A Calculus of Random Generation

Philippe Flajolet
INRIA Rocquencourt

February 1, 1993

[summary by Xavier Gourdon]

Abstract

A systematic approach to the random generation of labelled combinatorial objects is presented. It applies to structures that are decomposable, *i.e.*, formally specifiable by grammars involving union, product, set, sequence, and cycle constructions.

This work started with a question arising in statistical classification theory: How can one generate a random "hierarchy"? In combinatorial terms, the generation problem simply amounts to drawing uniformly at random a tree with internal nodes of degree at least 2 and with leaves (external nodes) labelled by distinct integers, the number n of leaves being fixed. The need arises in statistics as one would like to generate at random such hierarchies and compare their characteristics to hierarchical classifications obtained from real-life data.

There are well-known methods for coping with this type of tree generation problems, the general strategy relying on a divide-and-conquer principle: Generate the root with the suitable probability distribution, then recursively generate the root subtrees. Several of the basic principles of this recursive top-down approach have been formalized by Nijenhuis and Wilf in their reference book on combinatorial algorithms [7], by Hickey and Cohen in the case of context-free languages [5], and under a fairly general setting by Greene within the framework of labelled grammars [4]. This work is in many ways a systematization and a continuation of the pioneering research of these authors.

The original article can be found in [2].

1. Combinatorial structures and constructions

We consider *labelled objects*. We start from the *initial objects* $\mathbf{1}$ that designates the "empty" structure of size 0 that bears no label, and Z that generically designates a single labelled node of size 1. We operate with the usual collection of labelled *constructions*,

$$+,\ \cdot\ ,\ \mathit{sequence}(\,),\ \mathit{set}(\,),\ \mathit{cycle}(\,). \tag{1}$$

We deal with the following structures, similar to the ones considered in [1].

Definition 1. Let $\mathbf{T} = (T_0, T_1, \ldots, T_m)$ be an $(m+1)$-tuple of classes of combinatorial structures. A *specification* of $\mathbf{T}$ is a collection of $m+1$ equations, with the ith equation being of the form

$$T_i = \Psi_i(T_0, T_1, \ldots, T_m) \tag{2}$$

where Ψ_i is a term built from $\mathbf{1}$, Z, and the T_j, using the standard constructions listed in (1).

We also say, for short, that the system (2) is a specification of T_0. A structure that admits a specification is called *decomposable*. The framework of specifications resembles that of context-free grammars for formal languages, but enriched with additional constructions.

From the usual transformation rules on the exponential generating functions (egf), it is possible to derive from (2) a set of equations which specify the corresponding egf. A consequence is that given a specification, the corresponding enumerating sequences up to size n are all computable in $\mathcal{O}(n^2)$ arithmetic operations.

2. Standard specifications

In this section, we consider reduction of specifications to standard form. The standard specifications constitute the basis of the random generation procedures to be developed in the paper.

Besides the transformation into standard form, we need the pointing operator, defined as follows. Given a class A of structures, the pointing of A is a class denoted ΘA, $\Theta A = \bigcup_{n=1}^{\infty} (\mathcal{A}_n \times [1\,..\,n])$, where $\mathcal{A}_n$ is the subclass of objects in A having size n and $[1\,..\,n]$ is the integer interval $\{1, 2, \ldots, n\}$. In other words, an object in the class ΘA can be viewed as an object of A with the additional property that one of the labels, corresponding to the field in $[1\,..\,n]$, is distinguished. From the definition we have that $C = \Theta A$ implies $C_n = nA_n$. Thus, the egfs are still computable by the added rule

$$C = \Theta A \qquad \Longrightarrow \qquad C(z) = \Theta A(z), \text{ where } \Theta f(z) = z \cdot \frac{d}{dz} f(z).$$

Developments in this section are inspired by Joyal's elegant theory [6] and by Greene's work [4].

Definition 2. Let $\mathbf{T} = (T_0, T_1, \ldots, T_m)$ be a tuple of classes of combinatorial structures. A *standard specification* of $\mathbf{T}$ is a collection of $m+1$ equations, the ith equation being of one of the forms

$$T_i = \mathbf{1};\ T_i = Z;\ T_i = U_j + U_k;\ T_i = U_j \cdot U_k;\ \Theta T_i = U_j \cdot U_k,$$

where each $U_j \in \{\mathbf{1}, Z, T_0, \ldots, T_m, \Theta T_0, \ldots, \Theta T_m\}$.

