Liczby losowe

Krzysztof Słonka

II rok, Informatyka
Wydział Informatyki, Elektroniki i Telekomunikacji
Akademia Górniczo-Hutnicza im. St. Staszica w Krakowie

Spis treści

1. Liczby losowe

1. Definicja

Termin „liczba losowa” czy też „liczby losowe” nie jest prostym do zdefiniowania pojęciem, pomimo tego, iż na pierwszy rzut oka, może tak się wydawać.

Najogólniej mówiąc, jest to taka liczba która została wygenerowana na podstawie procesów losowych, czyli takich które są nieprzewidywalne i niemożliwe do odtworzenia.

Najważniejszą częścią takiej definicji nie jest sama liczba a proces w którym ona powstała, dlatego niemożliwe jest posiadając tylko jedną liczbę określić czy została ona wygenerowana w sposób losowy. Potrzebna jest większa porcja danych (ciąg), która umożliwi zbadanie danego procesu.
Sprawdzenie czy sekwencja danych jest losowa czy nie, także nie jest prostym zadaniem. Możliwe jest sprawdzić czy dana sekwencja posiada takie same właściwości statystyczne jak sekwencja losowa. Jednak to, że dany proces wygeneruje bity 00000 czy bity 01010 nie świadczy o jakości wygenerowanej sekwencji, ponieważ prawdopodobieństwo ułożenia takich bitów jest sobie równie prawdopodobne.

W związku z powyższymi problemami pojawiła się potrzeba ścisłego zdefiniowania tego co charakteryzuje sekwencję losową w praktyce. Wg Donalda Knutha jest to **sekwencja niezależnych liczb które reprezentują dany rozkład prawdopodobieństwa** (rozkład normalny, logarytmicznie normalny itd.) w zadanym przedziale liczbowym. Wg Bruce'a Schneiera jest to **sekwencja która posiada takie same właściwości jak losowy ciąg, jest nieprzewidywalna i nie daje się w prosty sposób powtórzyć**.

2. Testowanie losowości

Głównymi sposobami testowania losowości jest sprawdzanie statystycznych właściwości danego ciągu. Jednym z przykładów może być test częstotliwości występowania zer i jedynek (po prze-konwertowaniu ciągu do postaci binarnej). Wraz z wzrostem próbki danych ilość zer i jedynek powinna być równa, jeżeli tak nie jest, można stwierdzić, że dana sekwencja faworyzuje jedną z wartości. Kolejnym przykładem jest test współzależności, który sprawdza czy nie występują zależności pomiędzy bitami przystającymi do siebie, testuje to przewidywalność danego procesu (algorytmu) wykorzystywanego do generowania licz losowych. Innym przykładem może być test sprawdzający odległości pomiędzy zerami itd.

2. Generatory liczb losowych

1. Prawdziwie losowe

Prawdziwie losowe liczby można otrzymać jedynie poprzez zjawiska fizyczne takie jak szum termiczny, efekt fotoelektryczny, czy zjawiska występujące w fizyce kwantowej, ponieważ procesy w nich występujące, teoretycznie, są całkowicie nieprzewidywalne. Implementacja sprzętowa generatora takich liczb zazwyczaj składa się z przetwornika który zamienia mierzone

zjawisko w sygnał elektryczny który następnie wzmacniany jest poprzez wzmacniacz a na końcu konwertowany jest postaci binarnej.

Procesy kwantowe wykorzystywane w tego typu generatorach to

- szum śrutowy
- rozpad promieniotwórczy
- fotony przechodzące przez półprzeźroczyste lustro
- wzmocnienie spowodowane przez bazę tranzystora pn spolaryzowanego zaporowo

Inne procesy fizyczne

- szum termiczny
- przebicie elektryczne
- szum atmosferyczny

Wchodząc na stronę https://qrng.anu.edu.au/RainBin.php możemy otrzymać ciąg losowych liczb wygenerowanych w laboratoriach Narodowego Uniwersytetu Australii, które są otrzymywane za pomocą zjawisk kwantowych, a także przeglądnąć wszystkie testy jakie są wykonywane na tych danych.

