Rose-Hulman Undergraduate Mathematics Journal

Volume 15
Issue 2

Article 1

Efficiency of Lossless Compression of a Binary Tree via its Minimal Directed Acyclic Graph Representation

Mayfawny Bergmann
Purdue University

Rose-
Hulman
Undergraduate
Mathematics
Journal

Efficiency of lossless compression of a binary tree via its minimal directed acyclic graph representation

Mayfawny Bergmann[a]

Volume 15, No. 2, Fall 2014

Sponsored by

Rose-Hulman Institute of Technology
Department of Mathematics
Terre Haute, IN 47803
Email: mathjournal@rose-hulman.edu
http://www.rose-hulman.edu/mathjournal

[a]Purdue University

Rose-Hulman Undergraduate Mathematics Journal
Volume 15, No. 2, Fall 2014

Efficiency of lossless compression of a binary tree via its minimal directed acyclic graph representation

Mayfawny Bergmann

Abstract. We consider the minimal directed acyclyc graph (DAG) lossless compression strategy introduced in Kieffer et. al [3], with the aim of testing its asymptotic effectiveness on binary trees of size n. We have four models for studying the compression strategy: two ways of measuring size (either the number of leaves or the depth of the tree), and two types of probability distributions (all planar trees are equally likely, or all nonplanar trees are equally likely). We calculate the average compression achieved by Kieffer et. al's strategy for some specific example classes of binary trees, and then more generally, averaging over all (either nonplanar, or planar) binary trees of a fixed size n. We use the results to draw conclusions about the kinds of trees for which the strategy is effective. An ultimate goal is to determine the extent to which the size of the DAG is correlated with the information embodied in the associated tree.

Acknowledgements: The author extends a heartfelt thank you to Dr. Mark Daniel Ward for his help and guidance in this project.

We heartily thank Jie Zhang, En-hui Yang, and John C. Kieffer for the paper that inspired this investigation. We also appreciate Wojciech Szpankowski bringing this paper to our attention.

This work is supported by the Center for Science of Information (CSoI), an NSF Science and Technology Center, under grant agreement CCF-0939370. We would like to thank the Center for their support. We also heartily thank Doug Crabill (Senior Academic IT Specialist, Department of Statistics at Purdue University) for providing computational facilities to make this work possible.

1 Introduction

In their ISIT 2013 paper [3], Kieffer et. al study an algorithm that losslessly compresses a *planar binary tree* into a directed acyclic graph (DAG). In the present paper, we will examine the effectiveness of this compression strategy, in several ways. We study the extent to which this algorithm can effectively compress binary trees as their sizes increase. (We can view the size of a tree as either its number of leaves or its depth, depending on the situation at hand.) Using the ratio of the number of nodes in the DAG versus the number of leaves in the tree (Kieffer et. al refer to this ratio as $r(t)$), we make some observations about the amount of information stored in different kinds of binary trees. Since we only deal with binary trees, we will use "tree" throughout to mean a binary tree, that is, a rooted tree for which each node has either 0 or 2 children.

For consistency, throughout the paper, we follow the notation from Keiffer et. al [3]. However, Keiffer et. al only consider planar trees, and we consider *nonplanar trees* as well. In planar trees, the order of the children matters, while in nonplanar trees the order does not matter.

A DAG representation of a binary tree t has two outgoing edges from each non-leaf vertex whose root to leaf paths are in one-to-one correspondence with the root to leaf paths of t. Kieffer et. al chose to label these edges to distinguish the left and right children, so that planar binary trees could be losslessly compressed. In the present paper, we chose *not* to label these edges in the DAG, in order to compress nonplanar trees as well as planar. Our algorithm compresses nonplanar trees in a lossless way, and planar trees in a lossy way.

The compression algorithm assigns a unique DAG representation $D(t)$ to each nonplanar tree t. The DAG representation consists of exactly one vertex for each distinct kind of subtree in t. By "subtree", we mean throughout a *final* subtree, that is, one for which every leaf is also a leaf of the tree it came from. Thus, if $|D(t)|$ denotes the number of vertices in $D(t)$, then $|D(t)|$ is equal to *the number of distinct nonplanar subtrees in* t.

