Counting the Leaves of Trees [*]

Brian Cloteaux
National Institute of Standards and Technology
Applied and Computational Mathematics Division
Gaithersburg, Maryland, USA
brian.cloteaux@nist.gov

Luis Alejandro Valentin
Department of Computer Science
College of William and Mary
Williamsburg, Virginia, USA
lavalentin@wm.edu

Abstract

A number of important combinatorial counting problems can be reformulated into the problem of counting the number of leaf nodes on a tree. Since the basic leaf-counting problem is #*P*-complete, there is strong evidence that no polynomial time algorithm exists for this general problem. Thus, we propose a randomized approximation scheme for this problem, and then empirically compare its convergence rate with the classic method of Knuth. We then give an application of our scheme by introducing a new algorithm for estimating the number of bases of a matroid with an independence oracle.

1 Introduction

The importance of combinatorial counting problems stems from the number of application areas in which they are found. Counting problems appear in such diverse areas as bioinformatics, network reliability, and computational chemistry.

Unfortunately, many of these counting problems that we are interested in appear to be computationally intractable. More precisely, a number of these problems have been shown to be #*P*-hard. From its definition, solving a #*P*-hard problem in polynomial time implies $P = NP$. In fact, it can be shown that being able

[*]Official contribution of the National Institute of Standards and Technology; not subject to copyright in the United States.

to solve #P-hard problems in polynomial time collapses the entire polynomial-time hierarchy to P [15]. Thus, there is overwhelming evidence that for #P-hard problems, polynomial time algorithms cannot exist.

We also observe that many counting problems can be reformulated into instances of the problem of counting the number of leaves on some tree. In particular, if the combinatorial objects that are being counted have some structure that allows them to be sequentially built, we can construct a tree representation in which each node represents a partially constructed object and each edge represents the sequential addition of a substructure. The complete objects that we are interested in counting are the leaf nodes for this tree. Thus, being able to approximate the number of leaves for a tree gives us a powerful and general approach for many difficult enumeration problems. Since the basic problem of leaf-counting is #P-hard we investigate randomized methods for approximating the number of leaf nodes. We introduce a new randomized method and then show an application of this approach to a known #P-complete problem of counting the bases of a matroid.

2 Approximating the number of leaves

A straightforward method to count the number of leaves is by recursively summing over each of the subtrees. Thus, if $\#L(v)$ is the number of leaves on the tree rooted at the node v, then the total number of leaves in the tree is

$$\#L(v) = \begin{cases} 1 & v \text{ is a leaf node,} \\ \sum_{c \in child(v)} \#L(c) & \text{otherwise} \end{cases} \tag{1}$$

For a tree of depth n, it is easy to see that there is potentially an exponential number of leaf nodes. Thus, brute force counting requires exponential time. Further, since the basic problem is #P-hard, it is doubtful that any polynomial-time algorithm exists. In addition, it is thought that even polynomial-time deterministic approximations do not exist for this problem. Therefore, research into this problem has predominately focused on developing new randomized approximation techniques.

One of the first randomized algorithms was given by Knuth [11]. Knuth's interest in the problem stemmed from wanting to approximate the run-time of a recursive algorithm by estimating the size of its associated back-track tree. The basic idea behind his approach is to take a sample of the tree by starting at the root and randomly choosing a child node to explore until a leaf node is reached. By recording of the number of different choices available as we descend down the tree, we can multiply these values together to estimate the size of the tree. This approach is shown by the following recursive formulation.

$$K(v) = \begin{cases} 1 & v \text{ is a leaf node,} \\ degree(v) \cdot K(c) & \text{otherwise, for a random } c \in child(v) \end{cases} \tag{2}$$

Knuth proved that this approach returns the correct expected value of $E\left(K(v)\right) = \#L(v)$. The principle advantage of this algorithm is the simplicity of understanding and implementing it. Also, it often returns a reasonable estimate using only a few samples for short and wide trees [13]. Knuth noted that many of the trees arising from practical problems are amenable to this approach.

