Algorithms Seminar 2001–2002,
F. Chyzak (ed.), INRIA, (2003), pp. 173–176.

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

Génération aléatoire†

Alain Denise
LRI, Orsay (France)

March 19, 2002

Summary by Sylvie Corteel

Abstract

Le but de ce résumé est de présenter brièvement les techniques de génération aléatoire. Nous nous concentrerons sur deux aspects : l'approche récursive et les chaines de Markov. Pour une vue plus générale et détaillée, nous conseillons la lecture du rapport d'habilitation d'Alain Denise [1] dont est inspiré ce résumé.

1. L'approche récursive

1.1. **Premiers formalismes.** En 1977, Wilf considère une famille de structures combinatoires dont la construction peut être représentée par un chemin dans un graphe orienté acyclique. L'exemple des sous-ensembles de cardinal k d'un ensemble $E = \{e_1, e_2, \ldots, e_n\}$ illustre le principe. Les étapes successives de la construction d'un tel sous-ensemble peuvent être représentées par un chemin dans le plan discret de longueur n et de hauteur k qui n'emprunte que des pas $s_1 = (+1, +1)$ et $s_0 = (+1, 0)$. Lorsqu'on est au point (i, j), le choix du pas s_1 détermine l'appartenance de l'élément e_i au sous-ensemble ; le choix de s_0 détermine sa non-appartenance. Notons $C_{n,k}$ l'ensemble de ces chemins et convenons d'appeler respectivement longueur et hauteur les entiers n et k. Tout chemin de $C_{i,j}$ est soit un pas s_0 suivi d'un chemin de $C_{i-1,j}$, soit un pas s_1 suivi d'un chemin de $C_{i-1,j-1}$. À chaque étape, on va donc choisir d'engendrer un pas s_0 avec probabilité $|C_{i-1,j}|/|C_{i,j}| = (i-j)/i$ et un pas s_1 avec probabilité $|C_{i-1,j-1}|/|C_{i,j}| = j/i$.

1.2. **Spécifications pour les structures décomposables.** En 1994, Flajolet, Zimmermann et Van Cutsem publient un schéma général de composition de structures combinatoires qui repose sur la notion de *spécification combinatoire*. Dans ce formalisme, les objets primitifs sont l'objet « vide » (de taille 0) noté 1, et un ensemble fini d'*atomes* de taille 1. Le symbole Z désigne un atome générique. Cinq opérateurs permettent de définir récursivement des ensembles d'objets combinatoires à partir d'autres ensembles et des deux types d'objets primitifs :

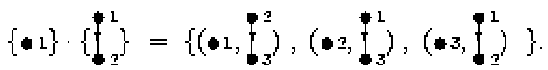
- L'union disjointe : $A \cdot B = \{\, i \mid i \in A \text{ ou } i \in B \,\}$.
- Le produit (non commutatif) : $A \cdot B = \{\, (a, b) \mid a \in A \text{ et } b \in B \,\}$.
- La séquence, l'ensemble et le cycle : `sequence`(A) (resp. `set`(A), `cycle`(A)) désigne l'ensemble des suites (resp. des ensembles, des cycles) finies d'éléments de A. Ces opérateurs peuvent être accompagnés d'un argument qui fixe une condition sur la cardinalité des suites (resp. ensembles, cycles).

†Notes de cours pour le cours donné pendant le groupe de travail ALÉA'02 au CIRM à Luminy (France).

Structures étiquetées	
Permutations	$P = \texttt{sequence}(Z)$ ou $P = \texttt{set}(\texttt{cycle}(Z))$
Partitions d'ensembles	$P = \texttt{set}(\texttt{set}(Z,\ \text{card} \geq 1))$
Surjections	$S = \texttt{sequence}(\texttt{set}(Z,\ \text{card} \geq 1))$
Structures non étiquetées	
Partitions d'entiers	$F = \texttt{set}(\texttt{sequence}(Z,\ \text{card} \geq 1))$
Compositions d'entiers	$F = \texttt{set}(\texttt{set}(Z,\ \text{card} \geq 1))$
Chemins de Dyck	$D = 1\ +\ Z \cdot D \cdot \bar{Z} \cdot D$

FIGURE 1. Quelques spécifications combinatoires.

Dans l'article initial, les objets sont *étiquetés* : à chaque atome d'un objet est associé un entier. Deux atomes différents ont toujours des étiquettes différentes. Cela implique que, lors des quatre dernières opérations ci-dessus, un réétiquetage des objets est effectué. Ainsi, par exemple, le produit n'est pas un produit cartésien, comme illustré ci-après :



Dans un travail ultérieur, les mêmes auteurs adaptent leur formalisme aux objets non étiquetés. Dans ce cas, les opérateurs restent le même et le produit est un produit cartésien. L'opérateur `set` pour les objets non étiquetés ne construit pas des ensembles, mais des multi-ensembles.