A tree t and its minimal DAG representation $D(t)$ are shown in Figure 1. Each node is numbered according to the type of distinct nonplanar subtree for which it is the root. That is, the roots of all identical nonplanar subtrees share a common number. In the minimal DAG representation we see that each distinct nonplanar subtree occurs only once, and that here $|D(t)| = 5$.

To measure how well $D(t)$ compresses t, and in turn how much information t holds, Kieffer et. al introduce a value $r(t)$, defined as follows:

$$r(t) = \frac{|D(t)|}{n}, \qquad 0 \leq r(t) \leq 1,$$

where n is equal to the number of leaves in t (see [3]). For the tree in Figure 5, since the tree has 7 leaves and the associated DAG has 5 nodes, then $r(t) = 5/7$. The overarching motivation is that, if $r(t)$ is small, then a minimal DAG representation can be viewed as compressing t efficiently. Furthermore, $r(t)$ can be used as a measure of the information contained in the tree: a smaller $r(t)$ generally indicates that the tree holds less information than a tree with a larger $r(t)$.

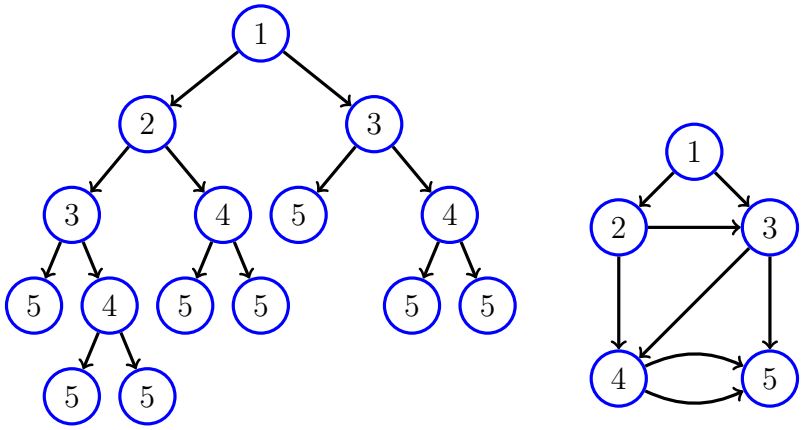

Figure 1: A tree and its minimal DAG representation

We can also compute the average $r(t)$ for a given subset F of trees, denoted $r(F)$ by the following computation:

$$r(F) = \sum_{t \in F} r(t)P(t),$$

where $P(t)$ is a probability distribution of t on F (see [3]).

Finally, if we let F_n be a subset of trees of size n, we can compute the limit of $r(F_n)$

$$\lim_{n\to\infty} \sum_{t \in F_n} r(t)P(t).$$

It already been shown that, if this limit is 0, and a certain requirement about $P(t)$ is fulfilled, then the compression is asymptotically optimal (see Kieffer, Yang, and Zhang [3]). In the specific case of planar trees with n leaves, which are sampled uniformly at random, Flajolet, Sipala, and Steyaert [2] have shown that,

$$\lim_{n\to\infty} \sum_{t \in F_n} r(t)P(t) = 2\sqrt{\frac{\ln(4)}{\pi \ln(n)}}\Big(1 + O\Big(\frac{1}{\ln(n)}\Big)\Big).$$

For the sake of completeness and symmetry among the various cases shown in our presentation, we include even this well-known case (planar trees, sampled uniformly at random) in our derivations, for comparison with the cases that are much less studied.

In this paper we will investigate $r(F_n)$ for increasing values of n. This will help us determine how well a minimal DAG representation compresses different types of trees, and whether the compression is asymptotically optimal. First we will examine $r(F_n)$ for some specific types of binary trees of size n, and then we will examine it for general binary trees of size n. We will measure n in two different ways: the number of *leaves* in t, and the *depth* of t. (Throughout this paper, depth refers to the length of the longest root-to-leaf path in a tree.) We will use two probability distributions for each of these measuring techniques: one distribution in which all *nonplanar* trees of size n are equally likely, and one in which all *planar* trees of size n are equally likely.

