Journal of Classification 6:223-231 (1989)

Fast Random Generation of Binary, t-ary and Other Types of Trees

Adolfo J. Quiroz

Universidad Simón Bolívar

Abstract: Trees, and particularly binary trees, appear frequently in the classification literature. When studying the properties of the procedures that fit trees to sets of data, direct analysis can be too difficult, and Monte Carlo simulations may be necessary, requiring the implementation of algorithms for the generation of certain families of trees at random. In the present paper we use the properties of Prufer's enumeration of the set of completely labeled trees to obtain algorithms for the generation of completely labeled, as well as terminally labeled t-ary (and in particular binary) trees at random, i.e., with uniform distribution. Actually, these algorithms are general in that they can be used to generate random trees from any family that can be characterized in terms of the node degrees. The algorithms presented here are as fast as (in the case of terminally labeled trees) or faster than (in the case of completely labeled trees) any other existing procedure, and the memory requirements are minimal. Another advantage over existing algorithms is that there is no need to store pre-calculated tables.

Keywords: Tree algorithms; Monte Carlo studies; Clustering methodology.

1. Introduction

Interest in the generation of binary trees at random has been recently motivated by the applications to Monte Carlo studies in clustering. For instance, De Soete, DeSarbo, Furnas and Carroll (1984) use random,

---

Author's Address: Adolfo J. Quiroz, Universidad Simón Bolívar, Departamento de Matemáticas, Aptdo. 80659, Caracas 1080A, Venezuela.

terminally labeled binary trees to evaluate an algorithm for the fitting of ultrametric and path length trees to rectangular matrices of proximity data. Fowlkes, Mallows and McRae (1983), use random binary trees to study the 'shape' of trees obtained from hierarchical clustering procedures. In computer science, the enumeration and random generation of trees other than binary have been considered. Nijenhuis and Wilf (1975) give algorithms for the random generation of completely labeled trees and unlabeled rooted trees. They consider 'free' trees, that is, no restrictions are placed on the node degrees. Trojanowski (1978) and Ruskey (1978) give procedures for the listing of all t-ary trees. When considering random generation of trees, most authors have devoted their attention to the uniform distribution, that is, the distribution that assigns equal probability to all members of the family of trees under consideration. The uniform distribution seems a natural default choice for the analysis of many procedures involving trees, and it also has the advantage that some theoretical results are available for that distribution (see, for example, Renyi 1959, Meir and Moon 1970, and Robinson and Schwenk 1975). It is nevertheless worth mentioning that in some cases other ways of generating random trees can be more appropriate and have been considered in the literature. In order to study the stability of the trees obtained from clustering procedures, Gnanadesikan, Kettenring and Landwehr (1977, section 2.4) introduce the notion of "shaking the tree": they add random multivariate normal noise to the original data set being clustered, and apply the same clustering procedure to the perturbed data. Furnas (1984), gives a complete survey of the different methods available for the random generation of several classes of binary trees. We refer the reader to that paper for both terminology and a more complete list of references on the subject. In this paper we give algorithms for the generation at random, under the uniform distribution, of several classes of trees. We consider completely labeled as well as terminally labeled trees, and rooted as well as unrooted trees. Our results generalize and improve, from the point of view of execution time, some of the results given by Furnas (1984) and Nijenhuis and Wilf (1975, Chapter 24). The main theoretical tools used here are the properties of the Prufer enumeration of the set of completely labeled trees.

For clarity of exposition, we begin by describing our methods in the particular case of t-ary (including binary) trees. Section 2 discusses completely labeled trees and Section 3 covers terminally labeled trees. Then, in Section 4, we indicate how the methodology can be extended to other families of trees.

2. Completely Labeled t-ary Trees and the Prufer Enumeration

For the necessary graph-theoretic terminology we refer the reader to Furnas (1984). With that notation, the family of rooted completely labeled full t-ary trees on n nodes (where n has to be of the form $1 + kt$, k a non-negative integer) can be characterized in terms of node degree as follows:

Proposition 2.1. *A completely labeled tree on $1 + kt$ nodes, k a positive integer, is a rooted full t-ary tree if and only if it has one node of degree t (the root), $k - 1$ nodes of degree $t + 1$ and $t + (k - 1)(t - 1)$ nodes of degree one (the leaves).*

Proof. Easy by induction on the number of nodes of degree $t + 1$, by noting that, whenever a leaf "has offspring" and becomes a node of degree $t + 1$, we lose a leaf and get t new ones.

