Random Sampling and Simulation

Philippe Duchon\textsuperscript{1} and Michèle Soria\textsuperscript{2}

1. Random Sampling of Combinatorial objects 13
2. Simulation of probability distributions 18

This chapter is devoted to Philippe Flajolet’s articles dealing with random sampling and simulation. His work in this area illustrates his combinatorial approach for designing unifying frameworks, and developing general efficient methods.

Random sampling is a companion of combinatorial modeling, that allows to extract different kinds of information on combinatorial objects. Philippe Flajolet designed generic methods, to produce sound and efficient algorithms for the random generation of discrete objects described by combinatorial specifications. These methods, largely based on analytic combinatorics, make it possible to sample objects and quantify parameters, for a large variety of combinatorial models, – including words, trees, graphs, automata, permutations, allocations, and so on. Sampling combinatorial objects rely on sampling numbers, since the construction of a random object depends on the choice of random numbers. Philippe Flajolet also designed and analysed various algorithms for generating numbers according to various probability laws, only using a source of perfect binary flips.

1. Random Sampling of Combinatorial objects

The random generation of combinatorial structures is a rich and ancient issue in combinatorial algorithmic. The objective is to construct an object of a desired type (without constructing all the objects) in such a way that all objects of the desired type are equally likely. The result can be considered as a "typical" object, on which characteristic properties can be visualized, such as the order of magnitude of a quantity of interest. For example the random generation of binary search trees shows a big difference in shape with term trees (height in $\log n$ in the first case, as opposed to $\sqrt{n}$
in the latter case, for a tree with $n$ nodes), and reveals the different underlying models (PFcite).

Random generation has always been of great help for testing combinatorial conjectures and investigating complex parameters of discrete structures. With the possibility of computer-aided generation, it has become an invaluable tool for testing correctness and robustness of software, and also validating models. For example, the recursive method (see below) originated from a question by Bernard Van Cutsem, a statistician who was working on classification. This is how Philippe Flajolet relates the question (cf PF Maple Sheet 1997 "A problem in statistical classification theory")

Van Cutsem’s original question was motivated by the following problem: Classification programmes in statistics build classification trees, usually proceeding by successive aggregations of closest neighbours amongst existing classes. How can we measure the way a classification carries useful information and not just ”random noise”? Certainly, ”good” classification trees should exhibit characteristics that depart significantly from random ones. Hence the need to simulate and analyse parameters of random classification trees.

This problem is at the origin of the development of the Combstruct package in Maple (see Chapter 2 in this volume), which offers a "draw" function for uniform drawing a an object of a given size, satisfying a combinatorial specification. Computer-aided experiments need to draw an important number of objects of huge size, thus the necessity of efficient sampling algorithms. The methods of random generation introduced by Philippe Flajolet, being generic and in quasi-linear time, make it possible to address applicative domains where large scale automatic sampling is needed.

Random generation first developped ad hoc methods relying on a deep understanding of the combinatorial structures. A first approach for systematization of uniform sampling was introduced by Nijenhuis and Wilf [?, ?], for recursive structures that can be enumerated in polynomial time: this method applies to structures that can be decomposable in terms of multisets of primitive objects, and provides algorithms for listing objects of a given size, ranking and unranking, as well as uniform sampling. In the case of context-free languages, Hickey and Cohen [?] give a similar method relying on the grammar for the recursive decomposition. An alternative method, was introduced by Jerrum and Sinclair [?, ?] for sampling objects that cannot be enumerated in polynomial time: using Markov chain allows to stochastically perturbate an arbitrary structure in order to obtain a uniformly random structure. The Markov chains methods, such as the Propp-Wilson technique of coupling from the past, are adapted to certain types of objects amenable to suitable transformations related to certain definite dynamic.

