Tampereen yliopisto
Sivuaineen tutkielma

Ville Helenius

Pseudosatunnaislukualgoritmit

Tietojenkäsittelytieteiden laitos
Tietojenkäsittelyoppi
Elokuu 2003

Sisältö

Johdanto

Tämän tutkielman tarkoitus on tutkia näennäis- eli pseudosatunnaislukujen generointia. Tutkielmassa pyritään vastaamaan sellaisiin kysymyksiin, kuten mihin generattoreita tarvitaan, miten niitä toteutetaan ja millaisia ominaisuuksia niillä on.

Lukijalle olisi hyväksi, jos tällä olisi perustiedot todennäköisyyslaskennasta ja algebrasta, erityisesti lukuteoriasta ja äärellisistä kunnista. Tämän lisäksi tietojenkäsittelyopista olisi hyvä tuntea algoritmien analyysiä sekä laskennan teoriaa.

Luvussa 1 selvitetään käyttettyjä merkintöjä ja määritelmiä, joita tarvitaan tutkielman sujuvaan lukemiseen.

Luvussa 2 tutustutaan tavallisimmin käytettyihin satunnaislukugeneraattoreihin yleisellä tasolla ja mietitään millaisia eroavaisuuksia niillä on. Luvussa tutustutaan muun muassa lineaarisen kongruenssin generaattoreihin (LCG), käänteisen kongruenssin generaattoreihin (ICG), potenssigeneraattoreihin (RSA,BBS), diskreetin eksponentin generaattoriin ja Tausworthe generaattoriin.

Luvussa 3 tutkitaan hieman syvällisemmin erityisesti lineaarisen kongruenssin generaattoreita sekä esitellään muutamia esimerkkejä.

Luvussa 4 tarkastellaan erilaisten generaattoreiden paremmuuden mittaamista tilastotieteen näkökulmasta käymällä läpi tavallisimman tilastolliset testit eli χ^2-testi, autokorrelaatio, Gap-testi sekä Run-testi.

Luvussa 5 käsitellään kryptologisessa mielessä hyviä generaattoreita ja tutustutaan kryptologisen satunnaislukugeneraattorin määritelmään. Lisäksi tutustutaan yksisuuntaisten funktioden määritelmään, jolla on yhteyttä $\mathcal{NP}$-ongelmaan. Lopuksi esitellään hyviä ehdokkaita yksisuuntaisiksi funktioiksi.

Luvussa 6 tutustutaan satunnaislukujen tavanomaiseen käyttötarkoitukseen eli simulointiin, erityisesti Monte Carlo menetelmään.

Satunnaislukuja tarvitaan erityisesti simulointiin, kun halutaan tuntemattomat tekijät, joiden jakauma tunnetaan tai satunnaistekijöitä ottaa mukaan simuloitavana olevaan malliin. Kryptografiassa satunnaislukuja tarvitaan muodostamaan lyhyestä jonosta bittejä pitempi jono bittejä, joilla voidaan sanoma salata. Tällöin on tärkeätä, että generoidusta jonosta ei voida liian helposti laskea alkuperäistä lyhyttä bittijonoa.

Yleensä puhutaan satunnaisluvuista vaikka tarkoitetaan pseudosatunnaislukuja. Tällöin muodostetaan bittijono, joka näyttäisi olevan sattumanvarainen eli ei ole löydettävissä selkeää yksinkertaista muotoa taikka jaksoa jonosta.

Kuinka tällaisia jonoja voisi muodostaa? Ajatellaan lukua $\pi = 3,14159\ldots_{10}$. Luvun π binääriesitys on $11,0010010000111\ldots_2$. Luvulla π ei näyttäisi olevan selvää muotoa bittijonossa, joten lukua π voisi ehkäpä käyttää pseudosatunnaislukuna. Ongelmatonta tämä ei kuitenkaan ole, sillä ei tiedetä, onko π niin sanottu *normaaliluku* eli luku, jonka jokainen bittilohko esiintyy yhtä suurella tiheydellä. Tämä ongelma on yhä avoin. Vaikka π olisikin normaaliluku, niin seuraava ongelma on luvun π laskeminen. Toki π osataan laskea monituisin tavoin, mutta käytännön sovellutusten kannalta nämä ovat tehottomia.