Soit $T = \{T_0, T_1, \ldots, T_m\}$ une famille de $m+1$ ensembles d'objets combinatoires (étiquetés ou non). Une *spécification combinatoire* de T est un ensemble de $m+1$ équations telles que la $i$ème (pour tout $0 \leq i \leq m$) s'écrit $T_i = \Psi_i(T_0, T_1, \ldots, T_m)$, où Ψ_i est une combinaison des cinq opérateurs définis ci-dessus appliquée aux T_i, à l'objet vide et aux atomes.

La Figure 1 présente quelques exemples de spécifications pour des objets combinatoires classiques.

Pour aboutir à des algorithmes de génération aléatoire efficace, il est nécessaire de transformer la spécification en *spécification standard*.

Soit $T = \{T_0, T_1, \ldots, T_m\}$ une famille de $m+1$ ensembles d'objets combinatoires étiquetés. Une *spécification standard* de T est un ensemble de $m+1$ équations telles que la $i$ème (pour tout $0 \leq i \leq m$) s'écrit $T_i = 1$ ou $T_i = Z$ ou $T_i = U_j + U_k$ ou $T_i = U_j \cdot U_k$ ou $\Theta T_i = U_j \cdot U_k$. Chaque U_j appartient à $\{1, Z, T_0, \ldots, T_m, \Theta T_0, \ldots, \Theta T_m\}$. Le symbole Θ désigne l'opérateur de *pointage* :

$$\Theta A \quad = \quad \bigcup_{n=1}^{\infty} (A_n \times \{1, 2, \ldots, n\})$$

où A_n est l'ensemble des objets de A de taille n.

1.3. **Algorithmes de génération.** L'algorithme de génération aléatoire se déduit de la spécification standard. Soit n la taille des objets à engendrer. La première étape est une étape de dénombrement : il s'agit de calculer, pour tout ensemble C intervenant dans la spécification standard et pour tout $0 \leq i \leq n$, le nombre c_i d'objets de taille i de C. On utilise les fonctions génératrices exponentielles (resp. ordinaires) pour les objets étiquetés (resp. non étiquetés). On déduit directement de la spécification standard les relations de récurrence pour les séries. Par exemple :

$$C = 1 \ \Rightarrow\ C(z) = 1\ ; \qquad C = Z \ \Rightarrow\ C(z) = z\ ;$$
$$C = A + B \ \Rightarrow\ C(z) = A(z) + B(z); \qquad C = A \cdot B \ \Rightarrow\ C(z) = A(z)B(z)\ .$$

	Dénombrement	Génération	Mémoire
Structures décomposables			
Cas général	$O(n(\log n)^2 \operatorname{loglog} n)$	$O(n \log n)$	$O(n)$
Cas holonome	$O(n)$	$O(n \log n)$	$O(1)$
Cas itératif	$O(n(\log n)^2 \operatorname{loglog} n)$	$O(n)$	$O(n)$
Langages algébriques	$O(n)$	$O(n \log n)$	$O(1)$
Langages rationnels	$O(n)$	$O(n)$	$O(1)$

TABLE 1. Quelques complexités.

Le nombre d'opérations arithmétiques effectuées au cours de l'étape de dénombrement est clairement en $O(n^2)$. Certaines classes d'objets possèdent une série génératrice *holonome* : il existe (et on sait calculer) une formule de récurrence linéaire à coefficients polynomiaux qui donne les termes de la série génératrice. Ceci implique que les termes jusqu'à l'ordre n peuvent être calculés en $O(n)$ opérations arithmétiques. Dans le cas général comme dans le cas holonome, $O(n)$ entiers doivent être stockés.

La phase de dénombrement n'est effectuée qu'une fois. Le processus de génération s'effectue de façon récursive. À chaque ensemble C d'objets représenté dans la spécification standard, on associe une procédure. Par exemple :

Cas : $C = 1$.
  gC := procedure(n: integer)
    if $n = 0$ then return(1)
  end

Cas : $C = A + B$.
  gC := procedure(n: integer)
    U:=Uniform($[0, 1]$);
    if $U < a_n/c_n$
      then return(gA(n))
      else return(gB(n))
  end

Cas : $C = Z$.
  gC := procedure(n: integer)
    if $n = 1$ then return(Z)
  end

Cas : $C = A \cdot B$.
  gC := procedure(n: integer)
    U:=Uniform($[0, 1]$);
    $k := 0$; $S := a_0 b_n/c_n$;
    while $U > S$ do
      $k := k + 1$; $S := S + a_k b_{n-k}/c_n$;
    return([gA(k),gB($n - k$)])
  end

Décrivons des moyens d'améliorer la complexité. Pour la complexité en temps, dans le cas $C = A \cdot B$, soit K la variable aléatoire qui représente la taille de l'élément de A, et $\pi_{n,k} = \mathbf{P}(K = k) = a_k b_{n-k}/c_n$. L'algorithme présenté ci-dessus ajoute successivement ces probabilités à S dans l'ordre $\pi_{n,0}, \pi_{n,1}, \pi_{n,2}, \ldots$ Considérons maintenant un algorithme qui effectue le même traitement, mais dans l'ordre suivant : $\pi_{n,0},\ \pi_{n,n},\ \pi_{n,1},\ \pi_{n,n-1},\ \ldots$ Cette variante est appelée « boustrophédon ». Cette seule modification donne une complexité arithmétique en $O(n \log n)$. Le principe de la preuve est extrêmement simple : soit $f(n)$ la complexité au pire. Elle satisfait une récurrence de type $f(n) = \max_{0\le k\le n}\bigl(f(k) + f(n-k) + 2\min(k, n-k)\bigr)$ dont la solution est en $O(n \log n)$.