2 Probability Distributions

2.1 Number of Trees of Size n

Before detailing the setup of the probability distributions to be used, it is necessary to provide formulas for the number of trees of size n. This can be counted in the four different ways listed below. We use the multiset (MSET) notation from [1], meaning an unordered collection (i.e., just a set) that allows for elements to be repeated.

2.1.1 Number of nonplanar trees with n leaves

Each tree in the set $\mathcal{H}$ of all nonplanar binary trees is composed of a root node which is either a leaf (having no children), denoted by ε, or which has two children, each of which are themselves trees. (Here we use $(\text{MSET}_2)$to mean a multiset with two elements.) Thus, $\mathcal{H}$ can be expressed

$$\mathcal{H} = \mathcal{Z} \times (\text{MSET}_2(\mathcal{H}) + \varepsilon),$$

where $\mathcal{Z}$ marks the total number of vertices in the tree. If $\mathcal{H}_n$ denotes the set of nonplanar trees of size n, and $h_n := |\mathcal{H}_n|$ denotes the number of such trees, and $H(z) = \sum_n h_n z^n$ is the associated ordinary generating function, then

$$H(z) = z\left(\frac{(H(z))^2}{2} + \frac{H(z^2)}{2} + 1\right).$$

We can use bootstrapping to expand this recursive equation for the sequence of the counts of nonplanar binary trees with n leaves. This sequence is

$$(h_n)_{n\geq 0} = 1, 1, 1, 2, 3, 6, 11, 23, 46, 98, 207, 451, 983, 2179, 4850, \ldots.$$

See also #A001190 in [4].

2.1.2 Number of planar trees with n leaves

It is well known that this can be determined by the Catalan numbers. If C_n denotes the nth Catalan number

$$C_n = \frac{2n!}{(n+1)!n!}.$$

This familiar Catalan sequence is

$$(C_n)_{n\geq 0} = 1, 1, 2, 5, 14, 42, 132, 429, 1430, 4862, \ldots.$$

See also #A000108 in [4].

2.1.3 Number of nonplanar trees of depth n

Each binary tree of depth n is composed of a root node and either two subtrees of depth $n-1$ or one subtree of depth $n-1$ and another of depth less than $n-1$. If d_n denotes the number of binary trees of depth n, then there are $d_{n-1} + \frac{(d_{n-1})(d_{n-1}-1)}{2} = \frac{d_{n-1}(d_{n-1}+1)}{2}$ trees of the first type and $d_{n-1}\sum_{i=0}^{n-2} d_i$ trees of the second type. By convention, we have $d_0 = 1$. The recurrence yields

$$d_n = \frac{(d_{n-1})(d_{n-1}+1)}{2} + d_{n-1}\sum_{i=0}^{n-2} d_i.$$

The first few counts in this sequence are

$$(d_n)_{n\geq 0} = 1, 1, 2, 7, 56, 2212, 2595782, 3374959180831, \ldots.$$

See also #A103410 in [4].

2.1.4 Number of planar trees of depth n

Again, each binary tree of depth n is composed of a root node and either two subtrees of depth $n-1$ or one subtree of depth $n-1$ and another of depth less than $n-1$. (Since we use d's in both the nonplanar and planar contexts, we will always be sure to indicate which context we are working in.) Because the order of the children must be taken into account when working with planar trees, there are d_{n-1}^2 subtrees of the first type and $2d_{n-1}\sum_{i=0}^{n-2} d_i$ subtrees of the second type. As before, we use $d_0 = 1$, and it follows that, for $n \geq 1$, we have

$$d_n = d_{n-1}^2 + 2d_{n-1}\sum_{i=0}^{n-2} d_i.$$

The first few counts in this sequence are

$$(d_n)_{n\geq 0} = 1, 1, 3, 21, 651, 657653, 210065930571, \ldots.$$

See also #A001699 in [4].