Käytännössä joudutaan tyytymään äärellisiin psedosatunnaislukuihin. Koska näillä ei ole samanlaisia asymptoottisia ominaisuuksia kuin äärettömillä jonoilla, niin seuraavat ominaisuudet ovat tärkeitä.

Ensimmäiseksi, pseudosatunnaislukugeneraattorin täytyy olla laskennallisesti tehokas, jotta se olisi käytännöllinen.

Toiseksi, koska generaattori on äärellinen, täytyy generaattorin tuottaa mahdollisimman pitkän sattumanvaraiselta vaikuttavan bittijonon.

Kolmanneksi, generaattorilla täytyy olla tarpeeksi hyvät tilastolliset ominaisuudet. Näitä tavanomaisesti ovat tilastollinen riippumattomuus sekä generaattorin täytyy näennäisesti noudattaa haluttua todennäköisyysjakaumaa.

1 Määritelmiä ja merkintöjä

Tässä luvussa esitellään määritelmiä ja merkintöjä.

Määritelmä 1.1.
Luonnollisten lukujen joukkoa merkitään $\mathbf{N} = \{0, 1, 2, 3, \ldots\}$.
Kokonaislukujen joukkoa merkitään $\mathbf{Z} = \{\ldots, -2, -1, 0, 1, 2, 3, \ldots\}$.
Alkulukujen joukkoa merkitään $\mathbf{P} = \{2, 3, 5, 7, 11, \ldots\}$.
Rationaalilukujen joukkoa merkitään $\mathbf{Q}$.
Reaalilukujen joukkoa merkitään $\mathbf{R}$.
Kompleksilukujen joukkoa merkitään $\mathbf{C}$.
Kokonaislukujen joukko modulo m *merkitään* $\mathbf{Z}_m$.

Määritelmä 1.2 *Merkintä* $a|\, b$ *tarkoittaa, että luku* a *jakaa luvun* b. *Merkintä* $a \nmid b$ *tarkoittaa, että luku* a *ei jaa lukua* b.

Määritelmä 1.3. *Merkintä* $a \equiv b \pmod{c}$ *tarkoittaa, että* $c|\, a - b$. *Tällöin sanotaan, että luvut* a *ja* b *ovat kongruentteja modulo* c. *Lukua* c *sanotaan moduliksi.*

Määritelmä 1.4. *Merkkijonolla tarkoitetaan joukon* $\{0,1\}^k,\ k \in \mathbf{Z}_+$ *alkiota. Merkkijonon* x *pituus* $\|x\|$ *on* k, *mikäli* $x \in \{0,1\}^k$. *Tyhjästä merkkijonosta käytetään merkintää* Λ.

Määritelmä 1.5. *Kaikkien merkkijonojen joukko on*

$$\{0,1\}^* = \bigcup_{k=0}^{\infty} \{0,1\}^k.$$

Määritelmä 1.6. *Lukujen* a, b *suurin yhteinen tekijä merkitään* $\mathrm{GCD}(a, b)$. *Lukujen* a, b *pienin yhteinen jaettava merkitään* $\mathrm{LCM}(a, b)$.

Määritelmä 1.7. *Merkitsemme tapahtuman* $x \in A$ *todennäköisyyttä*

$$P(x \in A).$$

Määritelmä 1.8. *Mikäli satunnaismuuttuja x noudattaa todennäköisyysjakaumaa f, niin merkitsemme $x \sim f$.*

Määritelmä 1.9. *Mikäli satunnaismuutuja x noudattaa tasajakaumaa parametrein minimi a, maksimi b, niin merkitsemme $x \sim Tas(a, b)$.*

Määritelmä 1.10. *Mikäli satunnaismuutuja x noudattaa normaalijakaumaa parametrein odotusarvovektori μ, kovarianssimatriisi Σ, niin merkitsemme $x \sim N(\mu, \Sigma)$.*

Määritelmä 1.11. *Mikäli satunnaismuutuja x noudattaa χ^2-jakaumaa vapausastein n, niin merkitsemme $x \sim \chi^2(n)$.*

Määritelmä 1.12. *Olkoon $a, b \in \{0, 1\}$. Määritellään laskutoimitukset $\odot, \oplus$ seuraavasti*

$\oplus$	0	1
0	0	1
1	1	0

sekä

$\odot$	0	1
0	0	0
1	0	1

.

