Wprowadzenie do symulacji i metod Monte Carlo 10.10.2019

Projekt nr 1

Paweł Lorek

1 Opis projektu

Opis pojęć pojawiających się poniżej podany jest w następnych rozdziałach. Generator liczb pseudolosowych będzie oznaczany przez PRNG. Dużo rzeczy, które są tutaj opisane występuje (być może trochę inaczej sformułowane) w skrypcie prof. Rolskiego: http://www.math.uni.wroc.pl/~rolski/Zajecia/sym.pdf

- Będziemy rozpatrywać wyniki działań następujących PRNG (większość do samodzielnego zaimplementowania, opisane w Rozdziale 4) Jeśli ziarno nie zostało wyspecyfikowanie - wybór należy do Ciebie.
  - LCG$(13, 1, 5)$
  - LCG$(2^{10}, 3, 7)$
  - GLCG$(2^{10}, \{3, 7, 68\})$
  - "state" (Matlabowski)
  - "twister" (Matlabowski)
    a) z ziarnem $u_0 = 0$
    b) z ziarnem $u_0 = 1812433253$
  - Excellowski:
    $$u_i = (0.9821 u_{i-1} + 0.211327) \bmod 1$$
  - RC4(32), rozpatrz dwie wersje:
    i) Zainicjalizuj jednym kluczem (np. jakąś funkcją obecnego czasu itp) i następnie badaj wynik PRGA, który powinien być tak długi jak potrzebujesz.
    ii) Skonkatenuj krótsze (ustal długość) wyniki PRGA dla kluczy [1] $K = 0, 1, 2, \ldots$

  Dodatkowo będziemy rozpatrywać binarne rozwinięcia liczb $\sqrt{2}, e, \pi$ dostępne pod adresem:

  π: http://www.math.uni.wroc.pl/~rolski/Zajecia/data.pi

  e: http://www.math.uni.wroc.pl/~rolski/Zajecia/data.e

  $\sqrt{2}$: http://www.math.uni.wroc.pl/~rolski/Zajecia/data.sqrt2

[1] związane jest to z tzw. "key-related attacks", zob. https://en.wikipedia.org/wiki/Related-key_attack

- **ZADANIE**.

  1) Dla RC(32) i minimum 3 innych generatorów (w tym min. jednego, który nie jest opisany w tym dokumencie) wykonaj wybrane testy statystyczne. Rozważ używanie testu χ^2 (licząc statystykę $\hat{\chi}^2$), testu Kołmogorowa-Smirnova (licząc statystykę $\hat{D}_n$), testu serii, testu odstępów dni urodzin, testu kolizji. Generatory porównuj poprzez porównanie stosownych p-wartości.
  2) Dla liczb $\pi, e, \sqrt{2}$ wykonaj *Frequency monobit test*
     a) dla całych ciągów (wyliczając jedną p-wartość)
     b) każdy ciąg podziel na m części (na każdej zapuszczając test, otrzymując łącznie m p-wartości). Zobacz opis w Rozdziale 3.2 (możesz przyjąć $\alpha = 0.05$).

  W powyższym punkcie 1) wykorzystaj min 3 testy, w tym także min. 1 test, który nie jest opisany w tym dokumencie. Rozważ jeden z testów dostępnych w pakiecie `DieHarder`, opisane są na stronie:

  https://sites.google.com/site/astudyofentropy/background-information/the-tests/dieharder-test-descriptions

  albo występują w `Nist Test Suite`, a opisane są w dokumencie [4] dostępnym również pod adresem:

http://www.math.uni.wroc.pl/~lorek/teaching/files/nistspecialpublication800-22r1a.pdf

  Całość opisz w raporcie w formacie .pdf. Raport powinien zawierać wyjaśnienie wszystkich użytych rzeczy (tak, aby ktoś niechodzący na wykład mógł zrozumieć całość) oraz wnioski.

2 O dwóch statystykach

Opiszemy krótko dwa testy, χ^2 oraz test Kołmogorowa-Smirnova. Generator liczb pseudolosowych będzie w skrócie oznaczany przez PRNG.

2.1 Test χ^2

Załóżmy, że mamy n obserwacji (funkcja ciągu generowanego przez PRNG), każda jest z jednej z k możliwych kategorii. Niech Y_s oznacza liczbę obserwacji z kategorii s, oraz niech p_s będzie prawdopodobieństwem "wpadnięcia" do kategorii s. Dla dużych n oczekujemy, iż

$$Y_s \approx np_s.$$

Przy założeniu, że obserwacje są niezależne i każda rzeczywiście wpada do kategorii s z prawd. p_s statystyka

$$\hat{\chi}^2 = \sum_{i=1}^{k} \frac{(Y_i - np_i)^2}{np_i}$$

ma rozkład χ^2 z $k-1$ stopniami swobody, co będziemy oznaczać przez $\chi^2(k-1)$.