A problem with this approach is that its variance can be exponentially large. Knuth showed an exponential upper bound for the variance of the method, and Stockmeyer proved sampling on a tree with N nodes can require $\Omega(\sqrt{N})$ samples to converge within polynomial bounds [14]. This result implies that in the general case we need to take an exponential number of samples to guarantee a polynomial-bounded variance.

This difficulty particularly manifests itself when sampling tall and skinny trees [13]. Here, we see that an exponentially large number of samples are required in order to find the deep nodes in these types of trees. Without reaching the deepest nodes in a tree, Knuth's algorithm underestimates the number of leaves and so convergence will be exponentially slow. In order to improve the results from the basic Knuth algorithm several extensions have been suggested: importance sampling [11], partial backtracking [13], and stratified sampling [5].

In his original paper, Knuth noted that if we bias the sampling probabilities of the child nodes, we can potentially lower the variance. (You can reference Beichl and Sullivan [3] for a nice introduction to importance sampling.) In fact, he pointed out that there must exist an importance function that reduces the sampling variance to zero. Unfortunately, determining this function is at least as hard as the exact counting problem and so is also $\#P$-hard. Further, even finding an importance function that gives polynomial bounds on the estimate is computationally difficult. It has been shown that sampling using an importance function that is able to estimate relative subtree size to within some constant bound is still unable to guarantee subexponential variance [9]. Thus, the use of importance functions for Knuth-style sampling seems limited to very specific families of trees.

To overcome some of the difficulties in Knuth's sampling method, Purdum introduced the idea of allowing a certain amount of backtracking when encountering leaf nodes during a sample [13]. Although this idea improves the sampling results for many trees, it requires allowing an exponential amount of backtracking in the tree to guarantee polynomial bounds on the estimate it produces. Finally, Chen offered a stratified sampling approach to the problem [5]. His approach does reduce the variance but it requires some level of specific domain knowledge of the family of trees being analyzed.

Because of the difficulties with these techniques for extending Knuth's algorithm, we reexamine the problem in view of more recent results in randomized algorithms. In particular, we look at the breakthrough result by Aldous and Vazirani on how to randomly explore a tree, and then tie it back to the leaf-counting problem.

3 Finding deep leaves in a tree

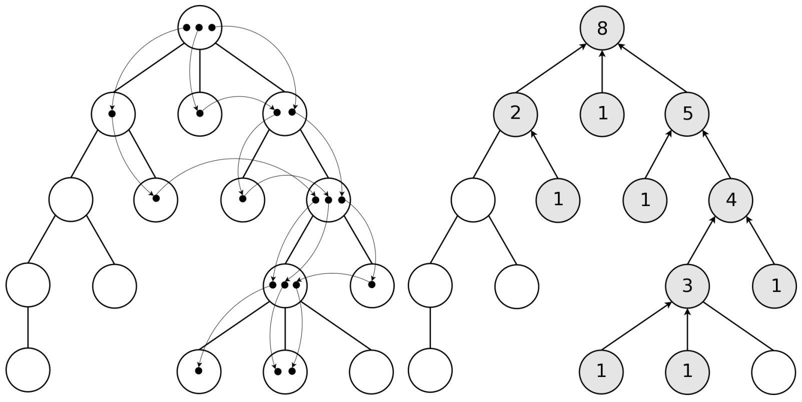
How can we improve Knuth's scheme? Let us put aside, for the moment, the problem of counting leaves in a tree, and instead look at the closely related problem of finding a deepest leaf node in a tree. Aldous and Vazirani gave a breakthrough result on this problem by publishing an algorithm that, using only a polynomial number of samples, has a high probability of finding a deepest leaf node [1]. They call their algorithm "Go with the winners" (GWTW).