2.2 Sigma-based Probabilities

The following notation is taken directly from Kieffer et. al [3]. Let t_L denote the subtree of t rooted at the left vertex directly below the root of t, and let t_R denote the one rooted at the right vertex. Let $\sigma(i,j)$ denote the probability that a tree t of size $i+j$ (where size can either be determined by the number of leaves or the depth) has the property $|t_L| = i$ and $|t_R| = j$ (see [3]). Thus,

$$\sum_{i+j=n} \sigma(i,j) = 1,$$

for all n (see [3]). Now, let V_t be the set of all non-leaf vertices v in t, and let $t(v)$ be the subtree in t rooted at v (see [3]). To find $P(t)$, the probability of a particular tree t among all trees of size $|t|$, we multiply together the sigmas corresponding to all vertices in t (see [3]):

$$P(t) = \prod_{v \in V_t} \sigma(|t(v)_L|, |t(v)_R|).$$

For the purposes of this paper, we will be using probability distributions in which all trees of size n are equally likely. The sigma based probability distribution can be used for cases when all *planar* trees of size n (counted either by leaves or by depth) are equally likely. The sigmas that will yield such a distribution can be calculated as follows:

$$\sigma(i, j) = \frac{N(i)N(j)}{N(i+j)},$$

where $N(n)$ denotes the number of trees of size n. This is easily understood by noting that the numerator is the number of trees for which $|t_L| = i$ and $|t_R| = j$ and the denominator is the total number of trees of size n.

3 Compression Rates for Two Extreme Types of Trees

Recall from the introduction that the compression rate $r(t)$ is equal to $|D(t)|$ divided by the number of leaves in t. It was mentioned that if $r(t)$ is close to zero, then a minimal DAG representation can be viewed as an efficient compression of t, and t holds relatively little information. To get a feel for the way $r(t)$ behaves, we will first look at $r(t)$ values as size increases for some basic types of trees.

3.1 Full Trees

By "full tree" we signify a tree that has the maximum possible number of leaves for its depth, that is, 2^d leaves (see Figure 4). Recall that to find $|D(t)|$ for a tree, we must count the number of distinct nonplanar subtrees in the tree. For a full tree t, it is clear that every subtree whose root is the same distance from the root of t is identical. In other words, there is only one distinct subtree per level of t. Thus, if d is the depth of t, $|D(t)| = d + 1$. So for a full tree,

$$r(t) = \frac{d+1}{2^d}.$$

Now let us consider a tree of depth d with $2^d - 1$ leaves, that is, a full binary of depth at least 2, with a pair of leaves clipped off leaving a single leaf l in its place (see Figure 2). Every vertex on the path between l and the root of t (not including l or the root of t) is the root of a new subtree distinct from those in the original full subtree. Thus, there are now

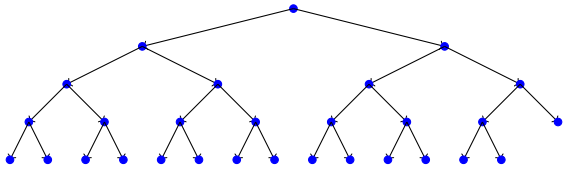

Figure 2: Almost full tree.

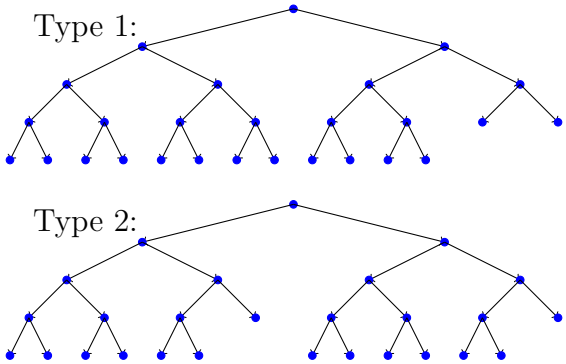

Figure 3: Almost-almost full trees.