2. Pseudolosowe

Generator liczb pseudolosowych to algorytm który na podstawie informacji wejściowych generuje ciąg bitów który pod pewnymi względami jest nieodróżnialny od ciągu uzyskanego za pomocą generatora prawdziwie losowego.

LCG - Linear congruential generator

Jednym z najbardziej znanych i najlepiej przestudiowanych jest LCG. Jego zaletami są prostota działania i łatwość implementacji. Generator przyjmuje następujące wartości początkowe:

- m – wartość modulo
- a, 0 < a < m – mnożnik,
- c, 0 < c < m – przyrost,
- X_0 , 0 < X_0 < m – wartość początkowa

Jego wartość obliczana jest za pomocą następującej zależności rekurencyjnej $X_{n+1} \equiv (aX_n + c) (mod\ m)$. Rozróżniamy dwa podstawowe rodzaje generatorów LCG:

- Addytywny LCG : $X_n = (a\, X_{n-1} + c)\, mod\ m$
- Multiplikatywny LCG: $X_n = a\, X_{n-1}\, mod\ m$

Podstawowa różnica pomiędzy nimi jest taka, iż generator addytywny LCG może generować liczby pseudolosowe z zakresu od 0 do m - 1, a generator multiplikatywne generuje je z zakresu od 1 do m – 1.

Zalety i wady LCG:

- mała ilość pamięci wymagana do przechowywania stanu generatora
- szybkość

- mniej znacząca część wygenerowanej liczby ma o wiele krótszy okres niż bardziej znacząca część
- występuje problem zależności stanu poprzedniego od następnego

W praktycznych realizacjach dąży się do dużych okresów generatora LCG – wtedy liczby pseudolosowe powtarzają się dopiero po wielu miliardach przebiegów. Jako przykład niech posłuży poniższy generator LCG zaproponowany przez prof. D. Knutha:

m = 34359738368 = 2^{35}
a = $\pi \cdot 10^9$
c = $e \cdot 10^9$ e – podstawa logarytmów naturalnych

Generator Fibonacciego

Jest to generator korzystający z ciągu Fibonacciego $X_n = X_{n-1} + X_{n-2} \bmod m$ .Generuje on ciąg deterministyczny tzn. dla takich samych danych wejściowych generuje taki sam ciąg pseudolosowych liczb. Ma o wiele lepsze parametry jakościowe od innych generatorów liniowych, jednak wymaga więcej czasu do generacji.

Możemy także uogólnić wzór generatora do $X_n = X_{n-j} @ X_{n-k} \bmod m, 0 < j < k$. Korzystając z faktu, iż wyrazy składające się na kolejną liczbę ciągu nie są już sztywno ustalone, oraz operacja dodawania została zamieniona na ogólną operację można otrzymać wiele kombinacji tego generatora. Jedne z bardziej znanych są to:

- ALFG - Addytywny Opóźniony Generator Fibonacciego (ang. Additive Lagged Fibonacci Generator) - gdy operacja @ zdefiniowana jest jako dodawanie.
- MLFG - Multiplikatywny Opóźniony Generator Fibonacciego (ang. Multiplicative Lagged Fibonacci Generator) - gdy operacja @ zdefiniowana jest jako mnożenie.

Generator ten musi przechowywać wartości k ostatnich elementów, także jest bardziej wymagający jeśli chodzi o aspekt pamięciowy.

RC-4

Jest to popularny szyfr strumieniowy używany w takich systemach jak WEP, WPA, BitTorrent, SSL, SSH, Remote Desktop Protocol itd. Jest odporny na kryptoanalizę liniową oraz różnicową.