Philippe Flajolet developed two models for random sampling, within the framework of Analytic Combinatorics: "Recursive sampling" and "Boltzman sampling". Both are decomposition-based methods, using the specification of combinatorial classes in terms of constructor such as Union, Product, Set, Sequence, Cycle. And the motto of Analytic Combinatorics does apply : "If you can specify it, you can analyse and sample it".
Both sampling methods introduced by Philippe Flajolet start from the specification of a combinatorial class, and automatically produce a probabilistic algorithm for sampling objects in the class.

**Specification** $C = F(A, B, \cdots)$ $\leadsto$ **Sampler** $\Gamma C = F(\Gamma A, \Gamma B, \cdots)$

The basic idea is to provide, for each combinatorial construction, a sampling rule that builds a random sampler for the composed class from the random samplers of the composite classes. The resulting sampling algorithms directly reflect the combinatorial structure, and the probabilistic choices depend on counting quantities: *enumeration sequences* in the case of "Recursive sampling", and *values of generating functions* for "Boltzman sampling". This automatic translation between specifications and sampling algorithms is an illustration of the "Symbolic Method". On the other hand, the study of the algorithms efficiency relies on "Complexity Analysis".

<table>
<thead>
<tr>
<th>Recursive sampling</th>
<th>Boltzman sampling</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specification of class $C$</td>
<td>Specification of class $C$</td>
</tr>
<tr>
<td>$n \to \gamma_n$; $\mathbb{P}(\gamma_n) = 1/c_n$</td>
<td>$x \to \gamma$; $\mathbb{P}(\gamma) \propto x^{\gamma}$</td>
</tr>
<tr>
<td>Simple probabilistic algorithm</td>
<td>Simple probabilistic algorithm</td>
</tr>
<tr>
<td>Precomputation of enumeration sequences</td>
<td>Evaluation of generating functions</td>
</tr>
<tr>
<td>Generation in $\Theta(n \log n)$</td>
<td>Approximate size, generation in $O(n)$ for most classes</td>
</tr>
</tbody>
</table>

**Recursive sampling.** In the beginning of the nineties, Philippe Flajolet, Bernard Van Cutsem and Paul Zimmermann (PFlajolet) proposed, a general schema for uniform sampling in labeled classes defined by combinatorial specifications. This work is in the continuity of the work of Nijenhuis and Wilf, with a wider formalization framework, well-suited for automatic average complexity calculation (see Chapter 3 in this volume).

The specification are recursively defined from two types of primitive objects (empty with size 0, and atom with size 1) by application of five basic operators: Union, Product, Set, Sequence, Cycle. The translation of combinatorial specifications into functional equations on generating functions is the core of the "Symbolic Method" (see Chapter 1 Volume I); but in the case of sampling, in order for the algorithms to be efficient, the specification has to be described in "normal form" (each specification involves only the sum or product of two classes, with an extra *pointing* operator, related to differentiation), and the sampling of an object of size $n$ is preceded by the computation of the number of objects of size up to $n$, in all classes appearing in the specification.

The algorithms for constructing a random object of size $n$ in $C = \text{Union}(A, B)$, and $C = \text{Product}(A, B)$ are given below. The number of objects of size $n$ in $C$ -resp. $A$ and $B$, are denoted by $c_n$ -resp. $a_n$ and $b_n$. For a sum, $c_n = a_n + b_n$, and for a product, $c_n = \sum_{k=0}^{n} a_k b_{n-k}$. 
\[ C = \text{Sum}(A, B) \]
\[ C = \text{Product}(A, B) \]

Procedure \( \Gamma C \) (\( n: \text{integer} \))

\[ \text{Proc } \Gamma C(n: \text{integer}) \]
\[ U := \text{uniform}(0, 1) \]
\[ k := 0 \]
\[ S := a_0 b_n / c_n \]
\[ \text{if } U < a_n / c_n \text{ then return } \Gamma A(n) \]
\[ \text{else return } \Gamma B(n) \]
\[ \text{while } U > S \text{ do } k := k + 1; S := a_k b_{n-k} / c_n \]
\[ \text{return } (\Gamma A(k), \Gamma B(n-k)) \]

The precomputation of the enumeration sequences up to \( n \) requires \( O(n^2) \) arithmetic operations (which can be reduced to \( O(n) \) in the D-finite framework), and \( O(n) \) space. The above sampling algorithm for product necessitates \( O(n^2) \) arithmetic operations in the loop. Flajolet, Van Cutsem and Zimmermann give a clever way (boustrophédon) to organise the computations so that only \( O(n \log n) \) arithmetic operations are needed.