$d-1+d-2 = 2d-1$ total distinct nonplanar subtrees. So for a full tree minus a pair of leaves,

$$r(t) = \frac{2d-1}{2^d-1}.$$

For a tree with yet another pair of leaves clipped off (see Figure 3), there is more than one possible expression for $r(t)$. If the clipped off pairs of leaves are on the same half of the tree (that is, if their root-to-leaf paths converge before the root node) then $|D(t)| = 2d-1$ as above (Type 1). If not, $|D(t)| = 2d-2$ (Type 2). Assuming all nonplanar trees are equally likely, the probability that they are on the same half of the tree is $\frac{d-2}{d-1}$. Thus, for a full tree minus two pairs of subtrees,

$$r(t) = \frac{(2d-1)(d-2)}{(2^d-2)(d-1)} + \frac{(2d-2)}{(2^d-2)(d-1)}.$$

Figure 5 displays the plot of all three types of full or almost full trees. Note that for these trees, $r(t) \to 0$ as $n \to \infty$. This indicates that a minimal DAG representation seems to be an efficient compression of full or nearly full trees. It also indicates that there is relatively little information stored in such trees.

3.2 Single-string Trees

Let "single-string tree" refer to a tree with the minimum number of leaves for its depth, that is, $d+1$ leaves. (In [3] this is kind of tree is referred to as a one-dimensional tree.) Each of

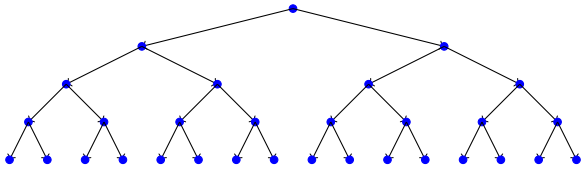

Figure 4: A full tree of depth 4.

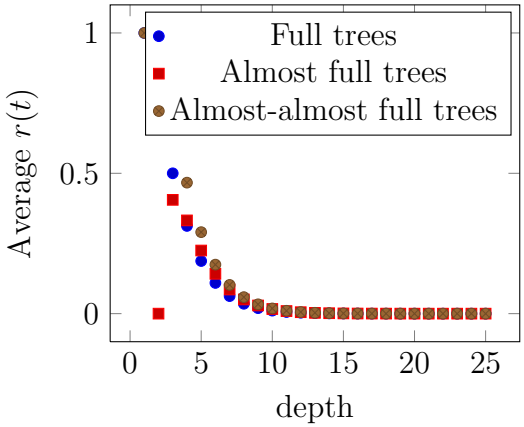

Figure 5: Average $r(t)$ for Full Trees

the d non-leaf vertices in a single-string tree is the root of a distinct subtree, and the leaves contribute a final distinct subtree. Thus $|D(t)| = d + 1$, so for a single-string tree,

$$r(t) = 1.$$

For a single-string tree with an extra pair of leaves, that is, with $d + 2$ total leaves (see Figure 6), the addition of the pair does not add to the number of distinct subtrees. Thus, $|D(t)|$ remains at $d + 1$ and for a minimal tree with one pair added,

$$r(t) = \frac{d+1}{d+2}.$$

Finally, let us consider a single-string tree with two extra pairs of leaves, or with $d + 3$ total leaves (see Figure 7). These extra pairs also do not add to the number of distinct subtrees,

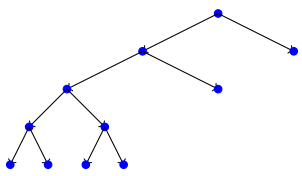

Figure 6: Almost single-string tree.

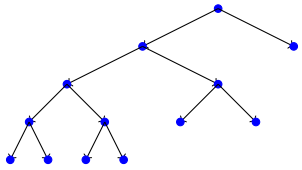

Figure 7: Almost-almost single-string tree.

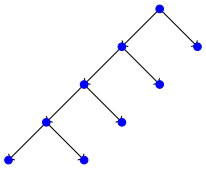

Figure 8: A single-string tree of depth 4.

so $|D(t)|$ remains at $d+1$. For a single-string tree with two pairs added,

$$r(t) = \frac{d+1}{d+3}.$$

The plots of the $r(t)$ of all three types of single-string or close to single-string trees are in Figure 9. Notice that for these trees, $r(t)$ approaches 1 as n gets large. In most cases, a large $r(t)$ value would indicate a rich, complex tree holding a lot of information, but this case provides a clear counterexample. Single string trees are in fact very simple trees, but they are compressed very inefficiently by a minimal DAG representation.