To describe their algorithm, we use the terminology from their paper. We visualize a directed tree with a root node at the top. Into the root node, we release N number of *particles*. For each discrete time step, the particles move to a random child node if possible. If a particle is at a leaf node and there are other particles at non-leaf nodes, then the particle will randomly jump to one of the non-leaf nodes that contain a particle. Finally, if all particles are at leaf nodes, then the algorithm ends. The basic GWTW algorithm is:

1. Start with N particles on the root node
2. If all N particles are on leaf nodes, then we are done
3. For each particle currently on a leaf node, randomly move that particle to a non-leaf node which contains another particle
4. For all particles, move to a random child
5. Jump back to step 2

We can think of the GWTW algorithm as a parallel implementation of Knuth sampling. Since we are allowing interactions among the simultaneous trials, instead of relying on backtracking when we are stuck at a leaf node (like in Purdom's method), the algorithm uses knowledge about where non-leaf nodes exist that is obtained from other particles. An example of how three particles explore a tree using the GWTW approach is shown in Figure 1(a).

This algorithm was originally conceived as a model for analyzing the effectiveness of heuristic search strategies such a simulated annealing. As a practical algorithm though, using GWTW to find the deepest nodes in a tree has limited usefulness. While the number of particles needed to have a high probability of finding a deepest node is polynomial, the polynomial itself can be quite large [7]. For our purposes though, we are not interested in always finding the deepest nodes in tree, but rather in obtaining a near uniform sampling of the tree in order to provide a good size estimate.



(a) An example run of the GWTW algorithm (b) Leaf counting on the sampled subtree

Figure 1: An example of how the GWTW algorithm explores a tree. For the sampled subtree, it is then shown how the AV-K algorithm estimates the number of leaves in the tree

4 A generalized counting scheme based on subtree sampling

We now propose a new method for counting the leaf nodes of a tree. This algorithm is composed of two steps. First, using a polynomial number of particles, we explore the tree T using the GWTW algorithm. We create a new tree S by recording each node that a particle visits in T along with number of children the node in T has. This will give us a polynomial sized subtree of the original tree.

For the second step, we recursively traverse the tree S to estimate the size of the original tree. For a node v, an estimate is made by combining the estimates for the explored subtrees of v with averages of those values for the unvisited subtrees. In other words, using the tree S, we approximate the number of leaf nodes in T using the following estimator G where $child_P(v)$ is the set of children nodes of v in the tree P.

$$G(v) = \begin{cases} 1 & v \text{ is a leaf node,} \\ \frac{|child_T(v)|}{|child_S(v)|} \sum_{c \in child_S(v)} G(c) & \text{otherwise} \end{cases} \tag{3}$$

For $O(n^c)$ particles where n is the tree depth and c is a constant, then the size of the tree S will be $O(n^{c+1})$ nodes. Thus, this algorithm runs in polynomial time in the depth of the tree T.

To see that the algorithm converges to the number of leaves in the tree, we observe that whenever a particle reaches a leaf node, a Knuth estimate is made of the number of leaves in the subtree back to the last node which shared multiple particles. Since the expected value of Knuth sampling is the number of leaves in the subtree [11], then from the linearity of expected values we can combine estimates for an expected value of the number of leaves at the root of the subtree. By inductively applying these estimates up the tree, we see the expected value at the root is the number of leaves in the tree.

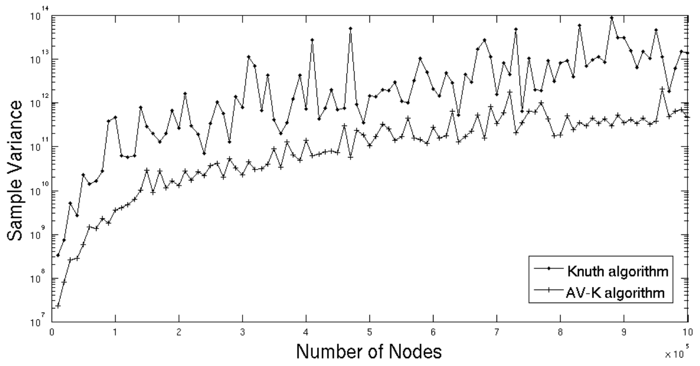
Since this algorithm combines the Knuth estimator with the tree exploration algorithm of Aldous and Vazirani, we denote it as the *AV-K* approach. An example run of this method is shown in Figure 1(b). The shaded nodes represent the subtree created from GWTW exploration shown in Figure 1(a). The labels on the nodes show the estimate of the number of leaf nodes given by the algorithm for the subtree rooted at that node.