2.2 Test Kołmogorowa-Smirnova

Załóżmy, że zmienna losowa X ma rozkład ciągły o dystrybuancie $F_X(x)$. Załóżmy także, że mamy n obserwacji $X_1, \ldots, X_n$ i chcemy testować czy są one niezależne i pochodzą z rozkładu $F_X(x)$. Zdefiniujmy *dystrybuantę empiryczną* $F_n(x)$ jako

$$\hat{F}_n(x) = \frac{1}{n} \sum_{i=1}^{n} \mathbf{1}(X_i \leq x)$$

Twierdzenie Gliwienki-Cantellego mówi, że jeśli $X_1, \ldots, X_n$ jest próbką z rozkładu F_X, to

$$\sup_{x \in \mathbb{R}} |\hat{F}_n(x) - F(x)| \to 0, \quad n \to \infty,$$

z prawdopodobieństwem 1. Co więcej, statystyka

$$\hat{D}_n = \sqrt{n} \sup_{x \in \mathbb{R}} |\hat{F}_n(x) - F(x)|$$

dąży, gdy $n \to \infty$, do znanego rozkładu:

$$Pr(\hat{D}_n \leq t) \to H(t) = 1 - \sum_{i=1}^{\infty} (-1)^{i-1} e^{-2i^2 t},$$

H jest dystrybuantą rozkładu Kolmogorova-Smirnova.

Dla rozkładu dyskretnego, sposób wyliczania D_n podany jest w [3].

3 Testowanie generatorów liczb pseudolosowych poprzez testy empiryczne

Istnieje wiele testów dla PRNG. Można je podzielić na dwie kategorie: **testy empiryczne** oraz **testy teoretyczne**. Testy teoretyczne wymagają wiedzy o działaniu samego generatora, ale sam ciąg nie musi de facto być generowany, dobre są zazwyczaj trudne do znalezienia. Z kolei testy empiryczne są wykonywane na ciągach otrzymanych z PRNG, żadna wiedza o tym jak zostały wygenerowane nie jest potrzebna.

Testy wymagają liczb o rozkładzie na odcinku $(0, 1)$ albo o rozkładzie dyskretnym na $\{0, 1, \ldots, M\}$. W zależności od generatora, jego wynikiem może również być albo ciąg liczb z przedziału (0,1) albo liczby dyskretne. Niech $U_1, U_2, \ldots$ oznaczają liczby z przedziału (0,1), wtedy możemy je przekształcić na liczby dyskretne ze zbioru $\{0, 1, \ldots, M\}$ poprzez

$$Y_i = \lfloor MU_i \rfloor.$$

Podobnie - dla liczb dyskretnych Y_i z przedziału $\{0, 1, \ldots, M\}$ możemy otrzymać liczby z (0,1) poprzez

$$U_i = \frac{Y_i}{M}.$$

3.1 p-wartość

W kontekście $\hat{\chi}^2$: możemy testować hipotezę:

H_0 obserwacje są niezależne, każda jest z kategorii s z prawd. p_s

H_1 obserwacje są nie pochodzą z takiego rozkładu

Jakie jest prawd., że jeśli obserwacje rzeczywiście są niezależne i pochodzą z omawianego rozkładu to zaobserwowalibyśmy coś $\geq \hat{Y}$? Jest to

$$p = Pr(\chi^2(k-1) > \hat{\chi}^2)$$

(co odczytujemy z tablic). Małe p-wartości oznaczają, iż prawdopodobieństwo zaobserwowania $\hat{\chi}^2$, lub czegoś większego, jest małe. Ustalając poziom ufności na α (typowo 0.05 lub 0.01) odrzucimy hipotezę H_0 jeśli $p \leq \alpha$.