The authors also designed similar algorithms for the case of unlabelled classes, with an extra \( \text{Diagonal} \) operator, in order to take care of symmetries. The unlabeled case has not been published, but all the algorithms are implemented in the package Combstruct of Maple [1].

**Boltzmann sampling.** The ideas leading to Boltzman sampling were born a decade after recursive sampling. At the dawn of the second millenium, Philippe Flajolet and his co-authors Philippe Duchon, Guy Louchard and Gilles Schaeffer, were separately developing ideas for different generation methods: generation of trees inspired by branching process methods [1], generation of convex animals with sampling sequences by approaching the singularity [1], generation of kernels of two-connected planar maps with algorithms of extraction-rejection, which gave objects of approximate size [1]. These ideas converged to an efficient generic method called "Boltzmann sampling" [1].

The Boltzmann model assigns to each object \( \gamma \) a probability which is proportional to an exponential of its size: \( P(\gamma) \propto x^{|\gamma|} \), where \( x \) is a given parameter. In the unlabelled universe \( P(\gamma) = x^{|\gamma|} / X \), and the normalizing constant \( X \) is equal to \( C(x) = \sum_{\gamma \in C} x^{|\gamma|} \), corresponding to the evaluation of the ordinary generating function of class \( C \) for the value \( x \) (there is a similar definition in the labelled universe, using the exponential generating function).

Boltzmann sampling is uniform: all objects of same size are equally likely. But the size of the sampling output is a random variable \( N \): the distribution is spread over all objects of the class, governed by the parameter \( x \): \( P(N = n) = c_n x^n / C(x) \) and \( E(N) = x C'(x) / C(x) \).

As for recursive sampling, Boltzmann sampling translate the specification of a class into a sampling algorithm, with a sampling rule associated to each combinatorial constructor. The case of Union and Product are described below. For Union, the Bernoulli choice relies on the evaluation of generating function. The magic strength

---

3. This combinatorial Boltzmann model is named after the Boltzmann model of statistical physics which assigns to each possible state of a system, a probability that is proportional to an exponential of its energy.
of Boltzmann model is that a Boltzmann sampler for a Product is obtained by two independent calls to Boltzmann samplers of its components.

\[ C = \text{Product}(A, B) \]

Procedure \( \Gamma C \) (\( x \) : real)

\[
\begin{align*}
U &:= \text{uniform}(0, 1) \\
\text{return} &\ (\Gamma A(x), \Gamma B(x)) \\
\text{if } U &< A(x)/C(x) \text{ then return } \Gamma A(x) \\
\text{else return } &\Gamma B(x)
\end{align*}
\]

The same principle of independent calls of the component sampler, applies to other combinatorial constructors: in the labeled universe, the sampling rules with parameter \( x \), for \( C = \text{Op}(A) \) (with \( \text{Op} \in \{\text{SEQ}, \text{SET}, \text{CYC}\} \)) consist in \( k \) independent calls to the sampler for \( A \), with the number of components \( k \) being distributed according to different probability laws (geometric, Poisson and log-series), with parameter \( A(x) \). Sampling in the unlabeled universe is more complicated because of symmetries, but similar algorithms were designed. In Boltzmann sampling, the generation process follows the syntactic decomposition of the object, so that arithmetic complexity is linear in the size of the output. This statement must be accompanied by two important remarks.

First, the evaluation of generating functions has to be done exactly, and in constant time (this "oracle" issue was solved in []). Second, as already mentioned, the size of the output is not fixed. If exact-size sampling is required, the cost of generation is quadratic (taking into account the cost of rejection, when the generated object is not of the exact given size). However, by tuning the parameter \( x \), one can target at a size range, and it can be shown ([?, ?]), using sophisticated methods of Complex Analysis, that the cost of generation stays linear (for a very large class of specifications) when a relative tolerance (10%, to 20%) on the size of the output is allowed.