Les *structures itératives* sont celles dont le graphe de dépendance (il existe un arc de l'ensemble A vers l'ensemble B s'ils sont respectivement dans le membre gauche et le membre droit de la même règle) de la spécification combinatoire est acyclique. Dans ce cas, la phase de génération est linéaire. Pour le cas holonome, Goldwurm a prouvé en 1995 que la génération d'un mot de longueur n peut s'effectuer en espace arithmétique $O(1)$, tout en conservant les complexités arithmétiques en $O(n)$ et $O(n \log n)$ respectivement pour les phases de dénombrement et de génération. Dans le cas général, Van der Hoeven a proposé en 1999 une méthode pour calculer les coefficients jusqu'à l'ordre n de toute série génératrice de structures décomposables en temps $0(M(n)\log n)$, où $M(n)$ désigne la complexité arithmétique de multiplication de deux polynômes de degré $n-1$. Le meilleur algorithme de multiplication présente une complexité arithmétique $M(n) = O(n\log n\log\log n)$. Les résultats sont regroupés en Table 1. Il existe de nombreuses autres techniques basées sur la récursivité : méthodes pour les langages algébriques, méthode paresseuse, méthodes à rejet, génération non uniforme controlée.

2. Chaînes de Markov

2.1. **Génération presque uniforme.** Si on peut déterminer en temps polynomial le nombre d'objets de taille n, au moins asymptotiquement, Jerrum, Valiant et Vazirani ont montré qu'il est toujours possible de concevoir un générateur aléatoire uniforme de complexité polynomiale. Il existe un grand nombre de structures combinatoires que l'on ne sait pas compter aussi facilement. Pour certaines, le problème de leur dénombrement est $\#P$-complet.

Pour ces cas difficiles, la génération aléatoire *presque* uniforme peut être envisagée. De plus, sous certaines conditions, un algorithme de génération presque uniforme peut mener à un algorithme probabiliste polynomial de dénombrement approximatif. Pour un ensemble E d'objets combinatoires, il s'agit de faire en sorte que la différence relative entre la probabilité $p(e)$ d'engendrer un objet $e \in E$ et la probabilité uniforme $p_u = 1/|E|$ soit inférieure à un réel ε fixé.

On considère une chaîne de Markov dont les éléments de E sont les états, et dont chaque transition est déterminée par une modification d'un objet. Si la chaîne est irréductible et apériodique et si la probabilité de transition de e à e' est égale à celle de la transition inverse pour tout e et tout e' de E, alors la loi du processus tend vers une unique distribution stationnaire uniforme. Donc, l'algorithme de génération consiste à partir d'un état quelconque et suivre les transitions avec les probabilités correspondantes. Le problème est de déterminer le temps suffisant pour que la génération soit uniforme à ε prés. S'il est polynomial en la taille n des objets à engendrer et en $\log(1/\varepsilon)$, on dit que la chaîne *se mélange rapidement.* Mais prouver ceci est loin d'être facile. Le premier résultat positif est dû à Jerrum et Sinclair, qui ont présenté en 1989 un algorithme polynomial pour la génération presque uniforme de couplages parfaits dans un graphe.

2.2. **Génération exactement uniforme.** Dans certains cas, l'approche par chaîne de Markov aboutit à une distribution exactement uniforme ; par exemple pour la génération d'arbres couvrants d'un graphe. Un arbre couvrant d'un graphe est sous-graphe connexe sans cycle qui contient tous les sommets du graphe. On sait compter les arbres couvrants d'un graphe, et plusieurs algorithmes de génération existent. Toutefois, l'algorithme présenté ici est à la fois extrêmement simple et plus performant que les précédents. Il a été découvert indépendamment par Aldous et Broder puis amélioré par Wilson. Le procédé consiste à construire l'arbre arête par arête comme suit : partir d'un sommet quelconque du graphe ; à chaque étape, choisir uniformément une arête adjacente au sommet courant et la traverser ; si elle n'appartient pas déjà à l'arbre et si elle n'y occasionne pas de circuit, alors l'ajouter à l'arbre ; stopper dès que l'arbre couvre tous les sommets. Le processus peut être vu comme une chaîne de Markov dont les états sont les sous-arbres du graphe enracinés au sommet courant. On montre qu'on obtient un arbre couvrant avec probabilité uniforme en complexité moyenne en $O(n\log n)$ pour presque tous les graphes, et $O(n^3)$ pour les pires d'entre eux.

Bibliography

[1] Denise (Alain). – *Structures aléatoires : Modèles et analyse des génomes.* – Habilitation à diriger des recherches, LRI, Université Paris-Sud, dec 2001. 76 pages.