W kontekście zmiennej losowej N o rozkładzie normalnym $N(0,1)$ p-wartość liczona jest jako (gdzie s jest zaobserwowaną statystyką) $p = Pr(|N| > |s|)$

3.2 Testy empiryczne

Idea testowania: dla określonych zdarzeń, których prawdopodobieństwa znamy (zakładając jednostajność) liczymy statystyki $\hat{\chi}^2$ oraz $\hat{D}_n$ (lub inne) i wyliczamy ich p-wartości. Mamy dwa podejścia:

- Dla całego ciągu wyliczmy daną p-wartość. Generatory porównujemy porównując p-wartości.
- (tzw. testy "second-level"). Dzielimy ciąg na m części, każda długości n. Na każdej wyliczamy statystykę i stosowne p_i-wartości, $i = 1, \ldots, m$. Rozkład otrzymanych p_i jest jednostajny (przy założeniu, że testowane bity/liczby pochodzą z rozkładu jednostajnego). Testujemy to (zalecenia z Rozdziału 4.2.2 "Nist Test Suite" [4]) za pomocą testu ξ^2 dzieląc odcinek $[0,1]$ na dziesięć części: Y_i niech oznacza liczbę obserwacji z przedziału $[(i-1)/10, i/10), i = 1, \ldots, 10$. Wówczas

  $$\hat{\chi}^2 = \sum_{i=1}^{10} \frac{(Y_i - n/10)^2}{n/10}$$

  ma rozkład $\chi^2(9)$ (przy założeniu, p-wartości $p_i, i = 1, \ldots, m$ pochodzą z rozkładu jednostajnego). Ostatecznym wynikiem jest "p-wartość p-wartości":

  $$p_{\chi^2} = P(X > \hat{\chi^2}),$$

  gdzie X jest zmienną losową o rozkładzie $\chi^2(9)$.

3.2.1 Frequency monobit test

Załóżmy, że $X_1, X_2, \ldots,$ są niezależnymi zmiennymi losowymi o jednakowym rozkładzie $Pr(X_i = -1) = Pr(X_i = 1) = 1/2$. Zauważmy, że $EX = 0, VarX = 1$. Zdefiniujmy

$$S_n = \frac{1}{\sqrt{n}} \sum_{i=1}^{n} X_i$$

Z CTG mamy, iż dla dużego n rozkład S_n jest w przybliżeniu $N(0,1)$. Dlatego, p-wartością jest

$$p = Pr(|N| > |S_n|) = 2(1 - \phi(|S_n|),$$

gdzie N jest zmienną losową o rozkładzie $N(0,1)$, a ϕ jego dystrybuantą.

3.2.2 Frequency Test

Jeden z podstawowych testów. Mając dany ciąg $U_1, \ldots, U_n$ możemy

- zastosować test K-S z dystrybuantą $F_X(x) = x, x \in (0,1)$.
- zastosować test χ^2 w następujący sposób: ustalmy M i wyliczmy $Y_i = \lfloor MU_i \rfloor$. Mamy M kategorii, $p_i = \frac{1}{M}$.

3.2.3 Test serii

Podobny do *Frequency test*, ale sprawdzane jest czy występowanie par elementów. Załóżmy, że $U_i \in \{0, 1, \ldots, M-1\}$. Podzielmy ciąg na podciągi długości $2n$:

$$(U_1, U_2), (U_3, U_4), \ldots, (U_{2n-1}, U_{2n}).$$

Policz wystąpienie każdej z par $(q, r), 0 \leq q, r < M$ i zastosuj test χ^2 z $M^2 - 1$ stopniami swobody.

Test może być łatwo rozszerzony na badanie dowolnych krotek (zob. skrypt [3]).

3.2.4 Test odstępów dni urodzin

Zapiszmy dni urodzin kolejnych osób $Y_1, \ldots, Y_n \in \{1, \ldots, k\}$ i ustawmy je w porządku niemalejącym $Y_{(1)} \leq \ldots \leq Y_{(n)}$. Zdefiniujmy odstępy

$$S_1 = Y_{(2)} - Y_{(1)}, \ldots S_{n-1} = Y_{(n)} - Y_{(n-1)}.$$

Niech K będzie liczbą równych odstępów (tzn. ile razy mamy równość pomiędzy S-ami)

- Okazuje się, że jeśli n jest duże i $\lambda = n^3/(4k)$ małe, to przy założeniu o równomiernym rozkładzie Y_i (hipoteza $\mathcal{H}_0$) zmienna K ma w przybliżeniu rozkład Poissona z parametrem λ. Jednym z wyborów jest $n = 2^{10}, k = 2^{24}$, wtedy $\lambda = 16$. Zatem, jeśli $K = y$, to p-wartość wynosi

$$Pr(K \geq y | \mathcal{H}_0) \approx 1 - \sum_{j=0}^{y-1} \frac{\lambda^j}{j!} e^{-\lambda}$$

- Knuth [1] sugeruje $k = 512 = 2^9$ oraz $n =^{25}$. Prawdopodobieństwa liczby równych odstępów podane są w Fig. 1. Używając tych prawdopodobieństw można zastosować test χ^2.

s	0	1	2	≥ 3
$Pr(K = s)$	0.368801	0.369035	0.183471	0.078692

Figure 1: Prawd. dla testu odstępów dni urodzin, $k = 2^{25}, k = 2^9$

3.2.5 Test kolizji

Zob. skrypt [3].