Similarly, for unrooted full t-ary trees we have:

Proposition 2.1b. *A completely labeled tree on $2 + kt$ nodes, $k > 1$, is an unrooted full t-ary tree if and only it has k nodes of degree $t + 1$ and $2 + k(t - 1)$ nodes of degree 1.*

Cayley's Theorem (1889) asserts that the number of completely labeled trees on n nodes is n^{n-2}. A simple proof of this result is due to Prufer (1918) (see also Moon 1967). Prufer's proof establishes a one-to-one map between the set of completely labeled trees on n nodes and the set of lists of $n - 2$ integers in $\{1, \ldots, n\}$ (allowing repeats). We call such a list a Prufer list. The one-to-one map is given by the following algorithm:

Algorithm A1. (Construction of a completely labeled tree on n nodes from a Prufer list.)

1. Let $\mathbf{a} = (a_1, a_2, \ldots, a_{n-2})$ be the Prufer list corresponding to a completely labeled tree on n nodes. Let $\mathbf{b} = (1, 2, \ldots, n)$.
2. Pick smallest number in **b** not in **a**. Call it x_1. Join a_1 to x_1 to form an edge. Erase a_1 from **a** and x_1 from **b**.
3. Repeat step 2, (joining a_2 to x_2, etc) until **a** is empty.
4. Join the two numbers left in **b** to form the last edge of the tree.

Prufer's algorithm allows us to analyze certain properties of trees by studying the more manageable set of Prufer lists. For example, Renyi (1959) has used Prufer lists to obtain the asymptotic distribution of the leaves-to-

nodes ratio in random completely labeled trees. For our application we will need the following:

Proposition 2.2. *Algorithm A1 can be implemented to be $O(n)$ in both time and space requirements.*

Nijenhuis and Wilf (1975, Chapter 24) provide an implementation of algorithm A1 which requires $O(n)$ storage but is $O(n^2)$ in execution time. The following implementation of A1 proves Prop. 2.2:

Procedure PruferTree. Input variables are the integer n and the array $A(n-2)$ containing a Prufer list. Output is the array EDGE$(n-1,2)$ containing the $n-1$ edges of the corresponding tree.

1. Initialize arrays $M(n)$ and $U(n)$ to be identically zero. $U(i)$ is equal to 1 if the node i has been erased from the list B, 0 otherwise.
2. For $i = 1$ to $n-2$ do

$$M(A_i) \leftarrow M(A_i) + 1$$

   $M(i)$ contains the multiplicity (number of occurrences) of i in the list A.
3. Find smallest i_0 such that $M(i_0) = 0$.

$$P_1 \leftarrow i_0 \,;\, P_2 \leftarrow i_0 \,.$$

   P_1 and P_2 are pointers to the smallest elements of B not in A.
4. For $i = 1$ to $n-2$ do

```
    begin
      if P2 < P1 then
        begin
          EDGE(i,1) ← P2
          EDGE(i,2) ← Ai
          U(P2) ← 1
          P2 ← P1
        end
          else
        begin
          EDGE(i,1) ← P1
          EDGE(i,2) ← Ai
          U(P1) ← 1
          P1 ← P1 + 1
          while M(P1) not equal 0 do
          P1 ← P1 + 1
```

end
$M(A_i) \leftarrow M(A_i) - 1$
if $(M(A_i) = 0$ and $A_i < P_1)$ then $P_2 \leftarrow A_i$
end

5. There are only two values of i left for which $M(i) > 0$. Use those two to form the last edge of the tree.

Clearly, steps 1,2,3 and 5 of PruferTree are $O(n)$ in execution time. Since the value of P_1 never decreases during step 4 and it is never more than n, the internal while loop in step 4 increments P_1 a total of no more than $n-1$ times. Therefore, step 4 and PruferTree are $O(n)$ in execution time.