This counterexample is important to note. It shows that a minimal DAG representation is not always an efficient way of compressing trees, and thus $r(t)$ is not a perfect indicator of the complexity of t. It seems that the closer a tree gets to becoming a single-string tree, the less efficient its minimal DAG compression will be. This is important to consider when using $r(t)$ values to evaluate the richness of the structure of trees.

4 $r(t)$ for General Trees

Now we will calculate the average $r(t)$ for *general* binary trees of size n as n increases. Before calculating the individual $r(t)$'s, we must establish a method of recursively generating all trees of size n. Then, in order to know how to properly weight the $r(t)$'s for the cases when all planar trees are equally likely, we must establish a method for counting the planar trees that correspond to a given nonplanar tree. Finally, we can calculate $r(t)$'s for the individual trees of size n and average them.

4.1 Recursively Generating Nonplanar Binary Trees of Size n

Recall that t_L and t_R refer to the subtrees rooted at the left and right vertices directly below the root of t (see [3]). Let i and j be the sizes of t_L and t_R, respectively (see [3]). We will

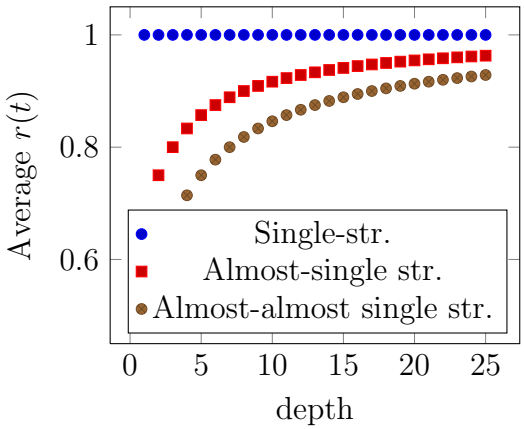

Figure 9: Average $r(t)$ for single-string Trees

measure size first by the number of leaves, and then by depth.

Let us first consider the case where n is the number of leaves in t. In this case,

$$i + j = n,$$

for every tree of size n. To find all nonplanar trees with n leaves, we go through all possible combinations of i, j such that $i \leq j$ and $i + j = n$. For each i, j, we go through all trees of sizes i and j to form all possible corresponding t_L, t_R combinations. For the cases where $i = j$, we must be careful not to double count the nonplanar trees.

Now for the case where size n is the depth of t. In this case,

$$\max(i, j) = n - 1,$$

for every tree of size n. For simplicity, let $i = n - 1$, and let $j \leq i$. We go through all possible j's, and for each j we go through all trees of sizes i and j to form all possible t_L, t_R combinations. Again, for the cases where $i = j$, we are careful not to double count the nonplanar trees.

4.2 Counting the Planar Trees Corresponding to a Nonplanar Tree

For every node in a nonplanar tree t whose two direct subtrees are *different*, there are two possible orders those direct subtrees could be in, each forming a distinct planar tree. Thus, our strategy will be to first count *the number of nodes in the tree t whose two direct subtrees are different*, and call this number $d(t)$.

To find $d(t)$ recursively, we first determine whether t_L and t_R are congruent. If they are congruent, then the root node does not contribute to $d(t)$ and

$$d(t) = d(t_L) + d(t_R).$$

If t_L is not congruent to t_R, then the root node contributes 1 to $d(t)$ and

$$d(t) = d(t_L) + d(t_R) + 1.$$

Once the number of nodes whose direct subtrees are different is found, then $2^{d(t)}$ is the number of planar trees corresponding to t.

4.3 Calculating Average $r(t)$

Recall that $r(t) = |D(t)|/n$ [3], where $|D(t)|$ is equal to the number of distinct nonplanar subtrees in t. As we generate the trees in section IV-A, we must keep for each tree a list of all the distinct nonplanar subtrees within that tree. To form this list for a tree t, we combine this lists of subtrees of t_L and t_R and eliminate any repeats. Finally, we add t itself to the list, and count the members of the list to find $|D(t)|$. Thus, if $R(t)$ denotes the number of repeated subtrees from the lists for t_L and t_R, then

$$|D(t)| = |D(t_L)| + |D(t_R)| - R(t) + 1.$$

To find $r(t)$ we then simply divide $|D(t)|$ by the number of leaves. This strategy works for both the depth and leaves methods of counting size. The only slight difficulty is that when using the depth method, we must also store the number of leaves in each tree, which can easily be recursively calculated by summing the leaves in t_L and t_R.