5 Comparing the algorithms

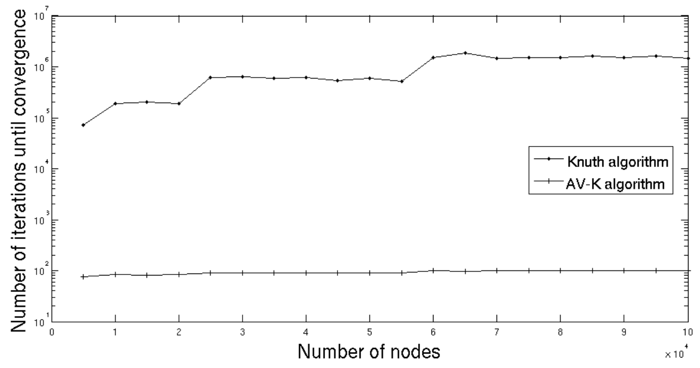
In the case of one particle, we get identical estimates between the two methods. As we will show, for multiple particles the AV-K method always has a smaller variance. Even if the particles do not share information during the sampling (such as when all leaves are on the same level of the tree), the AV-K method still gives an improved estimate over the simple Knuth method. Since the algorithm combines Knuth estimates over smaller subtrees, this reduces the growth in variance over Knuth's basic method. While this shows that the AV-K method has a lower variance, it is still open whether the AV-K approach allows us to estimate the number of leaves with polynomially bounded variance.

Since analytically comparing the average-case behavior of the two algorithms is extremely difficult, we opted to compare the rate of convergence between the AV-K method and Knuth's method through a series of computational experiments. Our experiments consisted of generating a set of random trees [17] with logarithmically bounded depth. We tested how many particles (or samples) were necessary to estimate the number of leaf nodes to within one percent. We also tested the sample variance between the two methods by running each method multiple times on the same random graph and recording the average sample variance. In Figure 2(a), we give the results of our experiments.

From our experiments, we see that the AV-K method shows a pronounced reduction in the sample variance when compared to Knuth's method. In our tests, the difference in variance between the two methods was between one and two orders of magnitude. This difference is magnified when examining the number of samples need for convergence to within one percent of the actual number of leaves. For the relatively small trees of our tests, we see differences involving four orders of magnitude. This suggests that a significant reduction in computational cost is possible by computing over a GWTW sampled tree.



(a) Variance comparison



(b) Convergence comparison

Figure 2: Comparison of the sample variance and convergence rate of Knuth sampling versus the AV-K approach. In Figure 2(a), the average sample variance is shown. In Figure 2(b), we see the average number of particles or samples required for convergence to one percent of the true number of leaves. The values are recorded on a logarithmic scale.

6 Counting the bases of matroids

There are a number of combinatorial problems that can be reformulated into leaf-counting problems (for example, many of the problems in Welsh's survey [16]). We show one application of the AV-K method by introducing a new algorithm for approximating the number of bases of a matroid that is represented by an independence oracle. (For definitions and basic results concerning matroids, see Oxley [12].)

We define the matroid $\mathscr{M} = (G, I)$, where G is the ground set and I is the independence set. An *independence oracle* takes as input a set $S \in 2^G$ and returns whether true if $S \in I$ and false otherwise. If the set $S \in I$, then the oracle will also return a base of the matroid that contains S as a subset. This type of query to an independence oracle is called a *probe*.

While for certain classes of matroids, such as the graphical matroids, there does exist a polynomial-time algorithm for counting the number of bases (for graphical matroids see the first chapter of Jerrum's book [10]), in general this problem is #P-hard. In fact, it has been shown that even deterministic approximation schemes for estimating the number of bases have exponential bounds for a polynomial number of probes [2]. Thus, research into this problem has predominantly been in creating randomized approximation schemes for estimating the number of bases (such as [8, 4]). While it is an open problem whether a fully polynomial randomized approximation scheme (fpras) exists for all matroids, the most successful polynomial-time algorithm has been the sampling method of Chavez-Lomeli and Welsh [4], which has been shown to work for almost all matroids [6].

