unlabeled d-ary plane increasing trees of size n is given by

$$H(F_n) = -\sum_{\mathtt{f}_n \in \mathcal{F}_n} \mathbb{P}(F_n = \mathtt{f}_n) \log(\mathbb{P}(F_n = \mathtt{f}_n)).$$

Let us establish the initial conditions of the entropy of our source. If $n = 0$ we have an empty tree, and $H(F_0) = 0$. If $n = 1$, we have one fixed tree and $H(F_1) = 0$. By (11) for $n > 1$ there is a bijection between a tree F_n and a tuple $(\mathbf{V}_n^{(d)}, F_{V_1}, \ldots, F_{V_d})$. Therefore, for $n > 1$, we have

$$\begin{aligned} H(F_n) &= H\left(\mathbf{V}_n^{(d)}, F_{V_{n,1}}, \ldots, F_{V_{n,d}}\right) = H\left(\mathbf{V}_n^{(d)}\right) + H\left(F_{V_{n,1}}, \ldots, F_{V_{n,d}} | \mathbf{V}_n^{(d)}\right) \\ &= H\left(\mathbf{V}_n^{(d)}\right) + \sum_{\|\mathbf{k}\|=n-1} H\left(F_{k_1}, \ldots, F_{k_d}\right) \mathbb{P}\left(\mathbf{V}_n^{(d)} = \mathbf{k}^{(d)}\right). \end{aligned}$$

Since subtrees $F_{k_1}, \ldots, F_{k_d}$ are conditionally independent given their sizes, we have

$$H(F_n) = H\left(\mathbf{V}_n^{(d)}\right) + d\sum_{k=0}^{n-1} H(F_k) \sum_{\|\mathbf{k}^{(d-1)}\|=n-1-k} \mathbb{P}\left(\mathbf{V}_n^{(d)} = \left(k, \mathbf{k}^{(d-1)}\right)\right).$$

For $k = 0, \ldots, n-1$, let $p_{n,k}$ be the probability that one specified subtree in a d-ary increasing tree is of size k, that is,

$$p_{n,k} = \sum_{\|\mathbf{k}^{(d-1)}\|=n-1-k} \mathbb{P}\left(\mathbf{V}_n^{(d)} = \left(k, \mathbf{k}^{(d-1)}\right)\right). \tag{13}$$

Then

$$H(F_n) = H\left(\mathbf{V}_n^{(d)}\right) + d\sum_{k=0}^{n-1} H(F_k)\, p_{n,k}. \tag{14}$$

The next lemma, proved in Section 4.1, gives an explicit formula for $p_{n,k}$.

Lemma 3.5. *For $k = 0, \ldots, n-1$ and $d > 1$, let $\alpha = \frac{d}{d-1}$, then*

$$p_{n,k} = \frac{(\alpha-1)}{n} \frac{n!\Gamma(k+\alpha-1)}{k!\Gamma(n+\alpha-1)}.$$

Remark 2. Observe that for $d = 2$, we have $\alpha = 2$ and $p_{n,k} = \frac{1}{n}$. It does not depend on k, which greatly simplifies computations as shown in [15]. Moreover, it equals $\mathbb{P}\left(Y_{n,1} = k\right)$ in the case of the binary search trees. Therefore, the two models, one for binary plane increasing trees and the other for binary search trees, are equal. For $d > 2$, this is not the case. For instance, for $d = 3$, we have $\alpha = \frac{3}{2}$ and

$$p_{n,k} = \frac{1}{2n} \frac{\binom{2k}{k} 2^{2n}}{\binom{2n}{n} 2^{2k}},$$

which clearly depends on k and does not equal $\frac{n-k-1}{\binom{n}{2}}$, which would be the case for 3-ary search trees.

The recurrence presented in (14) is a novel one that we need to solve. In the lemma below we propose a general solution for recurrences of this form. It is proved in Section 4.2.

Lemma 3.6 (Exact solution to a generalized entropy recurrence for d-ary trees). *For constant α, x_0 and x_1, the recurrence*

$$x_n = a_n + \frac{\alpha}{n} \frac{n!}{\Gamma(n+\alpha-1)} \sum_{k=0}^{n-1} \frac{\Gamma(k+\alpha-1)}{k!} x_k, \qquad n \geq 2 \tag{15}$$

has the following solution for $n \geq 2$:

$$x_n = a_n + \alpha(n+\alpha-1) \sum_{k=0}^{n-1} \frac{a_k}{(k+\alpha-1)(k+\alpha)} + \frac{n+\alpha-1}{\alpha+1}\left(x_1 + \frac{x_0}{\alpha-1}\right).$$

Remark 3. Observe again that for $d = 2$ and $x_0 = x_1 = 0$ and $a_n = o(n)$, we have $\alpha = 2$ and

$$x_n = 2n \sum_{k\geq 0} \frac{a_k}{(k+1)(k+2)} + o(n)$$

as in [15]. But with the same assumptions and $d = 3$ we have

$$x_n = \frac{3}{2} n \sum_{k\geq 0} \frac{a_k}{(k+\frac{1}{2})(k+\frac{3}{2})} + o(n).$$

This leads us to out first main result.

Theorem 3.7. *The entropy of an unlabeled d-ary plane tree, generated according to the d-ary plane increasing tree model, is given by (see Figure 4)*

$$H(F_n) = H\left(\mathbf{V}_n^{(d)}\right) + \alpha(n+\alpha-1) \sum_{k=0}^{n-1} \frac{H\left(\mathbf{V}_k^{(d)}\right)}{(k+\alpha-1)(k+\alpha)}, \tag{16}$$

where $\alpha = \frac{d}{d-1}$ and

$$H\left(\mathbf{V}_n^{(d)}\right) = -\sum_{\|\mathbf{k}\|=n-1} \mathbb{P}\left(\mathbf{V}_n^{(d)} = \mathbf{k}^{(d)}\right) \log \mathbb{P}\left(\mathbf{V}_n^{(d)} = \mathbf{k}^{(d)}\right).$$

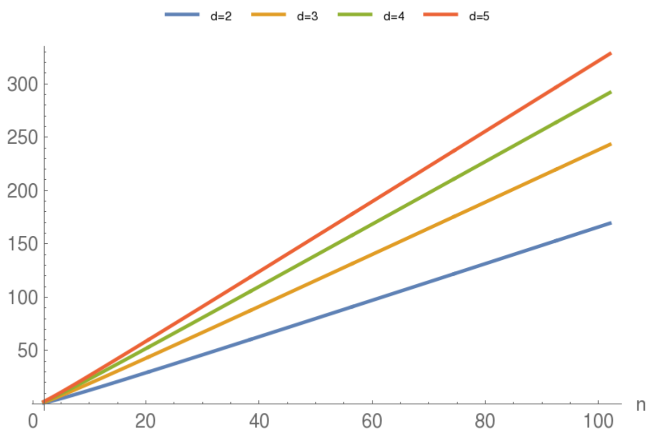


Fig. 4. Entropy of d-ary plane increasing trees for $d = 2, 3, 4, 5$ respectively and number of nodes $n \in [1, 100]$.