Finally, we are ready to calculate the average $r(t)$ for all trees of size n, $r(F_n)$. If all nonplanar trees of size n are equally likely, then

$$r(F_n) = \frac{\sum_{|t|=n} r(t)}{NPL(n)},$$

where $NPL(n)$ denotes the number of nonplanar trees of size n, discussed in Section 2.1. Now let $L(t)$ denote the number of planar trees corresponding to a nonplanar tree t, discussed in section 4.2. We can use this equation if all planar trees are equally likely

$$r(F_n) = \frac{\sum_{|t|=n} r(t)L(t)}{PL(n)},$$

where $PL(n)$ denotes the number of planar trees of size n, also discussed in Section 2.1.

Figures 10 and 11 display the plots of the first few $r(F_n)$'s for different types of general binary trees. Notice that when size is counted by the number of leaves, the plots do not appear to approach zero. This suggests that number of leaves is not an excellent indicator of how well the single-string DAG representation compresses a tree, and the DAG method is not asymptotically optimal over trees with n leaves. It is also interesting to note the pattern in the plot of nonplanar trees with n leaves: there are bigger jumps in average $r(t)$ before even numbers of leaves than before odd numbers of leaves. When size is counted by depth, however, the plot approaches zero quickly, and it is likely that it will approach

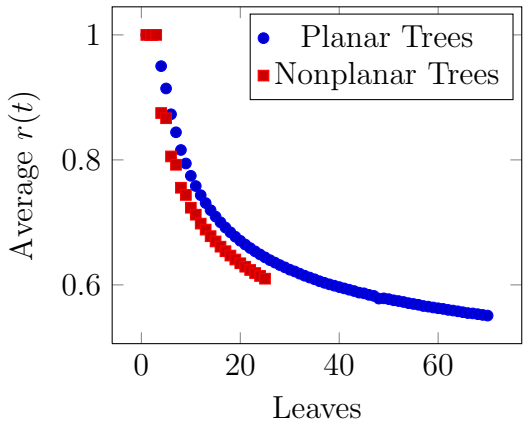

Figure 10: Average $r(t)$ for trees with n Leaves

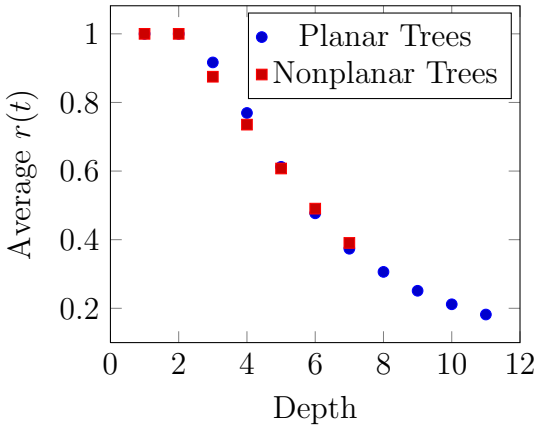

Figure 11: Average $r(t)$ for trees with Depth n

zero asymptotically. This suggests that depth is a good indicator of how well the single-string DAG method will compress a tree t, that is, deeper trees are generally more efficiently compressed than less deep trees, and it is possible that the DAG method is asymptotically optimal over trees of depth n.

Notice that the plots for planar trees extend farther than those for nonplanar trees. This is because the planar values are approximated in the cases $n \geq 26$ for leaves, and $n \geq 8$ for depth, by averaging over 100000 randomly generated planar trees of size n instead of all planar trees of size n. We can randomly generate planar trees, but not nonplanar trees, using the probability distribution outlined in section II-B.