We introduce a new algorithm for the base-counting problem by representing the independence set of the matroid as a tree. The idea behind this construction is to define the directed and rooted tree $T = (N_T, E_T)$ where each node $N \in N_T$ has a mapping f to an independence set, i.e. $f : N_T \to I$. The root of the tree is mapped to the empty set. For each node N_i in T there is an edge to a child node N_j (i.e. $(N_i, N_j) \in E_T$) if and only if there exists an element $e \in E$ such that $e \in f(N_j)$ and $f(N_j) - \{e\} = f(N_i)$. The depth of this tree is precisely the rank, r, of the matroid $\mathscr{M}$ and each leaf node maps to a base in $\mathscr{M}$. It is easy to show inductively that the number of leaves in T is exactly $|B(\mathscr{M})| \cdot r!$ where $B(\mathscr{M})$ is the set of bases for $\mathscr{M}$.

Algorithm 1 is a GWTW-type algorithm for sampling the tree T and constructing a sampled tree S. By traversing the sampled tree S, we return an estimate of the value $|B(\mathscr{M})| \cdot r!$ Since the bases all have identical rank r, the leaves in the resulting tree are all on the same level. Thus, no particle jumping can occur between steps. Even though we are not able to take advantage of sharing knowledge between particles as the tree is sampled, the resulting sampling does provide a good approximation of the number of bases.

To gauge the quality of the results, we performed a set of computational ex-

```
**CreateTree**($\mathscr{M}$, $n_p$)
**Input**: $\mathscr{M}$–a matroid where $\mathscr{M} = (G,I)$, $n_p$–number of particles for
        sampling
**Output**: $S$–a sampled tree of the bases
Create tree $S = (\{N_\emptyset\}, \emptyset)$
**foreach** $P_i$ *where* $1 \leq i \leq n_p$ **do**
    $P_i \leftarrow \emptyset$
**end**
$C \leftarrow 1$
**while** $C < r$ **do**
    **foreach** $P_i$ *where* $1 \leq i \leq n_p$ **do**
        $Q_i = \{g \in G | P_i \cap \{g\} = \emptyset \wedge P_i \cup \{g\} \in I\}$
        Randomly choose $q \in Q_i$
        $T \leftarrow P_i \cup \{q\}$
        Create node $N_T$
        **if** $N_T \notin S$ **then**
            $S \leftarrow (N(S) \cup \{N_T\}, E(S) \cup \{(N_T, N_{P_i})\} )$
        **end**
        $C \leftarrow C + 1$
        $P_i \leftarrow T$
    **end**
**end**
**return** $S$
```
Algorithm 1: Algorithm for creating a tree for counting the number of bases of a matroid. We use the shorthand notation of N_A to designate a node where $f(N_A) = A$.

periments to compare Chavez-Lomeli and Welsh sampling to the AV-K approach. We generated a number of random graphical matroids in order to be able to efficiently compute the exact answer. We then compared the average number of samples needed for the Chavez-Lomeli and Walsh method to converge to within one percent of the exact answer for the number the bases to the number of particles needed for convergence of the AV-K method. Figure 3 shows the results of this experiment.

All matroids that were generated had a rank of 20 with varying sizes of the ground sets. For each point, ten different random matroids were created. Then for each matroid, both algorithms were applied for 30 separate instances and the number of samples or particles needed for convergence to within one percent of the correct answer were recorded. The points in the figure represent the average number of samples needed for convergence for all the random matroids of that particular size. Although we were forced to limit the size of our tests in order to make the computations tractable, the results do show two orders of magnitude difference on average in the number of samples needed between the two methods.