We conclude this section with the following result.

Corollary 3.8. *The entropy rate $h_{d,f} = \lim_{n\to\infty} H(F_n)/n$ of the unlabeled d-ary plane trees, generated according to the model of d-ary plane increasing trees, is given by*

$$h_{d,f} = \alpha \sum_{k=0}^{\infty} \frac{H(\mathbf{V}_k)}{(k+\alpha-1)(k+\alpha)}, \tag{17}$$

with $\alpha = \frac{d}{d-1}$.

Proof. Having Theorem 3.7 in mind, we just need to prove that $H\left(\mathbf{V}_n^{(d)}\right) = o(n)$. Let us recall that $\mathbf{V}_n^{(d)} : \mathcal{G}_n \to \{0, \ldots, n-1\}^d$. Since the entropy of random variable is upper bounded by the logarithm of the variable image cardinality, we have

$$H\left(\mathbf{V}_n^{(d)}\right) \le \log\left(n^d\right) = o(n)$$

as needed. □

Remark 4. Taking a closer look at $H\left(\mathbf{V}_n^{(d)}\right)$ we find

$$H\left(\mathbf{V}_n^{(d)}\right) = \log\left(n\frac{g_n}{n!}\right) - d\sum_{k=0}^{n-1} p_{n,k}\log\left(\frac{g_k}{k!}\right).$$

In particular, $H\left(\mathbf{V}_n^{(2)}\right) = H\left(\mathbf{Y}_n^{(2)}\right) = \log(n-1)$ and the entropy rate $h_{2,f} \approx 1.73638$, which matches the entropy rate of the binary search trees. On the other hand, for $d = 3$ and $n > 0$ we have

$$H\left(\mathbf{V}_n^{(3)}\right) = \log\left(\frac{n}{2^n}\binom{2n}{n}\right) - \frac{3}{2n}\sum_{k=0}^{n-1}\frac{\binom{2k}{k}2^{2n}}{\binom{2n}{n}2^{2k}}\log\left(\frac{\binom{2k}{k}}{2^k}\right).$$

This allows us to check numerically that the entropy rate of the unlabeled 3-ary plane trees, generated according to the model of 3-ary increasing trees is $h_{3,f} \approx 2.470$.

3.3 The Entropy of the Unlabeled General Plane Trees

Let $r_n = |\mathcal{R}_n|$, the number of labeled plane increasing trees with n nodes. From [3, 8] we know that there are

$$r_n = (2n-3)!! = \frac{n!}{n2^{n-1}}\binom{2n-2}{n-1} \tag{18}$$

different labeled plane oriented increasing trees of size n.

As in the case of the d-ary plane increasing trees, let $\mathcal{R}_{\mathtt{t}_n}$ denote the subset of trees in $\mathcal{R}_n$ that have the same structure as a given unlabeled tree $\mathtt{t}_n \in \mathcal{T}_n$ (i.e., $\mathcal{R}_{\mathtt{t}_n}$ is the set of labeled *representatives* of $\mathtt{t}_n$); moreover, let $r_{\mathtt{t}_n} = |\mathcal{R}_{\mathtt{t}_n}|$ be the number of such trees. Observe that

$$\mathbb{P}(T_n = \mathtt{t}_n) = \frac{r_{\mathtt{t}_n}}{r_n}. \tag{19}$$

Let D_n denote the random variable representing the number of subtrees of the root. Observe that $\mathbb{P}(D_n = d) = \frac{r_n^{(d)}}{r_n}$, where $r_n^{(d)} = |\mathcal{R}_n^{(d)}|$ is the number of plane increasing trees with root degree equal to d. Suppose that the tree $\mathtt{t}_n$ has d subtrees $\mathtt{t}_{k_1}, \ldots, \mathtt{t}_{k_d}$ of sizes $k_1, \ldots, k_d$. Then by (2) and the fact that $\mathbb{P}(T_n = t_n) = \mathbb{P}(T_n = t_n, D_n = d)$ by assumption,

$$\mathbb{P}(T_n = \mathtt{t}_n) = \mathbb{P}(D_n = d)\,\mathbb{P}(T_n = \mathtt{t}_n | D_n = d) = \binom{n-1}{k_1, \ldots, k_d}\frac{r_{k_1}\cdots r_{k_d}}{r_n}\prod_{j=1}^{d}\mathbb{P}\left(T_{k_j} = \mathtt{t}_{k_j}\right). \tag{20}$$

Observe that $\binom{n-1}{k_1,\ldots,k_d}\frac{r_{k_1}\cdots r_{k_d}}{r_n}$ is the probability that the root of a plane increasing tree of size n has degree equal to d and the root's subtrees are of sizes $k_1, \ldots, k_d$. Let $\mathbf{W}_n^{(d)} : \mathcal{R}_n^{(d)} \to \{1, \ldots, n-d\}^d$, where its jth component $W_{n,j}$ denotes the size of the jth subtree when the root is of degree d. For $n \ge 1$ we have $W_{n,1} + \ldots + W_{n,d} = n-1$ and

$$\mathbb{P}(D_n = d)\,\mathbb{P}\left(\mathbf{W}_n^{(D_n)} = \mathbf{k}^{(D_n)} | D_n = d\right) = \binom{n-1}{k_1, \ldots, k_d}\frac{r_{k_1}\cdots r_{k_d}}{r_n}. \tag{21}$$

The entropy of unlabeled plane increasing trees of size n is given by

$$H(T_n) = -\sum_{\mathrm{t}_n \in \mathcal{T}_n} \mathbb{P}(T_n = \mathrm{t}_n) \log (\mathbb{P}(T_n = \mathrm{t}_n)).$$