RC4 generuje pseudolosowy strumień bitów. Szyfrowanie i odszyfrowywanie następuje poprzez wykonanie operacji XOR na tekście oryginalnym i strumieniu szyfrującym. Najważniejszym z punktu widzenia tego artykułu jest sposób generowania losowego strumienia szyfrującego. Na początku generowana jest tablica S, za pomocą dostarczonego klucza.

```
for i from 0 to 255
   S[i] := i
   j := 0
   for i from 0 to 255
     j := (j + S[i] + klucz[i mod długość_klucza]) mod 256
     zamień(S[i],S[j])
```

Następnie wykonywany jest algorytm modyfikujący stany tablicy S który generuje jeden

bajt strumienia szyfrującego, tak długi jak oryginalny tekst. Za każdym razem algorytm zwiększa i o 1, dodaje do j wartość w S wskazywaną przez i, zamienia ze sobą wartości S[i] i S[j] oraz jako wynik daje wartość z S wskazywaną przez S[i] + S[j] (modulo 256). Każda wartość S jest

```
i := 0
j := 0
while TworzonyStrumieńSzyfrujący:
   i := (i + 1) mod 256
   j := (j + S[i]) mod 256
   zamień(S[i],S[j])
   wynik S[(S[i] + S[j]) mod 256]
```

zamieniana co najmniej raz na każde 256 iteracji.

RC4 zapewnia wystarczająco dużo bezpieczeństwa przy szyfrowaniu informacji jeżeli jest użyty z hashowaniem wektora inicjalizującego oraz kodem uwierzytelniania wiadomości.

3. Zastosowania generatorów liczb losowych

1. Metoda Monte-Carlo

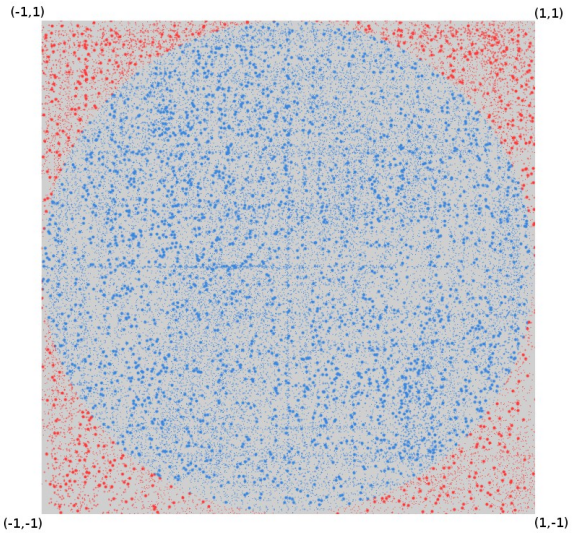
Jest to klasa algorytmów polegająca na obliczeniu wyników za pomocą wielokrotnego próbkowania używając losowego wejścia. Stosowana jest często do modelowania procesów matematycznych zbyt złożonych do obliczenia za pomocą podejścia analitycznego. Metoda Monte-Carlo może różnić się w zależności od danego problemu jednak zazwyczaj przyjmuje następujący szkielet:

1) Definicja możliwych danych wejściowych
2) Generowanie danych wejściowych zgodnie z znanym rozkładem prawdopodobieństwa
3) Przeprowadzenie deterministycznych obliczeń na danych wejściowych
4) Łączenie wyników

Przykładem może być wyliczenie wartości π :

1) Losujemy n punktów na przedziale (-1,1) x (-1,1)
2) Sprawdzamy czy dany punkt spełnia własność
   $x^2+y^2\leq 1$ jeżeli tak to zwiększamy licznik k

$$\pi = 4 \cdot \frac{k}{n}$$



Metoda Monte-Carlo **nie zawsze wymaga prawdziwie losowych generatorów liczb**, jednak w niektórych aplikacjach takich jak testowanie pierwszości jest to wymagane. Używanie deterministycznych generatorów wartości losowych pozwala na otrzymywanie powtarzalnych wyników. Najważniejszymi cechami którymi powinny charakteryzować się dane wejściowe są zgodność z zadanym rozkładem prawdopodobieństwa, długi okres, duża ilość danych.

