Générateur de nombres pseudo-aléatoires

Un **générateur de nombres pseudo-aléatoires**, pseudorandom number generator (**PRNG**) en anglais, est un algorithme qui génère une séquence de nombres présentant certaines propriétés du hasard. Par exemple, les nombres sont supposés être approximativement indépendants les uns des autres, et il est potentiellement difficile de repérer des groupes de nombres qui suivent une certaine règle (comportements de groupe).

Cependant, les sorties d'un tel générateur ne sont pas entièrement aléatoires ; elles s'approchent seulement des propriétés idéales des sources complètement aléatoires. John von Neumann insista sur ce fait avec la remarque suivante : « Quiconque considère des méthodes arithmétiques pour produire des nombres aléatoires est, bien sûr, en train de commettre un péché ». De vrais nombres aléatoires peuvent être produits avec du matériel qui tire parti de certaines propriétés physiques stochastiques (bruit d'une résistance par exemple).

La raison pour laquelle on se contente d'un rendu pseudo-aléatoire est : d'une part qu'il est difficile d'obtenir de « vrais » nombres aléatoires et que, dans certaines situations, il est possible d'utiliser des nombres pseudo-aléatoires, en lieu et place de vrais nombres aléatoires ; d'autre part, que ce sont des générateurs particulièrement adaptés à une implémentation informatique, donc plus facilement et plus efficacement utilisables.

Les méthodes pseudo-aléatoires sont souvent employées sur des ordinateurs, dans diverses tâches comme la méthode de Monte-Carlo, la simulation ou les applications cryptographiques. Une analyse mathématique rigoureuse est nécessaire pour déterminer le degré d'aléa d'un générateur pseudo-aléatoire. Robert R. Coveyou du Oak Ridge National Laboratory écrivit dans un article que « la génération de nombres aléatoires est trop importante pour être confiée au hasard ».

La plupart des algorithmes pseudo-aléatoires essaient de produire des sorties qui sont uniformément distribuées. Une classe très répandue de générateurs utilise une congruence linéaire. D'autres s'inspirent de la suite de Fibonacci en additionnant deux valeurs précédentes ou font appel à des registres à décalage dans lesquels le résultat précédent est injecté après une transformation intermédiaire. Certains générateurs pseudo-aléatoires sont dits cryptographiques quand certaines contraintes sont satisfaites. Citons entre autres Fortuna, Yarrow ou Blum Blum Shub.

Sommaire

Problèmes expliquant le terme « pseudo-aléatoire »

Utiliser des algorithmes pour créer des nombres aléatoires, sachant ce qu'est un algorithme, peut paraître assez douteux. De plus, étant donné qu'on ne sait pas bien définir le caractère aléatoire, on se demande comment on peut ne serait-ce que vouloir produire une séquence aléatoire à l'aide d'algorithmes.

Ces reproches sont tout à fait fondés et on ne peut les nier. Ce sont les raisons pour lesquelles on parle de générateurs de nombres pseudo-aléatoires. En pratique, il est possible de limiter les conséquences négatives de l'utilisation d'une méthode algorithmique pour générer des nombres aléatoires.

Biais inhérent aux méthodes employées

Comme un générateur de nombres aléatoires est exécuté sur un ordinateur déterministe, il devient de facto un algorithme déterministe. Ses sorties sont inévitablement entachées d'une caractéristique absente d'une vraie suite aléatoire : la périodicité. Avec des ressources limitées (mémoire, nombre de registres, etc.), le générateur retrouvera le même état interne au moins deux fois. Après coup, il entrera obligatoirement dans un cycle. Un générateur non périodique n'est pas impossible, mais nécessite une mémoire croissante pour ne pas se retrouver dans le même état. Pour contourner cet obstacle théorique, le générateur peut commencer dans un état quelconque (la « graine »). Il produira toutefois la même suite si la graine reste identique. De plus, le nombre de graines possibles n'est pas infini et le générateur ne peut produire qu'un nombre limité de séquences différentes.

Pour améliorer la qualité des sorties, on peut introduire des composantes « aléatoires » provenant des imperfections du système, comme par exemple le temps entre deux accès au disque dur ou le nombre de millisecondes depuis le lancement de l'ordinateur. Ces valeurs sont

toutefois comprises entre certains intervalles et/ou sont en partie prévisibles. Le système reste pseudo-aléatoire.