The initial conditions for the entropy are as follows. If $n = 1$, we have just a root node, so $H(T_1) = 0$. Similarly, if $n = 2$, we have one fixed tree, so $H(T_2) = 0$. Let us observe that for $n > 2$ and the tree's root degree equal to d, there is a bijection between a tree T_n and a tuple $(\mathbf{W}_n^{(d)}, T_{W_{n,1}}, \dots, T_{W_{n,d}})$ which is an immediate consequence of (20). Therefore, for $n > 2$, we have

$$\begin{aligned}
H(T_n) &= \sum_{d=1}^{n-1} H\left(\mathbf{W}_n^{(d)}, T_{W_{n,1}}, \dots, T_{W_{n,d}} | D_n = d\right) \mathbb{P}(D_n = d) \\
&= \sum_{d=1}^{n-1} \left(H\left(\mathbf{W}_n^{(d)} | D_n = d\right) + H\left(T_{W_{n,1}}, \dots, T_{W_{n,d}} | \mathbf{W}_n^{(d)}, D_n = d\right) \right) \mathbb{P}(D_n = d) \\
&= \sum_{d=1}^{n-1} H\left(\mathbf{W}_n^{(d)} | D_n = d\right) \mathbb{P}(D_n = d) + \\
&\quad \sum_{d=1}^{n-1} \sum_{\|\mathbf{k}\|=n-1} H(T_{k_1}, \dots, T_{k_d}) \cdot \mathbb{P}\left(\mathbf{W}_n^{(d)} = \mathbf{k}^{(d)} | D_n = d\right) \mathbb{P}(D_n = d).
\end{aligned}$$

From conditional independence of $T_{k_1}, \dots, T_{k_d}$, we conclude

$$\begin{aligned}
H(T_n) &= \sum_{d=1}^{n-1} H\left(\mathbf{W}_n^{(d)} | D_n = d\right) \mathbb{P}(D_n = d) \\
&\quad + \sum_{d=1}^{n-1} \mathbb{P}(D_n = d)\, d \sum_{k=1}^{n-d} H(T_k) \sum_{\|\mathbf{k}^{(d-1)}\|=n-1-k} \mathbb{P}\left(\mathbf{W}_n^{(d)} = \left(k, \mathbf{k}^{(d-1)}\right)\right).
\end{aligned}$$

For $k = 1, \dots, n-1$, let $q_{n,k}^{(d)}$ be defined as the probability that the root of a plane increasing tree has degree d and that one specified root subtree is of size k. Then

$$q_{n,k}^{(d)} = \mathbb{P}(D_n = d) \sum_{\|\mathbf{k}^{(d-1)}\|=n-1-k} \mathbb{P}\left(\mathbf{W}_n^{(d)} = \left(k, \mathbf{k}^{(d-1)}\right)\right). \tag{22}$$

Therefore

$$H(T_n) = \sum_{d=1}^{n-1} H\left(\mathbf{W}_n^{(d)} | D_n = d\right) \mathbb{P}(D_n = d) + \sum_{d=1}^{n-1} d \sum_{k=1}^{n-d} H(T_k)\, q_{n,k}^{(d)}. \tag{23}$$

We need an expression for the probability $q_{n,k}^{(d)}$ which we present in the next lemma proved in Section 4.3.

Lemma 3.9. *For $k = 1, \dots, n-1$ we have*

- $q_{n,n-1}^{(1)} = \frac{1}{2n-3}$ *and if* $k \neq n-1$: $q_{n,k}^{(1)} = 0$,
- *for* $d > 1$*:*

$$q_{n,k}^{(d)} = 2^d \frac{d-1}{k(n-1-k)} \frac{\binom{2k-2}{k-1}\binom{2(n-1-k)-d}{n-2-k}}{\binom{2n-2}{n-1}}.$$

The recurrence found in (23) is another one that we need to analyze. Its general solution is presented next and proved in Section 4.4.

Lemma 3.10 (Exact solution to a generalized entropy recurrence for unrestricted trees). *For constant y_1 and y_2, the recurrence*

$$y_n = b_n + \sum_{d=1}^{n-1} d \sum_{k=1}^{n-d} q_{n,k}^{(d)} \cdot y_k, \qquad n > 2 \tag{24}$$

has the following solution for $n > 2$:

$$y_n = \frac{2(2n-1)}{3} b_1 + b_n + \frac{1}{2}\left(n - \frac{1}{2}\right) \sum_{j=2}^{n-1} \frac{b_j}{\left(j-\frac{1}{2}\right)\left(j+\frac{1}{2}\right)}.$$

This leads us to our second main result.

Theorem 3.11. *The entropy of an unlabeled general plane tree, generated according to the model of plane increasing tree, is given by*

$$H(T_n) = \sum_{d=1}^{n-1} H\left(\mathbf{W}_n^{(d)} | D_n = d\right) \mathbb{P}(D_n = d) + \frac{1}{2}\left(n - \frac{1}{2}\right) \sum_{j=2}^{n-1} \frac{\sum_{d=1}^{j-1} H\left(\mathbf{W}_j^{(d)} | D_j = d\right) \mathbb{P}(D_j = d)}{\left(j-\frac{1}{2}\right)\left(j+\frac{1}{2}\right)}, \tag{25}$$

where

$$H\left(\mathbf{W}_n^{(d)} | D_n = d\right) = - \sum_{\|\mathbf{k}\| = n-1} \mathbb{P}\left(\mathbf{W}_n^{(d)} = \mathbf{k}^{(d)} | D_n = d\right) \log \mathbb{P}\left(\mathbf{W}_n^{(d)} = \mathbf{k}^{(d)} | D_n = d\right).$$

We conclude this section with the following result.