The connection between the node degrees in a completely labeled tree and the corresponding Prufer list is given by the next proposition. For a proof, see Even (1979, page 28) or Moon (1970).

Proposition 2.3. *The degree of the node labeled i in a completely labeled tree equals the number of times i appears in the corresponding Prufer list, plus 1.*

As a corollary of Propositions 2.1 and 2.3 we have the following characterization of completely labeled full t-ary trees in terms of their Prufer lists.

Proposition 2.4a. *The Prufer list of a completely labeled tree on* $1 + kt$ *nodes, the node labels being the elements of* $\mathbf{N} = \{1,2,\ldots,1+kt\}$, *corresponds to a rooted full t-ary tree if and only if there are* $k-1$ *numbers in* N *that appear exactly t times in the Prufer list and one number in* N *that appears exactly* $t-1$ *times in the Prufer list.*

And, similarly

Proposition 2.4b. *The Prufer list of a completely labeled tree on* $2 + kt$ *nodes, the node labels being the elements of* $\mathbf{N} = \{1,2,\ldots,2+kt\}$, *corresponds to an unrooted full t-ary tree if and only if there are k numbers in* N *that appear exactly t times in the Prufer list.*

Using proposition 2.4a we obtain the following algorithm for the random generation of completely labeled rooted full t-ary trees. A similar procedure, using 2.4b, would yield the corresponding unrooted trees.

Algorithm A2. (Random completely labeled rooted full t-ary tree on $1 + kt$ nodes.)

1. Let $\mathbf{N} = \{1,2,\ldots,1 + kt\}$.
2. Choose a subset $M = \{m_1,\ldots,m_{k-1}\}$ at random from N.
   Choose an element of $\mathbf{N} - \mathbf{M}$ at random. Call it m_0. (The root).
   Let

   $$\mathbf{a}' = (a_0',a_1',a_2',\ldots,a_{kt-1}')$$
   $$= (m_0,\ldots,m_0,m_1,\ldots,m_1,\ldots,m_{k-1},\ldots,m_{k-1}),$$

   where m_0 appears $t-1$ times, while each of $m_1,m_2,\ldots,m_{k-1}$, appears t times.
3. Choose a permutation π at random from $\mathbf{S}_{kt-1}$, the set of all permutations on $1,2,\ldots,kt-1$.
   Let $\mathbf{a} = (a'_{\pi_1}, a'_{\pi_2},\ldots, a'_{\pi_{kt-1}})$ and $\mathbf{b} = (1,2,\ldots,1 + kt)$.
4. With the lists **a** and **b** just defined, proceed to construct the tree by using algorithm A1. The list **a**, obtained in steps 1 and 2 of A2, is the Prufer list of a rooted t-ary tree, and clearly, all such lists are given the same probability. Steps 1 and 2 of A2 can be implemented to be $O(kt)$ in both space and time requirements. (See Knuth,1980, Vol. 2, Chapter 3). Since our implementation of algorithm A1 requires linear time and space, it follows that algorithm A2 is $O(kt)$ in both time and space requirements. Binary trees are obtained by letting $t = 2$.

3. Random Terminally Labeled t-ary Trees

It is usually the case, when trees are used in hierarchical clustering procedures, that the names given to the internal nodes of the tree are unimportant. Only the shape of the tree and the names of the terminal vertices matter, since these vertices represent the data being clustered. In this context, the concept of terminally labeled tree fits naturally. (As before, the reader will find the formal definitions in the paper of Furnas, 1984). Let Γ be the collection of completely labeled rooted trees on $n + m$ nodes, of which n are leaves, the leaves being labeled 1 to n and the root being labeled $n + 1$. Let $L \in \Gamma$. From L, we can produce a unique terminally labeled rooted tree T by erasing the labels from the internal nodes. Call C (for canonical) the map that sends L to T. Since we obtain the same T for all permutations of the labels of the internal nodes of L, not including the root, we have that the cardinality of $C(T)$ is $(m-1)!$. (See Furnas, 1984, Theorems 2.3.1 and 2.4.1). This, together with Proposition 2.1 implies that random terminally labeled full t-ary trees can be obtained by sampling from the uniform distribution on Γ and

then deleting the labels of the internal nodes in the resulting trees. This is done by the following algorithm.