Theorem 1 (Standardization algorithm). *Every decomposable structure admits an equivalent standard specification.*

The proof is actually a conversion algorithm. For example, the transformation rule for the sequence construction $B = \mathit{sequence}(A)$ is $B = \mathbf{1} + A \cdot B$. As for the set construction $B = \mathit{set}(A)$, the transformation is $\Theta B = B \cdot \Theta A$, this being understood as a fundamental combinatorial isomorphism: Pointing at a node in a set individuates the component containing the node and the component becomes pointed; this leaves aside a set of components, the non-marked ones. The cycle construction $B = \mathit{cycle}(A)$ can be translated into $\Theta B = C \cdot \Theta A$, $C = \mathit{sequence}(A)$. Similar combinatorial principle apply to the reduction of sequences, sets, and cycles under cardinality constraints.

As an illustration, a standard form for hierarchies defined by $H = Z + \mathit{set}(H, \text{card} \geq 2)$ is

$$\{H = Z + U_1,\ \Theta U_1 = U_2 \cdot \Theta H,\ \Theta U_2 = U_3 \cdot \Theta H,\ \Theta U_3 = U_3 \cdot \Theta H\}.$$

3. Basic generation schemes

From the preceding section, it is sufficient to exhibit generation routines for standard specifications. This goal is achieved by means of a set of translation rules or "*templates*", inspired by existing technology of random generation [4, 5, 7]. A *preprocessing stage* furnishes, once and for all in time $\mathcal{O}(n^2)$ and in storage $\mathcal{O}(n)$ the enumerating sequences, up to size n, of structures intervening in a specification.

Given any class C, recall that $c_n = C_n/n!$ is its normalized counting sequence, from now on assumed to be available. We let gC denote a random generation procedure relative to class C. The general strategy is based on the *divide-and-conquer* principle.

$\mathbf{T}_0$. *Initial structures.* The generation procedures corresponding to $\mathbf{1}$ and Z are trivial.

$\mathbf{T}_1$. *Unions.* If $C = A + B$, the probability that a C-structure of size n arises from A is simply a_n/c_n. The random generation procedure uses a variate U drawn uniformly from the real interval $[0, 1]$ to effect the choice.

$\mathbf{T}_2$. *Products.* If $C = A \cdot B$, the probability that a C-structure of size n has an A-component of size k and a B-component of size $n-k$ is

$$\binom{n}{k} \frac{A_k \cdot B_{n-k}}{C_n} \equiv \frac{a_k \cdot b_{n-k}}{c_n}.$$

The random generation procedure results from this equation.

$\mathbf{T}_3$. *Pointing.* Generating A and ΘA are clearly equivalent processes.

THEOREM 2 (SEQUENTIAL RANDOM GENERATION). *The templates* $\mathbf{T}_0$, $\mathbf{T}_1$, $\mathbf{T}_2$, *and* $\mathbf{T}_3$ *produce from any standard specification* Σ_0 *a collection of random generation routines* $g\Sigma_0$. *Each routine of* $g\Sigma_0$ *uses precomputed tables consisting of* $\mathcal{O}(n)$ *integers; its worst case time complexity is of* $\mathcal{O}(n^2)$ *arithmetic operations.*

4. Boustrophedonic random generation

The standardization theory implies that all the complexity lies in the random generation of products. More precisely, when measured in the number of while-loops executed, the cost of generating (α, β) by the sequential method is the size of the first component, $|\alpha|$. In fact, a worst-case complexity of $\mathcal{O}(n \log n)$ can be achieved for all decomposable structures. The principle is simply a *boustrophedonic*[1] search. Given a product $C = A \cdot B$, we let K be the random variable denoting the size of the A-component of a C-structure. Amongst C-structures of size n, we have

$$\pi_{n,k} := \Pr\{K = k\} = \frac{a_k \cdot b_{n-k}}{c_n}.$$

The idea is to appeal to a special search for the drawing of K with the probability distribution $\{\pi_{n,k}\}_{k=0}^{n}$. Instead of the order of increasing values of k, we explore the possibilities of K in the boustrophedonic order

$$\pi_{n,0}, \pi_{n,n}, \pi_{n,1}, \pi_{n,n-1}, \cdots,$$

that sweeps alternatively from left to right and back. The recurrence translating the cost admit $\mathcal{O}(n \log n)$ solutions (see [3, Sec. 2.2]), leading to the following result.