Déviation par rapport au hasard parfait

La longueur de la période maximale double à chaque fois qu'un bit est ajouté à l'état interne. Il est facile de construire des générateurs pseudo-aléatoires avec une période plus longue que ce que n'importe quel ordinateur pourrait calculer. Mersenne Twister, un excellent générateur de nombres aléatoires pour les applications non cryptographiques, a une période prouvée mathématiquement de $2^{19937} - 1$, un nombre astronomique.

La question est ouverte quant à savoir s'il est possible de distinguer une suite pseudo-aléatoire générée algorithmiquement d'une source parfaite d'aléa, et ceci sans connaître la graine du générateur. En cryptographie, la plupart des spécialistes partent du principe que c'est une chose impossible avec la puissance de calcul actuelle. Ce principe est utilisé pour les chiffrements par flot comme RC4 qui procède à un XOR entre une suite pseudo-aléatoire et les données.

En principe, la plupart des générateurs ont des problèmes mathématiques qui peuvent être décelés avec une analyse statistique. La qualité des sorties augmente souvent au détriment de la vitesse de génération et ces paramètres doivent être considérés lors de l'utilisation d'un générateur. Les failles peuvent être les suivantes :

- période plus courte avec certaines graines
- qualité du générateur qui varie fortement selon la graine
- distribution imparfaite, manque d'uniformité
- mauvaise distribution dans un espace de dimension supérieure à 1
- ou au contraire : distribution trop idéale, uniformité trop parfaite
- valeurs successives qui ne sont pas indépendantes (ce qui est toujours le cas, sauf si on injecte des données, issues de sources aléatoires, dans une étape de la génération)
- certains bits dans les sorties sont moins aléatoires (par exemple, le bit n°8 reste souvent à 1)

Les défauts peuvent être importants ou quasiment invisibles. L'algorithme RANDU utilisé pendant plusieurs décennies était biaisé et certains résultats obtenus à l'époque ne peuvent être complètement validés. Il arrive parfois qu'une application pratique permette de déceler le problème (par exemple une simulation physique avec des résultats complètement inattendus), mais il est préférable de faire des tests avant utilisation pour les déceler.

Historique du développement des générateurs pseudo-aléatoires

Le développement des algorithmes générant des nombres pseudo-aléatoires est très lié à celui de la cryptographie, l'importance militaire et économique de cette science ayant motivé de nombreuses recherches au cours de l'histoire.

Les chiffrements utilisés traditionnellement jusqu'au XIX^{e} siècle reposaient essentiellement sur le secret autour de la méthode utilisée, et sur l'absence de traitement de masse. De nos jours, de telles méthodes sont impraticables car il existe de nombreuses théories statistiques qui permettent de retrouver l'algorithme de génération à partir de ses résultats. En outre, les techniques de chiffrement ne peuvent plus être gardées secrètes, et en 1883, Auguste Kerckhoffs exposera une règle fondamentale de la cryptographie moderne : la sécurité d'un système ne doit pas reposer sur la méconnaissance de la méthode de chiffrement. Cette règle, accompagnée par le développement des algorithmes de chiffrement par clé, marquera le début de l'essor des générateurs pseudo-aléatoires.

Leur importance est toutefois restée limitée tant que les moyens physiques de calcul n'ont pu supporter les longs et répétitifs calculs qu'ils nécessitent. C'est pourquoi leur émergence n'a vraiment commencé qu'en 1946, quand John Neumann publie son générateur middle-square.

C'est durant les années qui suivirent que la plupart des algorithmes rapides et populaires virent le jour : en 1948 D. H. Lehmer introduit les générateurs congruentiels linéaires qui seront améliorés en 1958 par G.J. Mitchell et D.P. Moore ; et deviendront par la suite extrêmement répandus, la plupart des fonctions de chiffrement basique y ayant recours.

Ces premiers générateurs pseudo-aléatoires possèdent malgré leur large popularité des propriétés statistiques assez mauvaises, et ne répondaient pas aux besoins des cryptographes. Plus récemment, des algorithmes robustes vis-à-vis des analyses statistiques ont été élaborés, comme l'algorithme Mersenne Twister (1997) ou encore la Méthode de Fibonacci lacunaire.