Corollary 3.12. *The entropy rate $h_t = \lim_{n\to\infty} H(T_n)/n$ of the unlabeled general plane trees, generated according to the model of plane increasing trees, is given by*

$$h_t = \frac{1}{2} \sum_{j=2}^{\infty} \frac{\sum_{d=1}^{j-1} H\left(\mathbf{W}_k^{(d)} | D_n = d\right) \mathbb{P}(D_n = d)}{\left(j-\frac{1}{2}\right)\left(j+\frac{1}{2}\right)}. \tag{26}$$

Proof. From Theorem 3.11 we just need to prove that $\sum_{d=1}^{n-1} H\left(\mathbf{W}_n^{(d)} | D_n = d\right) \mathbb{P}(D_n = d) = o(n)$. Let us recall that the random vector $\mathbf{W}_n^{(d)} : \mathcal{R}_n^{(d)} \to \{1, \dots, n-d\}^d$ describes the split at the root of a tree: precisely that a tree root degree equals d and its subtrees are of sizes $W_{n,1}^{(d)}, \dots, W_{n,d}^{(d)}$. Since the entropy of random variable is upper bounded by the logarithm of the variable image cardinality, we have

$$\sum_{d=1}^{n-1} \mathbb{P}(D_n = d) H\left(\mathbf{W}_n^{(D_n)} | D_n = d\right) \le \sum_{d=1}^{n-1} \mathbb{P}(D_n = d) \log\left(n^d\right) = \log(n) \sum_{d=1}^{n-1} d\, \mathbb{P}(D_n = d).$$

Observe that $\mathbb{E}(D_n) = \sum_{d=1}^{n-1} d\, \mathbb{P}(D_n = d)$ is the expected value of the general plane increasing tree root degree. From [3] we know that $\mathbb{E}(D_n) = \sqrt{\pi n} + O(1)$, which gives us the desired result. □

Remark 5. Taking a closer look at $H\left(\mathbf{W}_n^{(D_n)} | D_n\right)$ we find that

$$H\left(\mathbf{W}_n^{(D_n)} | D_n\right) = \log\left(n \frac{r_n}{n!}\right) - \sum_{d=1}^{n-1} d \sum_{k=1}^{n-d} q_{n,k}^{(d)} \log\left(\frac{r_k}{k!}\right).$$

This allows us to check numerically that the entropy rate of the unlabeled general plane trees, generated according to the model of plane increasing trees is $h_t \approx 1.68$.

3.4 Optimal Compression

Here we address optimal compression of the trees generated by the sources under consideration. We will use a variant of the arithmetic coding method. We explain in detail the generalization for compression of unlabeled d-ary plane increasing trees (the case of general plane increasing trees can be handled along analogous lines).

In a nutshell, we first define a total order $<$ on the set of such trees with a given size. Having fixed this, we must show how to efficiently compute two quantities, for a given tree t: the probability of all trees $t' < t$, as well as the probability of t itself. Granted efficient procedures for computing these, we produce a subinterval $I(t)$ of $[0, 1]$, unique to t, which has length $|I(t)|$ equal to the probability of t and whose left endpoint is the probability of all trees $t' < t$. Arithmetic coding then prescribes that the code word corresponding to t be given by the binary expansion of the midpoint of the interval assigned to t, truncated to a length of $\lceil \log |I(t)| \rceil + 1$ bits. The expected length of this code is then easily at most the entropy of the source, plus at most 2 bits.

We now define the total order on the set $\mathcal{F}_n$ on unlabeled d-ary trees: we denote by $\prec$ the lexicographic order on tuples of non-negative integers, and then we have the following definition.

Definition 3.13 (Total order of the set of unlabeled d-ary plane trees). The relation $<$ on $\mathcal{F}$ is defined as follows: let $\mathtt{f}_1, \mathtt{f}_2 \in \mathcal{F}$ with subtrees sizes $(s_1, \ldots, s_d)$, $(k_1, \ldots, k_d)$ respectively, then $\mathtt{f}_1 < \mathtt{f}_2$ if and only if one of the following holds:

- $(s_1, \ldots, s_d) \prec (k_1, \ldots, k_d)$,
- or if $(s_1, \ldots, s_d) = (k_1, \ldots, k_d)$ and first subtree of $\mathtt{f}_1$ $<$ first subtree of $\mathtt{f}_2$,
- or if $(s_1, \ldots, s_d) = (k_1, \ldots, k_d)$, first subtree of $\mathtt{f}_1$ $=$ first subtree of $\mathtt{f}_2$ and second subtree of $\mathtt{f}_1$ $<$ second subtree of $\mathtt{f}_2$,
- ...
- or if $(s_1, \ldots, s_d) = (k_1, \ldots, k_d)$, first $d-1$ subtrees of $\mathtt{f}_1$ $=$ first $d-1$ subtrees of $\mathtt{f}_2$ and dth subtree of $\mathtt{f}_1$ $<$ dth subtree of $\mathtt{f}_2$.

It is simple to check that this is a total order. Next, we present an algorithm which computes the subinterval corresponding to an input tree $\mathtt{f} \in \mathcal{F}$ (see Algorithm 1).

This does exactly as intuitively described above: it implements a depth-first search of the input tree f, and at each step refining the current interval based on the split of vertices among the root subtrees of the current node.

Now, we explain more precisely the procedures CalculateSplitProbability and CalculateIntervalBegin. The former simply calculates the probability that a d-ary tree of size n has root subtrees of sizes $k_1, \ldots, k_d$ (giving the length of the next subinterval). This is illustrated in Figure 5.

The latter gives the probability that such a d-ary tree has a root subtree size tuple lexicographically less than $(s_1, \ldots, s_d)$. That is, it computes the expression

$$\sum_{\substack{(k_1, \ldots, k_d) \prec (s_1, \ldots, s_d) \\ k_1 + \ldots + k_d = n-1}} \binom{n-1}{k_1, \ldots, k_d} \frac{g_{k_1} \cdots g_{k_d}}{g_n}$$

Observe that a naive implementation of this calculation generates all $\Theta(n^d)$ integer partitions with d parts of the number $n-1$ and calculates the split probability for each of them. To reduce the

