Algorithms Seminar 2001–2002,
F. Chyzak (ed.), INRIA, (2003), pp. 19–22.

Available online at the URL
http://algo.inria.fr/seminars/.

A Relaxed Approach to Tree Generation

Philippe Duchon
Labri, Université de Bordeaux I (France)

November 5, 2001

Summary by Marni Mishna

Abstract

An algorithm for the uniform random generation of trees is described. The algorithm is notable for its simplicity and efficiency. These qualities stem largely from the fact that it does not precisely control the size of the final tree, rather, it is "relaxed." The complexity analysis yields that in certain cases the algorithm is linear. A family of variants with multiple parameters is also discussed.

1. Relaxed?

Efficient random tree generation is important in many contexts. Very often one does not specifically require trees of a particular size, but rather within a given size range. The idea at hand is to generate random trees of any possible size and reject those which are not in the given range. The generation is done in such a way to uniformly generate trees within a fixed size and to minimise the number of rejections. This simple algorithm is surprisingly efficient.

2. The Trees in Question

The trees are simple, as in the sense of Meir and Moon. That is, each family is linked to a set D of non-negative integers which dictates the possible number of children $f(s)$ a node s can have. As the trees are finite, zero is always contained in this set. More precisely, the set D corresponds to the family $\mathcal{T}_D$ of trees

$$\mathcal{T}_D = \{\, t \mid \forall s \in T, f(s) \in D \,\}.$$

Classic complete binary trees correspond to $\{0, 2\}$, for example. Other examples include 1–2 trees ($D = \{0, 1, 2\}$), general trees ($D = \mathbb{N}$) and linear? trees ($D = \{1, 0\}$).

They are simple in the sense that they easily admit a generating function decomposition. Consider $F_D(x) = \sum_n a_n x^n$ where a_n is the number of trees of size n. This can be rewritten

$$F(x) = \sum_{d \in D} xF(x)^d = \Phi_D\big(x, F(x)\big).$$

In the weighted model, the size is no longer the number of nodes, rather, a weighted sum where the weight depends on the degree of the node:

$$F(x) = \sum_{(d,w) \in D} x^{w(d)} F(x)^d = \Phi_D\big(x, F(x)\big),$$

where $w(d)$ is the weight of d.

2.1. **Restrictions on D.** We allow repetitions in the set. Further we impose a non-periodicity and a rationality requirement. These are well-described by M. Drmota in [1]. The essential characteristic guaranteed by these conditions is that F has a square root singularity. That is, $F(x_0 - h) = F_0 - C\sqrt{h} + O(h)$ and hence $a_n = C' x_0^{-n} n^{-3/2} \left(1 + O(n^{-1})\right)$. Given that F has a unique singularity x_0 of minimal modulus we set $F_0 = F(x_0) < \infty$. Hence, we have that (x_0, F_0) is a solution of $F_0 = \Phi_D(x_0, F_0)$, and further, $1 = \frac{\partial \Phi_D}{\partial F}(x_0, F_0)$.

3. Generation à la Galton–Watson

Galton–Watson trees (G.–W. trees) are formed recursively with respect to a given probability rule π. Suppose $\pi = (\pi_k)_{k\in\mathbb{N}}$ satisfies the three conditions $\sum \pi_i = 1$, $\pi_0 > 0$, $\pi_i \geq 0$. We form the G.–W. tree T_π recursively by the following method: beginning at the root node, determine the number of children k, with a random process with probability π_k, independently from the other vertices. If $k \neq 0$, recurse on each of the children. Notice that this does not a priori exclude infinite trees. However, a careful selection the probabilities can sufficiently increase the expected number of finite trees, as the following theorem indicates.

Theorem 1. *Let $m = \sum_k k\pi_k$.*

1. *If $m \leq 1$, then T_π is finite with probability 1.*
2. *In particular, if $m = 1$, The size of T_π is not integrable; the average size is infinite. We say the tree is* G.–W. critical.
3. *If $m < 1$ the expected size is $\mathbf{E}\big[|T_\pi|\big] = \frac{1}{1-m}$. We say the tree is* sub-critical.
4. *Otherwise, the tree is of infinite size with a strictly positive probability and we say it is* G.–W. sur-critical.