Mais aucun algorithme pseudo-aléatoire ne peut vraiment générer de suite à l'abri de toute analyse statistique, en particulier car la « graine » doit en théorie être elle-même aléatoire, et l'algorithme utilisé ne peut s'initialiser lui-même. Les générateurs cryptographiques actuels sont donc obligés de faire intervenir une part de hasard qui n'est pas générée par un moyen déterministe : on s'oriente vers des générateurs hybrides, possédant un algorithme de génération de nombres pseudo-aléatoires robuste, et s'initialisant sur un moyen physique de production de hasard.

Exemples d'algorithmes connus

La méthode de Von Neumann

En 1946, John von Neumann propose un générateur pseudo-aléatoire connu sous le nom de la méthode middle-square (carré médian). Très simple, elle consiste à prendre un nombre, à l'élever au carré et à prendre les chiffres au milieu comme sortie. Celle-ci est utilisée comme graine pour l'itération suivante.

Exemple

Soit le nombre « 1111 ».

1. $1111^2 = 1234321$
2. on récupère les chiffres du milieu : 3432. C'est la sortie du générateur.
3. $3432^2 = 11778624$
4. on récupère les chiffres du milieu : 7786.

et ainsi de suite.

Défauts

Von Neumann utilisa des nombres comportant 10 chiffres, le principe restant le même. Toutefois, la période du middle-square est faible. La qualité des sorties dépend de la graine, « 0000 » produit toujours la même séquence et constitue un « état absorbant » de l'algorithme. Von Neumann en était conscient, mais il craignait que des retouches a priori nécessaires n'apportent d'autres vices cachés. Sur l'ordinateur ENIAC qu'il utilisait avec sa méthode, il obtenait une génération 200 fois plus rapide que les résultats obtenus avec des cartes perforées. Selon Von Neumann, les générateurs basés sur du matériel ne pouvaient pas fonctionner correctement car il ne stockait pas les résultats (et on ne pouvait donc pas les vérifier). La méthode de Von Neumann montra vite ses limites lors d'applications utilisant des méthodes statistiques comme celle de Monte Carlo.

Méthode de Fibonacci

Cette méthode est basée sur la suite de Fibonacci modulo la valeur maximale désirée :

$x_n = (x_{n-1} + x_{n-2}) \ mod \ M$ avec x(0) et x(1) en entrée.

On peut employer une variante :

$x_n = (x_{n-1} + x_{n-k}) \ mod \ M$ avec x(1)....x(k-1) en entrée.

La qualité du générateur dépend de k et des nombres utilisés pour initialiser la suite. Ce générateur est par contre très simple à implémenter et ne consomme que peu de ressources .

Générateurs congruentiels linéaires

Article détaillé : Générateur congruentiel linéaire.

Introduits en 1948 par D. H. Lehmer sous une forme réduite (incrément nul), ils vont être généralisés et seront largement utilisés ensuite. Ils reposent sur une simple formule de récurrence :

$$X_{n+1} = (a \cdot X_n + c) \mod m$$

avec X_0 la graine (seed en anglais : un nombre employé pour produire une suite pseudo-aléatoire habituellement plus longue). En général, la graine est un nombre premier, mais les contraintes exactes à son sujet dépendent de l'algorithme . Certaines graines peuvent conduire à des séquences dégénérées.

La période de ce générateur est au maximum de m, c'est-à-dire qu'elle est relativement courte puisque m est souvent choisi de manière à être de l'ordre de la longueur des mots sur l'ordinateur (par exemple : 2^{32} sur une machine 32 bits). Mais cette méthode présente un avantage : on connaît les critères sur les nombres a, c et m qui vont permettre d'obtenir une période maximale (égale à m).