Algorithm 1 Unlabeled d-ary Plane Tree Compression

```
1:  function CompressDTree($\mathsf{f} \in \mathcal{F}$)                ▷ f tree to be compressed
2:      $[a, b) \leftarrow$ Explore(root of $\mathsf{f}$, $[0, 1)$)
3:      return first $\lceil -\log_2(b - a)\rceil + 1$ bits of $(a + b)/2$

4:  function Explore($v \in \mathsf{f}$, $[l, r) \subseteq [0, 1)$)
5:      $visited(v) \leftarrow true$
6:      $n \leftarrow$ size of a subtree of $\mathsf{f}$ hanging from node $v$
7:      $(s_1, \ldots, s_d) \leftarrow$ sizes of subtrees of $v$
8:      $a \leftarrow l + (r - l)\cdot$ CalculateIntervalBegin($n, s_1, \ldots, s_d$)
9:      $p \leftarrow (r - l)\cdot$ CalculateSplitProbability($n, s_1, \ldots, s_d$)
10:     $I_{\text{new}} \leftarrow [a, a + p)$
11:     for all $u$ descendant of $v$ do
12:         if not $visited(u)$ then
13:             $I_{\text{new}} \leftarrow$ Explore($u$, $I_{\text{new}}$)
14:     return $I_{\text{new}}$

15: function CalculateSplitProbability($n, k_1, \ldots, k_d$)
16:     return $\binom{n-1}{k_1,\ldots,k_d}\frac{g_{k_1}\cdots g_{k_d}}{g_n}$

17: function CalculateIntervalBegin($n, s_1, \ldots, s_d$)
18:     return
        $\frac{1}{g_n}\sum_{i=1}^{d}\frac{(n-1)!}{s_1!\cdots s_{i-1}!}\left(\prod_{j=1}^{i-1} g_{s_j}\right)\sum_{k=0}^{s_i-1}\frac{g_k}{k!}\binom{n-2-k-\sum_{l=1}^{i-1} s_l+\frac{d-i}{d-1}}{\frac{1-i}{d-1}}(d-1)^{n-1-k-\sum_{l=1}^{i-1} s_l}$
```

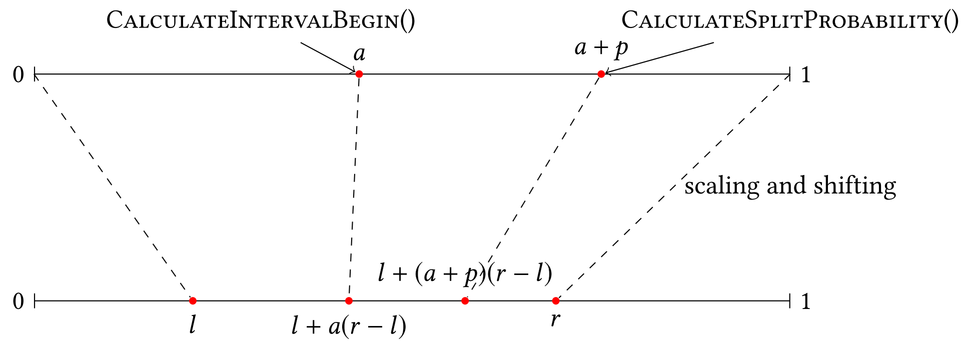
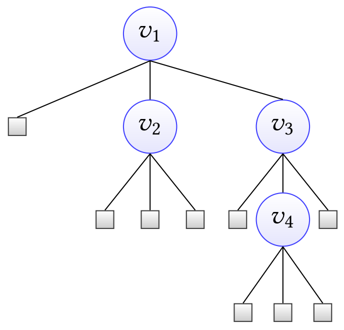


Fig. 5. Visualization of lines 8 – 10 in the Algorithm 1 listing.

time complexity, we rewrite the sum as follows:

$$\sum_{\substack{(k_1,\ldots,k_d) \prec (s_1,\ldots,s_d)\\ k_1+\ldots+k_d=n-1}} \binom{n-1}{k_1,\ldots,k_d}\frac{g_{k_1}\cdots g_{k_d}}{g_n} =$$

$$\frac{1}{g_n}\sum_{i=1}^{d} g_{s_1}\cdots g_{s_{i-1}}\sum_{k=0}^{s_i-1}\sum_{j_{i+1}+\ldots+j_d}\binom{n-1}{s_1,\ldots,s_{i-1},k,j_{i+1},\ldots,j_d} g_k g_{j_{i+1}}\cdots g_{j_d}.$$



The following example describes algorithm CompressDTree in detail for a given 3-ary tree. From Equation (9), probability of the given tree equals to $\frac{4!}{g_4 4\cdot 1\cdot 2\cdot 1} = \frac{1}{35}$. Moreover, consecutive calls of the Explore procedure for the tree vertices outputs the following intervals: $v_1 \to \left[\frac{1}{7}, \frac{8}{35}\right)$, $v_2 \to \left[\frac{1}{7}, \frac{8}{35}\right)$, $v_3 \to \left[\frac{6}{35}, \frac{1}{5}\right)$, $v_4 \to \left[\frac{6}{35}, \frac{1}{5}\right)$. In the last step $\lceil -\log_2\left(\frac{1}{5} - \frac{6}{35}\right)\rceil + 1 = 7$ bits of the $\frac{1}{2}\left(\frac{6}{35} + \frac{1}{5}\right) = \frac{13}{70} = 0.0010111110001010111 11...$ is returned, i.e. 0010111.

Fig. 6. Illustration to Algorithm 1

For a given i, the ith term of the outermost sum gives the contribution of all tuples of the form $(s_1, ..., s_{i-1}, k_i, k_{i+1}, ..., k_d)$ with varying $k_i, ..., k_d$. We can, furthermore, write the multinomial coefficient as a product of two other multinomial coefficients, one of which can be brought outside the k sum. The kth term of the resulting sum can then be written as follows:

$$\begin{aligned}
&\frac{1}{(n-1-k-\sum_{l=1}^{i-1} s_l)!} \sum_{j_{i+1}+\ldots+j_d} \binom{n-1-k-\sum_{l=1}^{i-1} s_l}{j_{i+1}, \ldots, j_d} g_{j_{i+1}} \cdots g_{j_d} \\
&= \left[z^{n-1-k-\sum_{l=1}^{i-1} s_l}\right] (1-(d-1)z)^{-\frac{d-i}{d-1}} \\
&= \binom{n-2-k-\sum_{l=1}^{i-1} s_l + \frac{d-i}{d-1}}{\frac{1-i}{d-1}} (d-1)^{n-1-k-\sum_{l=1}^{i-1} s_l}.
\end{aligned}$$

We thus get, finally,

$$\begin{aligned}
&\sum_{\substack{(k_1, \ldots, k_d) \prec (s_1, \ldots, s_d) \\ k_1 + \ldots + k_d = n-1}} \binom{n-1}{k_1, \ldots, k_d} \frac{g_{k_1} \cdots g_{k_d}}{g_n} \\
&= \frac{1}{g_n} \sum_{i=1}^{d} \frac{(n-1)!}{s_1! \cdots s_{i-1}!} \left(\prod_{j=1}^{i-1} g_{s_j}\right) \sum_{k=0}^{s_i-1} \frac{g_k}{k!} \binom{n-2-k-\sum_{l=1}^{i-1} s_l + \frac{d-i}{d-1}}{\frac{1-i}{d-1}} (d-1)^{n-1-k-\sum_{l=1}^{i-1} s_l}.
\end{aligned}$$

Observe that the resulting expression only requires to perform $O(n)$ calculations, where each of them is of the same order as the calculation of the split probability. An application of Algorithm 1 to a 3-tree is presented in Figure 6.