After seeing a preliminary draft of this paper, Wojciech Szpankowski asked us about the average size of the largest subtree that is repeated in the tree. This is analogous to the well-studied question from data compression methods, "What is the average size of the largest repeated section in a string to be compressed?" In the case of trees, using our methodology, it takes very little extra work to calculate the average size of the largest repeated subtree in

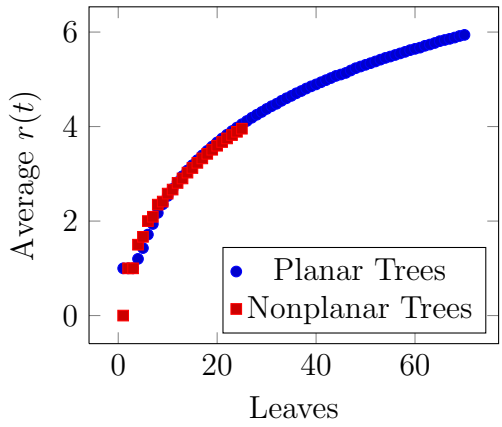

Figure 12: Average Largest Repeated Subtree for trees with n Leaves

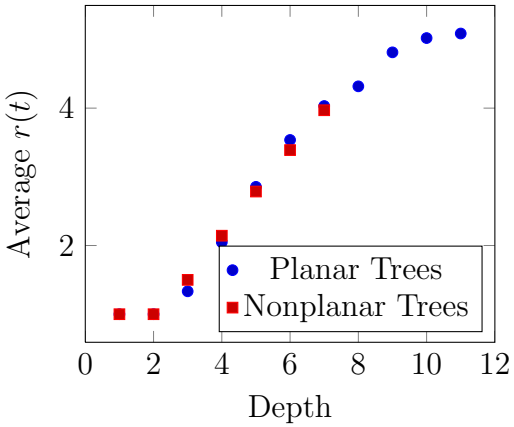

Figure 13: Average Largest Repeated Subtree for trees with Depth n

trees of size n. These calculations are displayed in plots in Figures 12 and 13 for size counted by leaves and depth, respectively. Again, the plots for planar trees are extended in the cases $n \geq 26$ for leaves and $n \geq 8$ for depth, using approximations of the probability distribution described in section II-B.

5 Open Questions

This paper barely scrapes the surface of the work that can be done to investigate the efficiency of the single-string DAG representation, and its implications about the amount of information stored in different kinds of trees.

One obvious question the paper leaves unanswered is, "What are the asymptotics of the four plots in section 4.3?" It is difficult to compute average $r(t)$ for large n's because the counts of trees of size n grow very rapidly. Knowing the asymptotics of the average $r(t)$ for general binary trees as n gets large would yield information about the efficiency of this

compression scheme and the amount of information stored in larger trees.

Another question to be further investigated is, "How does the 'fullness' of a tree relate to its $r(t)$ value?" In section 3 we saw that full trees and close-to-full trees have average $r(t)$ values that approach zero quite rapidly, while single-string trees and close-to-single-string trees have $r(t)$ values close to 1. This suggests that fullness may be closely related to $r(t)$ values, and that fuller trees are more efficiently compressed by a minimal DAG representation.

There are many different specific types of binary trees whose $r(t)$'s could be investigated as their sizes get bigger. For example "balanced binary trees" are commonly defined as trees for whom the leaves are all approximately the same distance from the root. One could also examine $r(t)$ for binary trees used in particular searching and sorting algorithms. And then of course one could investigate the $r(t)$ values for binary trees in which each node has 0, *1*, or 2 children, and for non-binary trees.

References

[1] Philippe Flajolet and Robert Sedgewick. *Analytic Combinatorics*. Cambridge, 2009.

[2] Philippe Flajolet, Paolo Sipala, and Jean-Marc Steyaert. Analytic variations on the common subexpression problem. In Mike S. Paterson, editor, *Proceedings of the 17th International Colloquium on Automata, Languages and Programming (ICALP 90)*, volume 443 of *Lecture Notes in Computer Science*, pages 220–234, Berlin/Heidelberg, 1990. Springer.

[3] J. C. Kieffer, E. Yang, and J. Zhang. Redundancy analysis in lossless compression of a binary tree via its minimal DAG representation. *International Symposium of Information Theory*, pages 1914–1916, 2013.

[4] N. J. A. Sloane. The on-line encyclopedia of integer sequences. Published electronically at `http://oeis.org/`.