Algorithm A3. (Random terminally labeled rooted full t-ary tree on $1 + kt$ nodes.)

1. Let $m = t + (k-1)(t-1)$; m is the number of leaves.
2. Let

   $$\mathbf{a}' = (a'_1, a'_2, \ldots, a'_{kt-1})$$
   $$= (m+1, \ldots, m+1, m+2, \ldots, m+2, \ldots, 1+kt, \ldots, 1+kt),$$

   where $m + 1$ appears $t - 1$ times and the rest of the numbers greater than m appear t times each.
3. Choose a permutation π at random from S_{kt-1} Let $\mathbf{a} = (a'_{\pi_1}, a'_{\pi_2}, \ldots, a'_{\pi_{kt-1}})$ and $\mathbf{b} = (1,2,\ldots,1+kt)$.
4. With the lists **a** and **b** just defined, proceed to construct the tree by using algorithm A1.

By leaving the numbers 1 to m out of the Prufer list, these numbers become automatically the labels of the leaves. We make $m + 1$ the label of the root. Since steps 1, 2 and 3 are linear in kt, algorithm A3 is $O(kt)$ in both space and time requirements. As before, the binary tree case is obtained by letting $t = 2$. This improves on the best results given by Furnas (1984) for binary trees. (His Construction Procedure III is $O(kt \log(kt))$ in execution time). A procedure similar to A3 yields the corresponding unrooted trees.

4. Random Generation of Other Classes of Trees

The only property of t-ary trees that we have used in the previous sections is the fact that a characterization in terms of node degrees (Proposition 2.1) is possible for this family of trees. Suppose that we have a family Γ of completely labeled trees on n nodes, that can be characterized as the family of trees with a_k nodes of degree k, $1 \le k \le K_{max}$, for some positive integer K_{max}. Then we can produce random trees from Γ by mimicking Algorithm A2 as follows:

Algorithm A4. (General procedure for random generation of completely labeled trees.)

1. Let $\mathbf{N} = \{1,2,\ldots,n\}$, **M** be the empty set , be the empty list and $k = 2$.
2. Choose a subset

$$\mathbf{M}_k = \{m_{k_1}, \ldots, m_{k_{ak}}\}$$

at random from $\mathbf{N} - \mathbf{M}$.
$\mathbf{M} \leftarrow \mathbf{M} \cup \mathbf{M}_k$.
$\mathbf{a}' \leftarrow (\mathbf{a}', m_{k_1}, \ldots, m_{k_1}, \ldots, m_{k_{ak}}, \ldots, m_{k_{ak}})$,
where each of $m_{k_1}, \ldots, m_{k_{ak}}$ appears $k-1$ times in the list being concatenated to $\mathbf{a}'$.

3. If $k < K_{max}$, $k \leftarrow k+1$ and go to step 1. Otherwise go to step 3.
4. Choose a permutation π at random from $\mathbf{S}_{n-2}$. Let $\mathbf{a} = (a'_{\pi_1}, a'_{\pi_2}, \ldots, a'_{\pi_{n-2}})$ and $\mathbf{b} = (1,2,\ldots,n)$.
5. With the lists **a** and **b** just defined, proceed to construct the tree by using algorithm A1.

Algorithm A4 is linear in time and space requirements, and can be adapted to produce terminally labeled trees in the same way described in section III for the t-ary trees. Since all Prufer lists corresponding to trees in Γ are given the same probability by algorithm A4, the resulting random trees have the uniform distribution on Γ. In some cases, even if the family of trees being considered does not have a complete characterization in terms of node degrees, the Prufer enumeration might still be used to obtain trees from that family at random. One such example is the family of 'free' trees considered by Nijenhuis and Wilf (1975, Chapter 24). Another example that has been studied in the literature is the family of terminally labeled rooted trees with n leaves and m internal nodes, where no restriction is imposed on the degrees of the internal nodes. Oden and Shao (1984), give an algorithm for the generation of these trees at random.