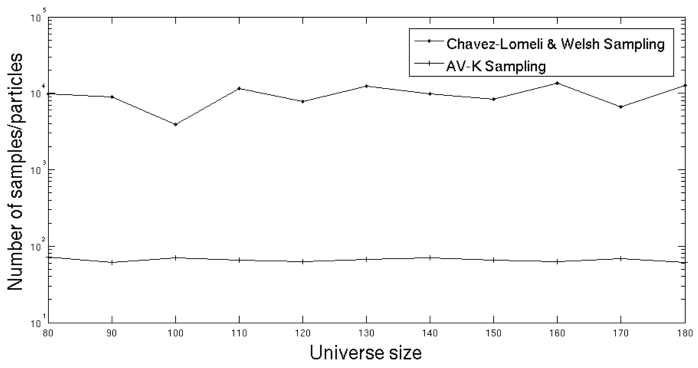


Figure 3: Comparison of the number of samples between the Chavez-Lomeli and Welsh algorithm and an AV-K algorithm needed to converge to within one percent of the actual answer for a number of random matroids.

7 Conclusion and future research

This research started out of a need for a practical method for attacking general combinatorial enumeration problems. By extending Knuth's sampling technique with the tree exploration algorithm of Aldous and Vazirani, we have created an algorithm which has empirically shown itself as a fast estimator for the number of leaves in a tree.

Future research involves establishing analytic bounds on the performance of the algorithm. In particular, we are interested in a bound on the number of particles versus the error bounds of the estimate.

References

[1] D. ALDOUS AND U. VAZIRANI, *"Go with the winners" algorithms*, in Foundations of Computer Science, 1994 Proceedings., 35th Annual Symposium on, 1994, pp. 492–501.

[2] Y. AZAR, A. Z. BRODER, AND A. M. FRIEZE, *On the problem of approximating the number of bases of a matroid*, Information Processing Letters, 50 (1994), pp. 9–11.

[3] I. BEICHL AND F. SULLIVAN, *The importance of importance sampling*, Computing in Science and Engineering, 1 (1999), pp. 71–73.

[4] L. CHAVEZ-LOMELI AND D. WELSH, *Randomised approximation of the number of bases*, Contemporary Mathematics, 197 (1996), pp. 371–376.

[5] P. C. CHEN, *Heuristic sampling: A method for predicting the performance of tree searching programs*, SIAM Journal on Computing, 21 (1992), pp. 295–315.

[6] B. CLOTEAUX, *Approximating the number of bases for almost all matroids*, Congressus Numerantium, 202 (2010), pp. 149–153.

[7] T. DIMITRIOU AND R. IMPAGLIAZZO, *Go with the winners for graph bisection*, in Proceedings of the ninth annual ACM-SIAM symposium on Discrete algorithms, SODA '98, Philadelphia, PA, USA, 1998, Society for Industrial and Applied Mathematics, pp. 510–520.

[8] T. FEDER AND M. MIHAIL, *Balanced matroids*, in STOC '92: Proceedings of the twenty-fourth annual ACM symposium on Theory of computing, New York, NY, USA, 1992, ACM, pp. 26–38.

[9] D. HARRIS. Personal communication, 2009.

[10] M. JERRUM, *Counting, sampling and integrating: algorithms and complexity*, Lectures in Mathematics ETH Zürich, Birkhäuser Verlag, Basel, 2003.

[11] D. E. KNUTH, *Estimating the efficiency of backtrack programs*, Mathematics of Computation, 29 (1975), pp. 121–136.

[12] J. G. OXLEY, *Matroid Theory*, Oxford University Press, New York, 2006.

[13] P. W. PURDOM, *Tree size by partial backtracking*, SIAM Journal on Computing, 7 (1978), pp. 481–491.

[14] L. STOCKMEYER, *On approximation algorithms for #P*, SIAM Journal on Computing, 14 (1985), pp. 849–861.

[15] S. TODA, *PP is as hard as the polynomial-time hierarchy*, SIAM Journal on Computing, 20 (1991), pp. 865–877.

[16] D. WELSH, *Some problems on approximate counting in graphs and matroids*, in Research Trends in Combinatorial Optimization, 2009, pp. 523–544.

[17] D. B. WILSON, *Generating random spanning trees more quickly than the cover time*, in STOC '96: Proceedings of the twenty-eighth annual ACM symposium on Theory of computing, New York, NY, USA, 1996, ACM, pp. 296–303.