Huomautus. Binäärioperaatiot $\odot, \oplus$ ovat itseasiassa tutut operaatiot AND, XOR. Toisaalta nämä laskutoimitukset voidaan määritellä myös jakojäännösaritmetiikan avulla eli $a \odot b$ on yhtä kuin $a \cdot b \bmod 2$ ja $a \oplus b$ on yhtä kuin $a + b \bmod 2$.

Määritelmä 1.13. *Olkoon f, g funktioita $\mathbf{N} \to \mathbf{N}$. Merkitsemme*

$$f(n) = \mathrm{O}(g(n)),$$

mikäli on olemassa sellainen vakio $c > 0$, että $f(n) \leq c \cdot g(n)$.

Määritelmä 1.14. *Olkoon f funktio $\mathbf{N} \to \mathbf{R}$. Sanomme, että funktio on **mitätön**, mikäli*

$$\lim_{n \to \infty} n^k \cdot f(n) = 0, \forall k \in \mathbf{N},$$

ja merkitsemme $f(n) = \mathrm{NEGL}(n)$.

Huomautus. Jos funktio on mitätön, niin ei ole olemassa sellaista $k \in \mathbf{N}$,

että $1/f(n) = \mathrm{O}(n^k)$. Siis funktio on mitättömämpi kuin minkä tahansa polynomin residuaali eli käänteisluku.

Määritelmä 1.15. *Merkitsemme satunnaismuuttujan x odotusarvoa $\mathrm{E}(x)$, varianssia $\mathrm{Var}(x)$ ja kahden satunnaismuutujan x, y kovarianssimatriisia merkitään $\mathrm{Cov}(x, y)$.*

Määritelmä 1.16. *Äärellisen joukon S alkioiden lukumäärä merkitään $|S|$.*

Määritelmä 1.17. *Eulerin funktio φ määritellään seuraavasti*

$$\varphi(P) = \left|\{x \in \{1, 2, \ldots, P-1\} : \mathrm{GCD}(x, P) = 1\}\right|.$$

Määritelmä 1.18. *Käytämme lukujonoon $x_1, x_2, \ldots$ merkintää (x_n).*

Määritelmä 1.19. *Luvun k kertoma $k!$ määritellään*

$$k! = k \cdot (k-1) \cdots 2 \cdot 1.$$

Määritelmä 1.20. *Satunnaistettu polynomisessa ajassa laskettava algoritmi on sellainen, että se voidaan suorittaa polynomisessa ajassa Turingin koneella, joka saa syötteekseen ongelman lisäksi satunnaissyötteen tai -syötteitä.*

Huomautus. Itse asiassa kyseessä ovat luokkaan $\mathcal{NP}$ kuuluvat algoritmit.

2 Tavallisimmat pseudosatunnaislukugeneraattorit

2.1 Lineaarisen kongruenssin generaattorit

Lineaarisen kongruenssin generaattorit perustuvat jakojäännös aritmetiikkaan ja ovat muotoa

$$x_{n+1} \equiv a_0 x_n + a_1 x_{n-1} + \cdots + a_j x_{n-j} + b \pmod{P}, \tag{1}$$

missä kertoimet $a_0, a_1, \ldots a_j$ ja b ovat ei-negatiivisia ja niillä ei ole yhteisiä tekijöitä luvun P kanssa. Tällainen generaattori vaatii alkuarvot $x_0, x_1, \ldots, x_j$. Näiden alkuarvojen oletetaan myös olevan luvun P kanssa jaottomia.