Their implementation is $O(n^2)$ in storage requirements, and also requires a one-time initialization which is $O(n^2)$ in execution time, while generating each tree, after the initialization, requires linear time. Our method allows us to give a $O(n+m)$ algorithm, in both execution time and memory requirements, to produce these trees at random:

Algorithm A5. (Random terminally labeled rooted tree with n leaves and m internal nodes.)

1. For $1 \le i \le m$, let $a'_i = n + i$. This step ensures that all nodes labeled $n+1$ to $n+m$ appear in the Prufer list and, therefore, are not leaves.
2. For $m < i \le n+m-2$, let a'_1 be a random number between $n+1$ and $n+m$ (inclusive).
3. Choose a permutation pi at random from $\mathbf{S}_{n+m-2}$.
   Let $\mathbf{a} = (a'_{\pi_1}, a'_{\pi_2}, \ldots, a'_{\pi_{n+m-2}})$ and $\mathbf{b} = (1,2,\ldots,n+m)$.
4. With the lists **a** and **b** just defined, proceed to construct the tree by using algorithm A1.

5. Choose the root of the tree at random from the internal nodes.
6. Remove (ignore) the labels of the internal vertices.

Fortran implementations of algorithms A1 to A4 are available from the author.

References

CAYLEY, A. (1889), "A Theorem on Trees," *Quarterly Journal of Pure and Applied Mathematics, 23*, 376-378.

DE SOETE, G., DESARBO, W. S., FURNAS, G. W., and CARROLL, J. D.(1984), "The Estimation Of Ultrametric And Path Length Trees From Rectangular Proximity Data," *Psychometrika 49, no. 3*, 289-310.

EVEN, S. (1979), *Graph Algorithms*, Rockville, MD: Computer Science Press.

FOWLKES, E. B., MALLOWS, C. L., and MCRAE, J. E. (1983), "Some Methods for Studying The Shape of Hierarchical Trees," Murray Hill, NJ: AT&T Bell Laboratories Technical Memorandum 83-11214-6.

FURNAS, G. W. (1984),"The Generation of Random, Binary Unordered Trees," *Journal of Classification 1*, 187-233.

GNANADESIKAN, R., KETTENRING, J. R., and LANDWEHR, J. M. (1977), "Interpreting and Assessing The Results of Cluster Analyses," *Bulletin of the International Statistical Institute, 47*, 451-463.

KNUTH, D. E. (1981), *The Art of Computer Programming*, Second edition, Reading, MA: Addison-Wesley.

MEIR, A., and MOON, J. W. (1970), "The Distance Between Points in Random Trees," *Journal of Combinatorial Theory, 8*, 99-103.

MOON, J. W. (1967), "Various Proofs of Cayley's Formula for Counting Trees," in *A Seminar on Graph Theory*, ed. F. Harary, New York: Holt, 70-78.

MOON, J. W. (1970), "Counting Labeled Trees," *Canadian Mathematical Monographs*, No. 1.

NIJENHUIS, A., and WILF, H. F.(1975), *Combinatorial Algorithms*, New York: Academic Press.

ODEN, N. L., and SHAO, K-T. (1984), "An Algorithm to Equiprobably Generate all Directed Trees with K Labeled Terminal Nodes and Unlabeled Interior Nodes," *Bulletin of Mathematical Biology, 46(3)*, 379-387.

PRUFER, H. (1918), "Neuer Beweis eines Satzes uber Permatationen," *Archives of Mathematical Physics, 27*, 142-144.

RENYI, A. (1959), "Some Remarks on The Theory of Random Trees," *Publications of the Mathematical Institute of the Hungarian Academy of Sciences, 4*, 73-85.

ROBINSON, R. W., and SCHWENK, A. J. (1975), "The Distribution of Degrees in a Large Random Tree," *Discrete Math, 12*, 359-372.

RUSKEY, F. (1978), "Generating t-ary Trees Lexicographically," *SIAM Journal on Computing, 7*, 424-439.

TROJANOWSKI, A. E. (1978), "Ranking and Listing Algorithm for k-ary Trees," *SIAM Journal on Computing, 7*, 492-509.