THEOREM 3 (BOUSTROPHEDONIC RANDOM GENERATION). *Any* decomposable *structure has a random generation routine that uses precomputed tables of size* $\mathcal{O}(n)$ *and achieves* $\mathcal{O}(n \log n)$ *worst case time complexity.*

The purpose of the next sections is to come up with adequate specifications that permit to attain a complexity of $\mathcal{O}(n \log n)$ involving low multiplicative factors by exploiting "natural" regularities present in combinatorial structures. To algorithms designers, the situation resembles that of heapsort —which has guaranteed $\mathcal{O}(n \log n)$ complexity— versus quicksort —which is $\mathcal{O}(n \log n)$ only on average but with small constants—, so that quicksort is often preferred in practice.

5. The cost algebra of sequential generation

It is possible to examine the cost structure underlying the random generation procedures of the *sequential* group. This can be achieved thanks to a cost algebra, developed in [2], and corresponding to that of *complexity descriptor* in [1]. For example, it can be proved that the generation algorithm for binary plane trees corresponding to the standard specification $\{B = Z + U_1;\ U_1 = B \cdot B\}$ has average case complexity $\gamma B_n = \frac{1}{2}\sqrt{\pi} n^{3/2} + \mathcal{O}(n)$.

[1] Boustrophedonic: turning like oxen in ploughing (Webster).

6. The analysis of cost generating functions

The cost algebra we mentioned in the previous section attains its full dimension when we examine it in the light of asymptotic properties of combinatorial structures. This means that orders of growth of coefficients should be taken into account. Consideration of asymptotic properties of structures using the classical arsenal of complex analysis does provide, in all cases of practical interest, valuable guidelines regarding the design of generation algorithms.

Let's see what happens on the family of non-plane trees, whose specification is $A = Z \cdot set(A)$. It furnishes a first example where two random generation algorithms derived from combinatorially equivalent specifications lead to rather different complexity behaviours. We make use of the general principles of the standardization method, our starting point being the pair of combinatorially equivalent specifications

$$\Theta A \cong A + (\Theta A \cdot A) \cong A + (A \cdot \Theta A).$$

Applying our cost algebra leads to the following theorem.

THEOREM 4 (NON PLANE TREES). *(i). The random generation algorithm for labelled trees corresponding to the standard specification* $\Theta A = A + (\Theta A \cdot A)$ *has average cost*

$$\gamma A_n = \sqrt{\frac{\pi}{2}} n^{3/2} + \mathcal{O}(n).$$

(ii). The generation algorithm for labelled trees corresponding to the specification $\Theta A = A + (A \cdot \Theta A)$ *has average cost*

$$\gamma A_n = \frac{1}{2} n \log n + \mathcal{O}(n).$$

This result suggests optimization transformations. The pointed trees are much more numerous than the basic trees, the ratio being $\Theta A_n / A_n = n$. Accordingly, the mark tends to fall on larger portions of the tree, thus leading to the complexity $\mathcal{O}(n \log n)$.

In order to make this discussion precise, we introduce a formal definition.

DEFINITION 3. Given two generating functions F and G, F *dominates* G, in symbols $F \gg G$, if

$$\frac{f_n}{g_n} \to \infty \qquad \text{as} \quad n \to +\infty.$$

The considerations regarding labelled trees then suggest a simple heuristic:

> **Big-endian heuristic.** Given a standard specification Σ_0, reorganize all comparable pairs in products each time $A \gg B$ using the isomorphism transformation
>
> $$(A \cdot B) \hookrightarrow (B \cdot A).$$