In many applications approximate-size samplers are sufficient, since for testing or validating, only the order of magnitude of the size matters. The generation of objects with several millions of atoms in less than a second on a standard PC becomes in many cases possible. In this way, Boltzmann random generation can address applications in domains where intensive simulations are needed, such as software testing and complex networks modeling.

Following the seminal paper, many authors have been working on Boltzmann model and generation algorithms. The modeling capacity of Bolzmann models has been enriched in the direction of new operators, typically shuffle operator and languages ([]), holonomic systems and classes ([]), multiplicative structures ([]). Efficient and certified implementations have been designed, including the automatic resolution of the oracle for combinatorial systems ([]). Today, the most challenging issue is the development of Boltzmann models for the generation of non-uniform and constrained objects, the generation according to several simultaneous parameters, and complex notions of size.
2. Simulation of probability distributions

The question of generating random numbers according to nonuniform distributions, using a source of uniform random bits, goes back to the years 1950, when the emergence of faster computers made it possible to compute random numbers instead of obtaining them from tables. John von Neumann discussed an elegant method to simulate the exponential distribution using a sequence of independent uniform random variables, by exploiting the power series expansion of its density. As a result, the scheme only relies on comparisons of uniform variables and some bookkeeping.

The von Neumann scheme is an exact simulation method in the sense that, provided it has access to a perfect source of randomness, its output is distributed exactly according to the target distribution - no numeric computations are involved.

In the 1970s, Knuth and Yao [] picked up this theme of exact simulation of probability distributions and reformulated von Neumann’s scheme in terms of independent random bits. Since the target distribution has a density, “exact simulation” has to be understood in the sense that the algorithm outputs, in an interruptible manner, the bits of an exponentially-distributed random variate. Knuth and Yao experimentally observed that the expected number of consumed random bits before the algorithm is able to output $k$ bits is $k + O(1)$.

Over the years, the problem of generating (pseudo-)random numbers according to specific probability distributions - either continuous or discrete - received much attention from scientists interested in running simulations; see for instance [?] for a huge collection of techniques and examples. Most of the time, numeric methods using floating-point arithmetic and approximate special function evaluations are quite sufficient; in such context, the problem of generating an exponential random variate is rather trivial, since $|\ln(U)|$ is indeed exponentially distributed if $U$ is uniform over the interval $[0, 1]$ (and this is what the basic technique of the inverse distribution function suggests). The quality of the simulation is of course dependent on that of the original source of (pseudo-)random numbers, but also on that of the logarithm function which has to rely on floating-point arithmetic.

Philippe Flajolet’s work in the field of random number generation concentrated on combinatorial methods and exact simulation algorithms. The algorithms rely on analytic identities (of the individual probabilities for discrete distributions, or of densities or distribution functions for continuous distributions), and avoid floating point operations. While it may seem that such restrictions drastically reduce the power of the methods, the more recent work on Buffon machines shows a much wider than anticipated range of applicability.

The first contribution of Philippe Flajolet in this field was in, where he and Nasser Saheb studied the expected complexity of the von Neumann/Knuth-Yao algorithm for exponential variates. By noticing the similarity with the construction of a digital trie, they were able to provide a detailed analysis of the simulation scheme, thus confirming analytically the low expected complexity that was observed experimentally by Knuth and Yao.
This theme of exact simulation of probability distributions resurfaced in Philippe Flajolet’s work decades later with the idea of Buffon machines: low complexity, exact simulators using random bits as their source of randomness, this time for discrete probability distributions. Part of the motivation was to obtain random-bit-level generators for the discrete distributions used Boltzmann samplers. This goal was immediately attained as the geometric, Poisson and logarithmic distributions corresponding to sequence, set and cycle constructions proved amenable to Buffon simulation.