Exemples d'algorithmes

Algorithme	a	c	m	Remarques
RANDU	65539	0	2^{31}	Biaisé et fortement déconseillé
générateur de Robert Sedgewick	31415821	1	10^8	Intéressant mais déconseillé (essayer avec $X_0 = 0$)
Standard minimal	16807	0	2^{31}-1	

D'autres exemples utilisant les congruences

L'itération de Lehmer n'est pas la seule possible ; on peut notamment utiliser d'autres formes de congruence :

- $X_{n+1} = X_n(X_n + 1) \mod 2^e$ avec $X_0 \mod 4 = 2$
- $X_n = X_{n-j} + X_{n-k} + c \mod m$ (pour j, k fixés)

En particulier : $X_n = X_{n-24} + X_{n-55} \mod m$ proposée en 1958 par G.J. Mitchell et D.P. Moore a passé avec succès de nombreux tests statistiques ; on sait par ailleurs que cette suite possède une très longue période. Son plus gros défaut est l'absence de théorie la concernant . Ici encore, les valeurs initiales sont cruciales . Si les X_n sont nuls pour tout n appartenant à l'ensemble initial, alors la suite produira toujours un résultat nul.

Mersenne Twister

Article détaillé : Mersenne Twister.

Inventé par Makoto Matsumoto et Takuji Nishimura en 1997, Mersenne Twister est particulièrement réputé pour sa qualité. Avec sa période de 2^{19937}-1 itérations, il distribue de manière uniforme sur 623 dimensions (pour des nombres de 32 bits) et s'avère être plus rapide que la plupart des méthodes statistiquement plus faibles. Il est toutefois possible d'analyser la sortie de Mersenne Twister et de se rendre compte qu'il n'est pas entièrement aléatoire. L'algorithme de Berlekamp -Massey ou l'algorithme de Reed-Sloane permettent de répondre à cette question. À ce jour, les seuls effets négatifs sont liés à la cryptographie, Mersenne Twister ne pouvant accéder au titre de générateur cryptographique.

Générateurs pseudo-aléatoires cryptographiques

Certains générateurs pseudo-aléatoires peuvent être qualifiés de cryptographiques quand ils font preuve de certaines propriétés nécessaires pour qu'ils puissent être utilisés en cryptologie. Ils doivent être capables de produire une sortie suffisamment peu discernable d'un aléa parfait et doivent résister à des attaques comme par exemple l'injection de données forgées de manière à produire des imperfections dans l'algorithme, ou encore des analyses statistiques qui permettraient de prédire la suite.

Dans la catégorie des générateurs suffisamment sûrs, on trouve :

- Yarrow
- Fortuna
- Blum Blum Shub (sûr mais lent)
- ISAAC

Une méthode courante pour générer de l'aléa en cryptographie consiste à « accumuler de l'entropie » (via diverses sources disponibles sur un ordinateur : temps entre deux accès au disque, taille de la mémoire, mouvements du pointeur de la souris…) et à faire passer le résultat dans une fonction de hachage cryptographique comme MD5 ou SHA-1. Ce principe est utilisé par Yarrow et Fortuna qui ne génèrent un nombre que lorsque l'entropie est suffisante.

Domaines d'application

Les générateurs pseudo-aléatoires ont, à travers le temps, acquis une grande importance dans divers domaines, allant du médical à la sécurisation, et se sont montrés être d'une grande efficacité quant à leurs apports dans ces domaines.

Confidentialité des échanges sur les réseaux sans fils

L'échange par réseaux sans fil pose de nombreux problèmes concernant la confidentialité des éléments échangés. Ceci dit, il faut alors un mécanisme qui peut conserver le mieux possible cette confidentialité, ce mécanisme est alors le WEP pour Wired Equivalent Privacy. Le principe consiste à définir une clé chiffrée sur une longueur allant de 40 à 128 bits. Ensuite, la clé est déclarée au niveau du point d'accès et du client. Le but de la clé est de créer un nombre pseudo aléatoire d'une longueur égale à celle de la trame de transmission. Le nombre pseudo aléatoire ainsi généré sert à chiffrer la transmission, et alors à assurer la confidentialité de cette dernière.

Toutefois, le WEP n'assure malheureusement pas une sécurité optimale, les compagnies Fluhrer et Shamir ont montré que la génération de la chaîne pseudo aléatoire rend possible la découverte de la clé de session en stockant 100 Mo à 1 Go de trafic créé intentionnellement. 24 bits de la clé sont dédiés à l'initialisation, ce qui veut dire que chiffrer à 128 bits est de loin meilleur que de chiffrer à 64 bits dans la mesure où chiffrer à 128 bits offre une possibilité réelle de chiffrer à 104 bits, tandis que chiffrer a 64 bits n'en offre que 40 bits.