This heuristic applied to the two specifications of non-plane trees leads to the "good choice" with an $\mathcal{O}(n \log n)$ behaviour. A further optimization that this discussion suggests consists in obtaining, as much as possible, specifications where products are *imbalanced* so as to take full advantage of the big-endian heuristic. To that purpose, the Θ operator can be employed. For instance, let us re-examine the binary trees, $B = Z + B \cdot B$. Consider the induced relation obtained by differentiation,

$$\Theta B = Z + \Theta B \cdot B + B \cdot \Theta B.$$

Let K designate the size of the first component in $B \cdot B$, and K' denote the size of the first component in $B \cdot \Theta B$. The expectation of K is $n/2$ while that of K' turns out to be $\mathcal{O}(\sqrt{n})$, so that a global gain of order close to $\mathcal{O}(\sqrt{n})$ is to be anticipated if the big endian heuristic is employed. This dictates a new heuristic:

> **Differential heuristic.** Replace in specifications polynomial relations by differential relations.

For binary trees, the differential algorithm corresponding to the specification

$$\Theta B = Z + (B + B) \cdot \Theta B$$

leads to the average behaviour $\frac{1}{2} n \log n + \mathcal{O}(n)$.

Thanks to these optimization transformations, all polynomial families of trees as well as functional graphs can be generated in time asymptotic to $\frac{1}{2} n \log n$. Furthermore, the class of iterative structures admits $\mathcal{O}(n)$ random generation algorithms.

7. Numerical data

The generation method for decomposable structures has been implemented in the symbolic manipulation system MAPLE by P. Zimmermann. The complete programme tests specifications for well-foundedness, puts them in standard quadratic form, and compiles two sets of procedures from standard specifications: the counting routines that implement the convolution recurrences, and the random generation routines based on the templates. The whole set, in its current stage, represents some 1500 lines of Maple code. The random generation procedures produced are in the Maple language itself, and they take advantage of the multiprecision arithmetic facilities available in MAPLE.

The version of the Maple programme that was written furthermore compiles random generation routines by automatically implementing a version of the big-endian heuristic based on "probing". As an outcome, all our eleven reference structures are generated in time between 2 and 9 seconds on a machine of 20 Mips for size $n = 400$. Gains involving a factor of about 10 for $n = 400$ result from optimizations dictated by the cost calculus.

8. Unlabelled structures

Many important structures of computer science and combinatorics are *unlabelled*.

Work currently under redaction shows that the framework presented here extends to unlabelled combinatorial structures. (The treatment is only made more complex because of the occurrence of Pólya operators.) As a result: (*i*) all unlabelled decomposable structures including context-free languages and term trees of symbolic computation can be generated in worst-case time $\mathcal{O}(n \log n)$; (*ii*). Wilf's `RANRUT` Algorithm [8] has expected case complexity which is $\sim \frac{1}{2} n \log n$.

Bibliography

[1] Flajolet (P.), Salvy (B.), and Zimmermann (P.). – Automatic average-case analysis of algorithms. *Theoretical Computer Science, Series A*, vol. 79, n° 1, February 1991, pp. 37–109.

[2] Flajolet (Philippe), Zimmerman (Paul), and Van Cutsem (Bernard). – *A Calculus for the Random Generation of Labelled Combinatorial Structures.* – Research Report n° 1830, Institut National de Recherche en Informatique et en Automatique, January 1993. 29 pages. To appear in *Theoretical Computer Science*.

[3] Greene (D. H.) and Knuth (D. E.). – *Mathematics for the analysis of algorithms.* – Boston, Birkhauser, 1981.

[4] Greene (Daniel Hill). – *Labelled formal languages and their uses.* – PhD thesis, Stanford University, June 1983.

[5] Hickey (T.) and Cohen (J.). – Uniform random generation of strings in a context-free language. *SIAM Journal on Computing*, vol. 12, n° 4, 1983, pp. 645–655.

[6] Joyal (André). – Une théorie combinatoire des séries formelles. *Advances in Mathematics*, vol. 42, n° 1, 1981, pp. 1–82.

[7] Nijenhuis (Albert) and Wilf (Herbert S.). – *Combinatorial Algorithms.* – Academic Press, 1978, second edition.

[8] Wilf (Herbert S.). – *Combinatorial Algorithms: An Update.* – Philadelphia, Society for Industrial and Applied Mathematics, 1989, *CBMS-NSF Regional Conference Series.*