This leads us to the time complexity of the algorithm: observe that for each vertex v of an input tree f of size n, the procedure Explore calls one time both procedures CalculateSplitProbability and CalculateIntervalBegin. Therefore we get $O(n)$ calls of both procedures. Since the CalculateIntervalBegin procedure performs $O(n)$ calculations where each of them is of the same time complexity as one CalculateSplitProbability procedure, we have that the compression algorithm runs in time $O(n^2 \cdot f(d, n))$, where $f(d, n)$ denotes the number of bit operations in the CalculateSplitProbability procedure (it is not too difficult to see that $f(d, n)$ can be bounded by a small polynomial function of n alone, by taking into account cancellations of factors in the expression for the split probabilities).

n	50	100	150	200	250	300	350	400	450	500
$d = 3$										
$H(F_n)$	115.63	238.17	361.12	484.24	607.46	730.73	854.04	977.39	1100.75	1224.14
code length	117.04	239.46	362.07	484.94	608.70	731.56	853.72	978.67	1103.61	1223.66
$d = 8$										
$H(F_n)$	192.63	394.08	595.95	797.99	1000.12	1202.31	1404.54	1606.80	1809.09	2011.39
code length	193.85	396.05	599.79	799.56	1001.24	1205.07	1404.22	1609.57	1810.42	2012.90

Table 1. Comparison of the entropy of d-ary trees and the compressed code length obtained from experimental results (100 uniformly drawn trees for each d and n sizes of a tree).

Finally, standard arithmetic coding arguments show that the algorithm is *optimal* up to a small constant number of bits, in expectation. Experimental results confirming this statement can be seen in Table (1).

Remark 6. It is instructive to compare the performance of our algorithm with a natural compact representation (which we think of as an uncompressed representation) of the trees in question. In particular, in the *parenthesis representation*, a plane tree is encoded as a bit string via a depth-first search. When a node is first encountered (and pushed), a 0 is appended. When the same node is popped (i.e., the search leaves that node's subtree), a 1 is appended.

In such a representation, the number of bits needed per vertex of a d-ary tree is approximately $2d$. In Figure (7) we highlight the gain obtained by using our optimal compression algorithm. Although we observe that $h_{d,f} = \Theta(d)$ the difference in constant is significant.

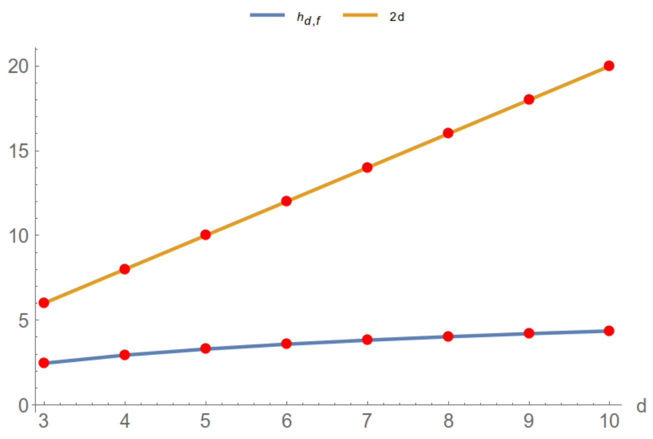


Fig. 7. Entropy rate of d-ary plane increasing trees and the number of bits needed to encode one vertex of d-ary tree in the parenthesis representation.

4 PROOFS OF TECHNICAL LEMMAS

4.1 Proof of Lemma 3.5

Using (12), we can rewrite (13) as

$$p_{n,k} = \frac{(n-1)!g_k}{k!(n-1-k)!g_n} \sum_{k_2+\ldots+k_d=n-1-k} \binom{n-1-k}{k_2,\ldots,k_d} g_{k_2}\cdots g_{k_d}.$$

Let us define the exponential generating function $G(z) = \sum_{n\geq 0} g_n \frac{z^n}{n!}$ with $g_0 = 1$. From [8] we know that

$$G(z) = (1-(d-1)z)^{-\frac{1}{d-1}}.$$

Observe that

$$\sum_{k_2+\ldots+k_d=n-1-k} \binom{n-1-k}{k_2,\ldots,k_d} g_{k_2}\cdots g_{k_d}$$

is the $n-1-k$th coefficient of the function $G(z)^{d-1}$ (denoted as $\left[\frac{z^{n-1-k}}{(n-1-k)!}\right] G(z)^{d-1}$). Hence

$$p_{n,k} = \frac{(n-1)!g_k}{k!(n-1-k)!g_n}\left[\frac{z^{n-1-k}}{(n-1-k)!}\right] G(z)^{d-1} = \frac{(n-1)!g_k}{k!g_n}\left[z^{n-1-k}\right]\frac{1}{1-(d-1)z}$$
$$= \frac{(n-1)!g_k}{k!g_n}(d-1)^{n-1-k}.$$

For $d = 2$, we have $g_n = n!$ and the result is immediate. For $d > 2$, from (7) we find

$$p_{n,k} = \frac{(\alpha-1)}{n}\frac{(-1)^n n!\Gamma(2-\alpha-n)}{(-1)^k k!\Gamma(2-\alpha-k)}.$$

From [21] we know that $\Gamma(z-n) = \frac{(-1)^n\pi}{\Gamma(n+1-z)\sin(\pi z)}$; hence

$$(-1)^n\Gamma(n+\alpha)\Gamma(2-\alpha-n) = \frac{\pi\cdot(n-1+\alpha)}{\sin(\pi(2-\alpha))}, \tag{27}$$

and then

$$p_{n,k} = \frac{(\alpha-1)}{n}\frac{n!\Gamma(k+\alpha)(n+\alpha-1)}{k!\Gamma(n+\alpha)(k+\alpha-1)}.$$

Since $\Gamma(z+1) = z\Gamma(z)$ we get the desired result.