Sécurisation des applications web

Les applications WEB qui ont pour utilisateurs les navigateurs WEB ont un système qui sert à protéger leurs clients par la création de sessions. Les sessions sont des espaces de travail qui ont un espace de stockage d'informations connu, elles sont aussi privées et appartiennent à un utilisateur particulier dûment authentifié.

Néanmoins, ces sessions ne présentent pas toujours le niveau de sécurité souhaité. C'est ici qu'interviennent les nombres pseudo aléatoires. L'exemple le plus connu est celui d'attaque par session de type prédiction (méthode de détourner ou de personnifier un utilisateur WEB, elle s'accomplit par la déduction ou l'estimation de la valeur unique qui identifie la session associée). Pour éviter ce genre d'attaques, il est devenu courant d'utiliser un gestionnaire d'applications WEB capable de générer des identifiants qu'on ne peut pas prévoir. Il n'y a plus à chercher, car les applications de génération de nombres pseudo aléatoires ont été suffisamment développées pour ce faire.

Système de chiffrement

La cryptanalyse est une discipline récente, et signe un grand niveau de fiabilité quant aux méthodes de chiffrement qu'elle utilise, notamment le chiffrement à flot. Ce denier consiste à additionner bit à bit au message clair une suite binaire pseudo aléatoire de même longueur appelée suite chiffrante. Ce procédé est reconnu pour sa rapidité et sa compétitivité vis-à-vis des autres procédés dans la mesure où il est peu sensible aux erreurs introduites lors de la transmission. Des registres à décalage avec rétroaction linéaire font souvent partie des générateurs pseudo aléatoires utilisés pour produire les suites chiffrantes.

Domaine biomédical

Une application importante au niveau biomédical concerne l'amélioration de la modélisation du dépôt de dose dans le corps et aussi pour accélérer le calcul de traitement. En effet, sur les grilles de calcul, il est impératif que chaque processus de calcul puisse disposer d'une sous séquence aléatoire issue d'une séquence globale de nombres aléatoires à générer. L'un des fruits des recherches menées à ce sujet, est la plate-forme de simulation Monte-Carlo GATE, qui est un logiciel de calcul de dépôt de dose sur plusieurs grappes, et qui a permis de mettre en évidence des facteurs d'accélération de l'ordre de 30 du temps de calcul en comparaison avec une utilisation en milieu hospitalier.

Une application originale : le GPS

Les codes pseudo-aléatoires connaissent une application originale dans le cadre du système de localisation par satellites GPS. Ils y sont en effet utilisés non pas en fonction de leur prévisibilité vis-à-vis d'une analyse extérieure pour protéger des données, mais parce qu'ils présentent une séquence connue, facile à calculer, et qui n'a aucune similitude avec les séquences obtenues en changeant de graine.

Ces caractéristiques sont utiles aux GPS à cause de l'utilisation qui en est faite : Le but est de mesurer une distance entre deux équipements, en l'occurrence un satellite de la constellation GPS et un récepteur.

Cette mesure est réalisée en générant une séquence pseudo-aléatoire sur chacun des équipements, que l'on supposera synchronisés, et générant la séquence à la même vitesse. La séquence du satellite est alors modulée et transmise au récepteur, qui la compare a sa propre séquence. Il y a alors un décalage entre les deux chaînes, et ce décalage correspond au temps de propagation de l'information du satellite au récepteur. Il suffit alors de mesurer ce décalage, de le multiplier par la vitesse des ondes électromagnétiques dans l'air et on obtient la distance recherchée.