Example. Let us take a look at what happens in the case of binary trees. Let $\pi = (p, 0, 1-p, 0, \dots)$, $p > 0$. Suppose t is such a tree of size $2n-1$ nodes. Then there are exactly n internal nodes and $n-1$ leaves. Consequently,

$$\mathbf{P}(T_\pi = t) = \prod_{s\in t} \mathbf{P}\big(X = f(s)\big) = \prod_{s\in t} \pi_{f(s)} = p^{n+1}(1-p)^n,$$

which depends only on the size of t.

Example. The binary trees seem perhaps a special case. What can we say about 1–2 trees? Let $\pi = (\alpha, \beta, \gamma, 0, \dots)$, and let $N_i(t)$ be the number of nodes of t with degree i. Then,

$$\mathbf{P}(T_\pi = t) = \prod_{s\in t} \mathbf{P}\big(X = f(s)\big) = \alpha^{N_0(t)} \beta^{N_1(t)} \gamma^{N_2(t)}.$$

The probability depends on number of vertices with each type of degree. However, if we set $\alpha = \beta = \gamma = 1/3$, then the value $\mathbf{P}(T_\pi = t) = 1/3^{|t|}$ depends only on the size of t.

4. Probability that $T_\pi = t$?

The previous examples give us the intuition to answer the following question: under which conditions does the probability rely only on the size of the object?

An answer is found in Theorem 2, but first we motivate it with some observations. Notice that $|t| = 1 + \sum_{d\in D} dN_d(t)$. We choose some $0 < u < 1$ and let π_d be proportional to u^d. Then we get

$$\mathbf{P}(t) = \prod_{d\in D} \left(\frac{u^d}{\Phi_D(u)}\right)^{N_d(t)} = \frac{u^{\sum_d dN_d(t)}}{\Phi_D(u)^{\sum_d N_d(t)}} = \frac{u^{|t|-1}}{\Phi_D(u)^{|t|}} = \frac{1}{u}\left(\frac{u}{\Phi_D(u)}\right)^{|t|},$$

which only depends on $|t|$.

In the weighted case, we say that the size is $|t| = \sum_d w(d)N_d(t)$. We assign to $N(t) = \sum_d N_d(t)$, the number of vertices. In this case, if we select $x,\ u > 0$ and set $\pi_d = x^{w(d)}u^d/\Phi_D(x,u)$, we see that

$$\mathbf{P}(t) = \prod_{d\in D}\left(\frac{x^{w(d)}u^d}{\Phi_D(x,u)}\right)^{N_d(t)} = \frac{x^{|t|}}{u}\left(\frac{u}{\Phi_D(x,u)}\right)^{N(t)}.$$

Thus, if are principally interested by the weighted size, we can generate them uniformly with a careful selection of (x,u).

Theorem 2. *Let $\mathcal{T}_D$ be a simple labelled family of trees for which the generating series is $F(x) = \Phi_D(x,F(x))$. Further suppose that (x,u) is a couple satisfying $u = \Phi_D(x,u)$. Then the G.–W. tree defined by the generation law*

$$\pi_d = \frac{x^{w(d)}u^d}{\Phi_D(x,u)}$$

conditioned to be of weighted size n is a tree of $\mathcal{T}_D$, uniformly generated among those of weighted size n.

5. **Relaxed Random Generation**

Theorem 2 can be exploited for our random generation purposes. It implies the existence of a generation scheme where all trees of a given size are equally probable. This notion converts directly into an algorithm. But first recall the novelty here was to consider a more general situation. Instead of requiring a tree of size *exactly* n we consider an acceptable range. We will examine three ranges $[n_1,n_2]$ here: strict ($n_2 = n_1$), linear ($n_2 = (1+\lambda)n_1$) and geometric ($n_2 = n_1 + \lambda n_1^\alpha$). Consider the following algorithm which seemingly does the most naive thing.

Algorithm. Input: $\Phi_D(x,F)$, $[\,n_1,n_2\,]$.