4.2 Proof of Lemma 3.6

Let us multiply both sides of the recurrence by the normalizing factor $\frac{\Gamma(n+\alpha-1)}{n!}$. Define also

$$\hat{x}_n = \frac{x_n\Gamma(n+\alpha-1)}{n!}, \quad \hat{a}_n = \frac{a_n\Gamma(n+\alpha-1)}{n!}.$$

Then

$$\hat{x}_n = \hat{a}_n + \frac{\alpha}{n}\sum_{k=2}^{n-1}\hat{x}_k. \tag{28}$$

To solve the recurrence (28) we compute $n\hat{x}_n - (n-1)\hat{x}_{n-1}$. This leads us to

$$\hat{x}_n = \hat{a}_n - \left(1-\frac{1}{n}\right)\hat{a}_{n-1} + \left(1+\frac{\alpha-1}{n}\right)\hat{x}_{n-1},$$

which holds for $n \geq 3$. Then after iterating the above we arrive at

$$\hat{x}_n = \hat{x}_2 \prod_{j=3}^{n} \left(1 + \frac{\alpha - 1}{j}\right) + \sum_{k=3}^{n} \left(\hat{a}_k - \left(1 - \frac{1}{k}\right) \hat{a}_{k-1}\right) \prod_{j=k+1}^{n} \left(1 + \frac{\alpha - 1}{j}\right). \tag{29}$$

The product $\prod_{j=k+1}^{n} \left(1 + \frac{\alpha-1}{j}\right) = \frac{k!\Gamma(n+\alpha)}{n!\Gamma(k+\alpha)}$, and after some standard calculations we obtain

$$\hat{x}_n = \hat{a}_n + (\hat{x}_2 - \hat{a}_2)\frac{2\Gamma(n+\alpha)}{\Gamma(\alpha+2)n!} + \frac{\Gamma(n+\alpha)}{n!} \sum_{k=2}^{n-1} \hat{a}_k \frac{k!}{\Gamma(k+\alpha)} \frac{\alpha}{k+\alpha}.$$

Going back from $\hat{x}_n$ and $\hat{a}_n$ to x_n, a_n, respectively, we obtain

$$x_n = a_n + \alpha(n+\alpha-1) \sum_{k=2}^{n-1} \frac{a_k}{(k+\alpha-1)(k+\alpha)} + (x_2 - a_2)\frac{n+\alpha-1}{\alpha+1}.$$

Observe that $x_2 - a_2 = x_1 + \frac{x_0}{\alpha-1}$. This completes the proof.

4.3 Proof of Lemma 3.9

If $d = 1$ then the root has only 1 subtree with all other nodes, so its size has to be equal to $n-1$ and

$$q_{n,n-1}^{(1)} = \frac{r_{n-1}}{r_n} = \frac{1}{2n-3};$$

moreover, if $k \neq n-1 : q_{n,k}^{(1)} = 0$. In the case of $d > 1$, using (21), we can rewrite (22) as follows

$$q_{n,k}^{(d)} = \frac{(n-1)!r_k}{k!(n-1-k)!r_n} \times \sum_{k_2+\ldots+k_d=n-1-k} \binom{n-1-k}{k_2, \ldots, k_d} r_{k_2} \cdots r_{k_d}.$$

Let us define the exponential generating function $R(z) = \sum_{n\geq 0} r_n \frac{z^n}{n!}$ with $g_0 = 0$. Observe that

$$\sum_{k_2+\ldots+k_d=n-1-k} \binom{n-1-k}{k_2, \ldots, k_d} r_{k_2} \cdots r_{k_d}$$

is the $n-1-k$th coefficient of the function $R(z)^{d-1}$ (denoted as $\left[\frac{z^{n-1-k}}{(n-1-k)!}\right] R(z)^{d-1}$). Therefore,

$$q_{n,k} = \frac{(n-1)!r_k}{k!r_n} \left[z^{n-1-k}\right] R(z)^{d-1}.$$

>From (18) we find $R(z) = 1 - \sqrt{1-2z}$, which is also the solution of the equation

$$R = \frac{z}{1 - \frac{R}{2}}.$$

Hence, by Lagrange's inversion formula (see [10]), we obtain explicit formula for

$$[z^{n-1-k}]R(z)^{d-1} = 2^{d-n+k} \frac{d-1}{n-1-k} \binom{2(n-1-k)-d}{n-2-k}.$$

Putting everything together we arrive at the desired result.

4.4 Proof of Lemma 3.10

Using Lemma 3.9, for $n > 2$, we find

$$y_n = b_n + \frac{y_{n-1}}{2n-3} + \sum_{d=2}^{n-1} d(d-1)2^d \sum_{k=1}^{n-d} \frac{y_k}{k(n-1-k)} \frac{\binom{2k-2}{k-1}\binom{2n-2k-2-d}{n-k-2}}{\binom{2n-2}{n-1}}.$$

Multiplying both sides by $\frac{\binom{2n-2}{n-1}}{n}$ and substituting $\hat{y}_n = \frac{y_n\binom{2n-2}{n-1}}{n}$, $\hat{b}_n = \frac{b_n\binom{2n-2}{n-1}}{n}$ we get

$$\hat{y}_n = \hat{b}_n + \frac{2\hat{y}_{n-1}}{n(n-1)} + \frac{1}{n}\sum_{d=2}^{n-1} d(d-1)2^d \sum_{k=1}^{n-d} \frac{\hat{y}_k}{(n-1-k)}\binom{2n-2k-2-d}{n-k-2}.$$

Changing the order of summation gives us

$$\sum_{d=2}^{n-1} d(d-1)2^d \sum_{k=1}^{n-d} \frac{\hat{y}_k}{(n-1-k)}\binom{2n-2k-2-d}{n-k-2} = \sum_{j=1}^{n-2} \frac{\hat{y}_j}{n-j-1} \sum_{s=0}^{n-j} s(s-1)2^s \binom{2n-2j-2-s}{n-j-2}.$$