2. Kryptografia

Dlaczego losowe liczby są potrzebne w kryptografii?

- Generowanie kluczy publicznych, prywatnych, kluczy sesji
- Generowanie paddingu przy szyfrowaniu blokowym
- Hasła i numer PIN
- Liczby pierwsze
- Znaczniki jednorazowe (ang nonce)

Jednym z ciekawszych ataków jaki został przeprowadzony, w którym wykorzystano słabość wygenerowanych liczb losowych był atak brute-force na sesje PHP na portalu facebook.com. Sesja PHP to 160 bitowy ciąg składający się z:

- 32 bitowego adresu IP
- 32 bitowej wartości Epoch
- 32 bitowa wartość mikrosekund
- 64 bitowa wartość uzyskana z generatora LCG

Atakującemu udało się zdobyć adres IP ofiary poprzez wysłanie go na swoją stronę internetową i sprawdzenie logów serwera www, wartość Epoch (składająca się z roku, miesiąca, dnia itd.) jest możliwa do otrzymania poprzez sprawdzenie pakietów które wysyła chat strony facebook.com do wszystkich użytkowników których znajomi znajdują się na wyżej wymienionym czacie. Wartość milisekund nie jest możliwa do zgadnięcia, jednak milisekundy to tylko wartości z zakresu od 0 do 999 999 które zajmują mniej niż 20 bitów. Przy generowaniu wartości lcg wykorzystane były wartości Epoch oraz numer procesu, które za pomocą odpowiednich zabiegów mogą być albo skrócone (tak jak wartość milisekund) albo zgadnięte całkowicie. Za pomocą odpowiedniego oprogramowania atakującemu **udało się zredukować 160 bitów entropii do 20 bitów** które mogą być złamane używając ataku brute-foce przy średnio **500 000 zapytań** do serwera www.

Przykład ten pokazuje jak bardzo ważne jest dobieranie odpowiednich danych wejściowych do naszego generatora liczb pseudolosowych lub też jak bardzo ważne może być stworzenie własnego mechanizmu sesji a co za tym idzie własnego generatora liczb losowych. Po wykonaniu tego ataku, został wysłany raport do programistów PHP którzy zmienili algorytm generowania liczby losowej.

4. Bibliografia

1. Knuth Donald, The Art of Computer Programming, Vol. 2, 2nd ed., Addison-Wesley, Reading, (1981).
2. Schneier Bruce, Applied Cryptography: Protocols, Algorithms, and Source Code in C, John Wiley & Sons, New York, (1996).
3. R. Davies, Hardware random number generators. Int. 15th Australian Statistical Conference, Jul. (2000).
4. Michael Luby, Pseudorandomness and Cryptographic Applications, Princeton University Press, ISBN 9780691025469
5. Autor zbiorowy, Generator liczb pseudolosowych, Strona internetowa, 29 listopada 2012, Creative Commons http://pl.wikipedia.org/wiki/Generator_liczb_pseudolosowych
6. Tomasz Lubiński, Generator LFG (Opóźniony Generator Fibonacci'ego), Strona internetowa, 29 września 2008, http://www.algorytm.org/liczby-pseudolosowe/generator-lfg-opozniony-generator-fibonacciego.html
7. Tomasz Mostowski, Metoda Monte Carlo i jej zastosowania, Prezentacja, 31 marca 2008, http://coin.wne.uw.edu.pl/~tmostowski/pliki/matlab/matlab7.pdf
8. Samy Kamar, How I Met Your Girlfriend: The discovery and execution of entirely new classes of Web attacks in order to meet your girlfriend, Strona internetowa, 27 października 2010, http://www.youtube.com/watch?v=fEmO7wQKCMw
9. Australian National University, Strona internetowa, https://qrng.anu.edu.au/RainBin.php