Ce principe est toutefois contrarié dans la réalité par de multiples facteurs qui conditionnent le choix du générateur pseudo-aléatoire à utiliser :

- Les séquences générées par les codes pseudo-aléatoires ne sont pas infinies et se répètent périodiquement, la période dépendant du code utilisé et de la machine sur laquelle il est implanté. Ce « défaut » entraîne une ambiguïté sur la distance mesurée par le récepteur : le décalage mesuré est de t, mais il peut aussi être de t+T (où T est la période de la séquence), de t-T, ou de tout nombre entier de périodes. Il est a priori impossible de lever cette ambiguïté sans traitement informatique qui relie la mesure de distance réalisée a la position du récepteur estimée précédemment. C'est pourquoi une caractéristique importante des codes pseudo-aléatoires utilisés dans le système GPS est leur période, qui doit être la plus longue possible : l'un des codes utilisés, appelé code P, possède une longueur d'onde supérieure au diamètre de la terre, ce qui fait disparaître de problème de l'ambiguïté.

- La vitesse de génération du code par les équipements conditionne la précision de la mesure de distance. En effet, la précision ne pourra jamais être plus grande que la distance élémentaire obtenue en multipliant le temps nécessaire a calculer ou émettre un bit par la vitesse de la lumière. En d'autres termes, on ne peut pas être plus précis que la distance parcourue par le signal du satellite pendant l'émission d'un bit du code. Il est donc primordial de posséder un code généré extrêmement rapidement, et dont la formule de calcul soit la plus simple possible. À titre d'exemple, pour obtenir une précision idéale de l'ordre du mètre, il faudrait générer un bit toutes les 3 nanosecondes.

- La méthode de mesure du décalage entre les deux codes est réalisée par une fonction de corrélation. La première caractéristique que cela impose au code est d'avoir une autocorrélation aussi proche que possible d'un pic de Dirac : si l'autocorrélation du code est importante quand il y a décalage de phase, il deviendra impossible de mesurer le décalage car il y aura ambiguïté entre deux possibles. De même, si entre deux satellites la corrélation des codes n'est pas faible, leurs signaux se perturberont mutuellement et cela rendra la mesure difficile voire irréalisable.

L'ensemble de ces facteurs ont conduit les constructeurs du système GPS à utiliser les codes de Gold. Ces codes, basés sur la combinaison de deux séquences binaires, présentent des caractéristiques intéressantes pour le GPS : ils sont relativement faciles à calculer, et possèdent des périodes appropriées. Le point fort des codes de Gold est leur excellente réponse au 3^{e} critère : la corrélation de deux codes est faible, que ce soient deux codes décalés ou différents, et le seul cas ou la corrélation est forte est celui de deux mêmes codes en phase. De ce fait, ils sont très utilisés en télécommunications pour réaliser le multiplexage par codes ou CDMA.

Voir aussi

- hasard
- probabilité
- entropie
- GPS

Références

- Donald E. Knuth, The Art of Computer Programming, Volume 2: Seminumerical Algorithms, 3rd edition (Addison-Wesley, Boston, 1998).
- J. Viega, Practical Random Number Generation in Software (http://www.acsac.org/2003/papers/79.pdf), in Proc. 19th Annual Computer Security Applications Conference, Dec. 2003.
- John von Neumann, "Various techniques used in connection with random digits," in A.S. Householder, G.E. Forsythe, and H.H. Germond, eds., Monte Carlo Method, National Bureau of Standards Applied Mathematics Series, 12 (Washington, D.C.: U.S. Government Printing Office, 1951): 36-38.
- Jeremy Lainé et Laurent Leconte, Le Système GPS (http://www.jerryweb.org/jeremy/ea_gps.pdf), École polytechnique, Dec. 2001

- P. Bernard , V. Breton , M. Fogli , D. Hill, J-P. Reveillés : Rapport de fin de projet retenu dans le cadre de la campagne ACI GRID – Projet GLOP . Décembre 2005

Liens externes

- **(en)** The GNU Scientific Library (http://www .gnu.org /software /gsl/) . Une bibliothèque scientifique en C distribuée sous licence GPL .
- **(en)** How computers generate random numbers ? (http://members .cox .net/srice 1/random / introduction .htm) , par David W. Deley. Comment les ordinateurs génèrent-ils des nombres aléatoires ?

Générateurs de nombres pseudo -aléatoires

Rapides : Générateur congruentiel linéaire, Mersenne Twister, RANDU

Cryptographiques : Blum Blum Shub, Fortuna, ISAAC, Yarrow

Briques : Congruence sur les entiers, Fonction de hachage, Registre à décalage

Voir aussi le portail de la cryptologie