Since for $N > 0$:

$$\sum_{s=0}^{N} s(s-1)2^s \binom{2N-2-s}{N-2} = (N-1)2^{2N-1},$$

we obtain

$$\hat{y}_n = \hat{b}_n + \frac{2\hat{y}_{n-1}}{n(n-1)} + \frac{1}{n}\sum_{j=1}^{n-2} \hat{y}_j 2^{2n-2j-1}.$$

Dividing both sides by 2^{2n} and substituting $\tilde{y}_n = \frac{\hat{y}_n}{2^{2n}}$, $\tilde{b}_n = \frac{\hat{b}_n}{2^{2n}}$ we find

$$\tilde{y}_n = \tilde{b}_n + \frac{1}{2n}\sum_{j=1}^{n-1} \tilde{y}_j.$$

Solving this recurrence relation by calculating $n\tilde{y}_n - (n-1)\tilde{y}_{n-1}$ we obtain

$$\tilde{y}_n = b_1 \frac{\Gamma\left(n+\frac{1}{2}\right)}{\Gamma\left(\frac{5}{2}\right) n!} + \tilde{b}_n + \frac{\Gamma\left(n+\frac{1}{2}\right)}{n!} \sum_{j=2}^{n-1} \frac{\tilde{b}_j}{2j+1} \frac{j!}{\Gamma\left(j+\frac{1}{2}\right)}.$$

Substituting $\tilde{y}_n$ into y_n with $\tilde{y}_n = y_n \frac{\binom{2n-2}{n-1}}{n2^{2n}}$, we find the desired result.

5 CONCLUDING REMARKS

In this paper we focused on finding entropies of various trees: namely, m-ary search trees, d-ary increasing trees, and general increasing trees. In the course of deriving these entropies we encountered novel recurrences that we showed how to solve in a general setting. These recurrences find ample applications in analyzing variations on such general trees. For example, as in [15], the next natural question is to find entropy of non-plane d-ary trees and general trees. For arbitrary d, such a problem is quite challenging as one can see the approach of [7] for the binary case. Generalization to the d-ary case is highly non-trivial.

Finally, we presented optimal, polynomial-time (with small exponents) compression algorithms that achieve these entropies via an arithmetic coding approach.

ACKNOWLEDGMENT

This work was supported by NSF Center for Science of Information (CSoI) Grant CCF-0939370, by NSF Grant CCF-1524312, and NIH Grant 1U01CA198941-01. Z. Golebiewski was partially supported by Polish NCN grant 2013/09/B/ST6/02258.

REFERENCES

[1] M. Adler, M. Mitzenmacher, Towards Compressing Web Graphs, *Data Compression Conference* 2001, pp. 203-212.

[2] D. Aldous and N. Ross, Entropy of Some Models of Sparse Random Graphs With Vertex-Names. *Probability in the Engineering and Informational Sciences*, 2014, 28:145-168.

[3] François Bergeron, Philippe Flajolet, and Bruno Salvy. Varieties of increasing trees. In Jean-Claude Raoult, editor, *CAAP '92, 17th Colloquium on Trees in Algebra and Programming, Rennes, France, February 26-28, 1992, Proceedings*, volume 581 of *Lecture Notes in Computer Science*, pages 24–48. Springer, 1992.

[4] F. Chierichetti, R. Kumar, S. Lattanzi, M. Mitzenmacher, A. Panconesi, and P. Raghavan. On Compressing Social Networks, *Proc. ACM KDD*, 2009.

[5] Hua-Huai Chern and Hsien-Kuei Hwang. Phase changes in random m-ary search trees and generalized quicksort. *Random Struct. Algorithms*, 19(3-4):316–358, 2001.

[6] Yongwook Choi, Wojciech Szpankowski: Compression of Graphical Structures: Fundamental Limits, Algorithms, and Experiments. *IEEE Transactions on Information Theory*, 2012, 58(2):620-638.

[7] J. Cichon, A. Magner, W. Szpankowski, K. Turowski, On symmetries of non-plane trees in a non-uniform model, *ANALCO*, Barcelona, 2017.

[8] Michael Drmota. *Random Trees, An Interplay between Combinatorics and Probability.* Springer-Verlag Wien, 2009.

[9] James Allen Fill and Nevin Kapur. Transfer theorems and asymptotic distributional results for m-ary search trees. *Random Structures & Algorithms*, 26(4):359–391, 2005.

[10] Philippe Flajolet and Robert Sedgewick. *Analytic Combinatorics.* Cambridge University Press, 2009.

[11] Zbigniew Golebiewski, Marcin Kardas, Jakub Lemiesz, Krzysztof Majcher, On structural entropy of uniform random intersection graphs, *ISIT'17*, Aachen, 2017.

[12] J. C. Kieffer, E.-H. Yang, W. Szpankowski, Structural complexity of random binary trees. *ISIT 2009* , pp. 635-639.

[13] Donald E. Knuth. *The Art of Computer Programming, Volume 3: (2Nd Ed.) Sorting and Searching.* Addison Wesley Longman Publishing Co., Inc., Redwood City, CA, USA, 1998.

[14] T. Luczak, A. Magner and W. Szpankowski, Structural Informtion and Compression of Scale-Free Graphs, preprint 2017.

[15] A. Magner, W. Szpankowski, K. Turowski, Lossless Compression of Binary Trees with Correlated Vertex Names, *ISIT'16*, Barcelona, 2016.

[16] M. Mohri, M. Riley, A. T. Suresh, Automata and graph compression. *ISIT 2015*, pp. 2989-2993.

[17] L. Peshkin, Structure induction by lossless graph compression, *In Proc. of the IEEE Data Compression Conference*, 53–62, 2007.

[18] W. Szpankowski, *Average Case Analysis of Algorithms on Sequences.* John Wiley & Sons, Inc., New York, NY, 2001.

[19] J. Sun, E.M. Bollt, and D. Ben-Avraham, Graph compression–save information by exploiting redundancy, *Journal of Statistical Mechanics: Theory and Experiment*, P06001, 2008.

[20] J. Zhang, E.-H. Yang, J. C. Kieffer, A Universal Grammar-Based Code for Lossless Compression of Binary Trees. *IEEE Transactions on Information Theory*, 2014, 60(3):1373-1386.

[21] Daniel (ed.) Zwillinger. *CRC Standard Mathematical Tables and Formulae.* CRC Press, Boca Raton, Florida, 2011.