1. Determine a couple (x,u) satisfying $u = \Phi_D(x,u)$;
2. Generate a G.–W. tree t with the generation rule $\pi_d = x^{w(d)}u^{d-1}$ until the process stops naturally, or until the total size is greater than n_2;
3. If the size of $t \notin [\,n_1,n_2\,]$ reject t and go to 2. Otherwise, output t.

Some questions must be answered here. For example, how simple is it to assure the non-periodicity of Φ_D? How do we solve for u? How many digits would be required in a numerical approximation to avoid bias?

5.1. **Complexity.** For the analysis purposes assume that determining (x,u) is inconsequential to the complexity. We evaluate:

- $E_< = \mathbf{E}\big[|T| \;/\; |T| < n_1\big]$ (average size of rejected small tree),
- $P_< = \mathbf{P}\big(|T| < n_1\big)$ (too small),
- $P_> = \mathbf{P}\big(n_2 < |T|\big)$ (too big),
- $P_= = \mathbf{P}\big(n_1 \le |T| \le n_2\big)$ (just right).

The average number of rejections for being too large is $\frac{P_>}{P_=}$. The average number of rejections for being too small is $\frac{P_<}{P_=}$. Thus, the average cost is at most

$$n_2 + n_2\frac{P_>}{P_=} + E_<\frac{P_<}{P_=}.$$

Consider now a quick calculation of $P_=$ for these three ranges.

The strict case is classic: $\mathbf{P}\big(|T| = n_1\big) = \Theta(n_1^{-3/2})$.

The linear case yields:

$$\sum_{n \le k \le (1+\lambda)n_1} \mathbf{P}\big(|T| = n_1\big) \sim n_1^{3/2} \sum_{0 \le k \le \lambda n_1} C(1 + k/n_1)^{-3/2} = \Theta(n_1^{-1/2}).$$

The geometric case yields:

$$P_= = \sum_{n_1 \le k \le n_1 + \lambda n_1^\alpha} \mathbf{P}\big(|T| = k\big) \sim \Theta(n_1^{-3/2}) \sum_{0 \le k \le \lambda n_1^\alpha} (1 + k/n_1)^{-3/2} = C n_1^{-3/2} \lambda n_1^\alpha \Theta(1) = \Theta(n_1^{\alpha - 3/2}).$$

We can summarise the average cost for our three range types:

Case	$P_=$	Average cost
$n_2 = n_1$	$\Theta(n^{-3/2})$	$n_1 + \frac{\Theta(n_1^{1/2})}{\Theta(n_1^{-3/2})} = \Theta(n_1^2)$
$n_2 = (1+\lambda)n_1$	$\Theta(n_1^{-1/2})$	$\Theta(n_1)$
$n_2 = n_1 + \lambda n_1^\alpha$	$\Theta(n_1^{\alpha - 3/2})$	$\Theta(n_1^{2-\alpha})$

Most notably, in the linear case, the algorithm is linear in n_1. This is quite efficient.

6. **The Multivariate Case**

We can extend the allowable families of trees by looking at G.–W. trees with k types of vertices. Now we have k probability laws and for each vertex the probability of a vertex of type i to have d_j children of type j is $\pi_{i,d}$, where d indicates the collection of d_j and the probability is independent of all others, except for its ancestors. It is less clear how to verify the desired properties such as periodicity and rationality.

Theorem 3. *Let $(x, u_1, \dots, u_k)$ be such that $u_i = \Phi(x, u_1, \dots, u_k)$. The multi-type branching process defined by the laws of progeny*

$$\pi_{i,d} = \frac{x^{w_i(\mathbf{d})} u_1^{d_1} \cdots u_k^{d_k}}{\Phi_i(x, u_1, \dots, u_k)}$$

attribute to each tree, of which the root is of type 1, a probability which depends only on its size.

7. **Complications and Restrictions**

We have already discussed some of the problems of implementing such an algorithm. However, in the univariate case, they can be overcome as precise numerical evaluation is possible, and often it is easy to calculate the singularities directly. In the multivariate case there is some difficulty to verify that the rationality and non-periodicity requirements are met.

Bibliography

[1] Drmota (Michael). – Systems of functional equations. *Random Structures & Algorithms*, vol. 10, n° 1-2, 1997, pp. 103–124. – Average-case analysis of algorithms (Dagstuhl, 1995).