3.2.6 Test pokerowy

Zob. skrypt [3].

4 Generatory liczb pseudolosowych

Tutaj nalezaloby krotko wypisac podstawowe, tak by studenci mogli zaprogramowac (glownie LCG) z roznymi parametrami, by byly tez "kiepskie".
Moze tutaj wersja RC4 ? (krotki opis)

4.1 LCG(M, a, c)

Generatory LCG (z ang. *linear congruential generator*) zmieniają swój stan zgodnie z rekurencją

$$x_n = (as_{n-1} + c) \bmod M$$

Stan początkowy: x_0

4.2 GLCG$(M, \{a_i\}_{i=1}^k)$

Generatory te zmieniają stan zgodnie z rekurencją

$$x_n = (a_1 x_{n-1} + \ldots a_k x_{n-k}) \bmod M$$

Stan początkowy: $x_0, x_1, \ldots, x_{k-1}$

4.3 RC4(n)

RC4 (w opisie będziemy pomijali (n)) jest tzw. szyfrem strumieniowym[2], który może być wykorzystany jako PRNG. Oznaczmy $[n] := \{0, 1, \ldots, n-1\}$.
Jego tzw. *stanem wewnętrznym* jest (S, i, j), gdzie S jest permutacją $[n]$, a $i, j \in [n]$ są indeksami. Jako wejście algorytm bierze klucz: L liczb, każda z $[n]$ (możemy myśleć, iż klucz jest dodatkowym parametrem naszego PRNG). Następnie:

[2] http://pl.wikipedia.org/wiki/Szyfr_strumieniowy

- KSA (z ang. *key scheduling algorithm*) inicjuje S permutacją identycznościową, następnie używając klucza K zmienia S w inną permutację (w zamyśle *losową*
- PRGA (z ang. *Pseudo Random Generation Algorithm*) zwraca elementy tej permutacji jednocześnie ją updateując.

KSA oraz PRGA przedstawione są na Fig. 2. W oryginalnym RC4 mamy $n = 256$, natomiast liczby zwracane przez PRGA są "XORowane" z wiadomością otrzymując kryptogram. RC4 używany w trybie PRNG po prostu zwraca wyjście PRGA.

`KSA(`K`)`

for $i := 0$ to $n - 1$ **do**
  $S[i] := i$
end for

$j := 0$
for $i := 0$ to $n - 1$ **do**
  $j := j + S[i] + K[i \bmod L]$
  swap$(S[i], S[j])$
end for
$i, j := 0$

`PRGA`

while $r \in \mathcal{N}_+$ **do**
  $i := i + 1$
  $j := j + S[i]$
  swap$(S[i], S[j])$
  $Y_r \leftarrow S[S[i] + S[j]]$
end while

Figure 2: RC4: KSA i PRGA. Wszystkie dodawania są wykonywane mod$n$

Uwaga. *RC4 z $n = 256$ był do niedawna intensywnie używany. Jeśli w KSA zamienimy linijkę $j := j+S[i]+K[i \bmod L]$ na $j :=$ **random**(n), to można o tym algorytmie myśleć jako o tasowaniu kart. W kroku i-tym zamieniamy kartę na pozycji i z kartą na losowej pozycji. Jest to tzw. tasowanie Cyclic-To-Random, o którym wiadomo, iż potrzebne jest $O(n \lg n)$ kroków, by uzyskać jednostajną permutację. KSA wykonuje ich jednak tylko n, co jest jednym z jego poważnym mankamentów.*

References

[1] D. E. Knuth. *The Art of Computer Programming, Volume 2 (3rd Ed.): Seminumerical Algorithms,*. Addison-Wesley, 1997.

[2] P. L'Ecuyer, R. Simard, TestU01: A c library for empirical testing of random number generators, *ACM Trans. Math. Sofw.* 33(4), 2007.

[3] T. Rolski, *Symulacje stochastyczne i teoria Monte Carlo*, skrypt UWr, 2017. http://www.math.uni.wroc.pl/~rolski/Zajecia/sym.pdf

[4] Lawrence E. Bassham, Andrew L. Rukhin, Juan Soto, James R. Nechvatal, Miles E. Smid, Stefan D. Leigh, M Levenson, M Vangel, Nathanael A. Heckert, D L. Banks , *A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications*, 2010, https://www.nist.gov/publications/statistical-test-suite-random-and-pseudorandom-number-generators-cryptographic