Koska alkuarvoja on $j + 1$ kappaletta, jotka voivat saada arvot välillä $0, 1, \ldots, P-1$, niin tällainen generaattori tuottaa korkeintaan P^{j+1} eri satunnaislukua. Koska generaattoreilta vaaditaan ominaisuus, että ne ovat mahdollisimman pitkiä, niin täytyy valita joko j tai P mahdollisimman suureksi. Matemaattisesti ajateltuna luvun P ollessa suuri on helpompi tarkistaa, jakautuuko sarja tasaisesti välille $[0, P-1]$. Toisaalta tällöin yleensä sarjan hilarakenne jää heikoksi ja joissakin sovelluksissa tästä seuraa ongelmia. Jos taas j valitaan suureksi, niin on kovin vaikeaa sanoa, kuinka pitkä tai tasaisesti jakutunut jono (x_n) on.

Simuloinnissa lineaarisen kongruenssin generaattori on varsin tavallisesti käytetty. Kryptografiaan tälläinen generaattori ei sovellu helpon ennustettavuuden vuoksi. Tälläinen generaattori saadaan kuitenkin toimimaan erinomaisesti joissakin sovelluksissa kunhan parametrit $a_0, a_1, \ldots, a_j$ ja alkuarvot $x_0, x_1, \ldots, x_j$ valitaan huolella. Tarkastelemme tätä ongelmaa lähemmin luvussa 3.

Tavallisimmat tällaiset generaattorit, joiden toiminta tunnetaan ovat niin sanottu multiplikatiivisen kongruenssin relaatio, joka on muotoa $x_{n+1} \equiv ax_n + b \pmod{P}$, sekä Fibonacci-generaattori, joka nimensä mukaisesti on muotoa $x_{n+1} \equiv x_n + x_{n-1} \pmod{P}$.

2.2 Käänteisen kongruenssin generaattorit

Käänteisen kongruenssin generaattori on muotoa

$$x_{n+1} \equiv a\overline{y}_n + b \pmod{P}, \tag{2}$$

missä P on alkuluku ja luku $\overline{c}$ on luvun c multiplikatiivinen inverssi modulo P eli $c \cdot \overline{c} \equiv 1 \pmod{P}$.

Kun luvuksi P valitaan alkuluku, niin jokaiselle luvulle $1, 2, \ldots P-1$ löytyy inverssi. Tämä inverssi löytyy ratkaisemalla lineaarinen Diofantoksen yhtälö $c \cdot x + P \cdot y = 1$. Kuten tunnetaan, ratkaisut x, y saadaan käyttämällä Eukleideen algoritmia. Siis $\overline{c} = x$. Luvut a ja b valitaan siten, että luku a on nollasta eriävä ja luvut a, b ovat keskenään jaottomia.

Tällaisella generaattorilla tiedetään yleensä olevan lineaarista generaattoria parempi hilarakenne. Toisaalta tämä generaattori on hieman hitaampi kuin lineaarisen kongruenssin generaattori. Tämä johtuu tietysti tarpeesta ratkaista jokaisen generoinnin yhteydessä Diofantoksen yhtälö.

2.3 Potenssigeneraattorit

Potenssigeneraattorit ovat muotoa

$$x_{n+1} \equiv x_n^d \pmod{P}. \tag{3}$$

Tästä generaattorista on kaksi erityistapausta, jotka tekevät siitä kryptografisesti vahvan. Nämä ovat RSA-generaattori sekä Blum-Blum-Shub-generaattori (BBS-generaattori).

RSA-generaattori saa nimensä keksijöidensä Rivest, Shamir ja Adelman mukaan. RSA-generaattori toimii siten, että valitaan luvuksi P kahden suuren alkuluvun tulo $p_1 \cdot p_2$. Lisäksi luvuksi d valitaan sellainen luku, että $GCD(d, \varphi(P)) = 1$. Tiedetään, että $\varphi(P) = (p_1 - 1)(p_2 - 1)$, kun $P = p_1 \cdot p_2$. Lisäksi tiedetään, että

$$x_n^{\varphi(P)+1} \equiv x_n \pmod{P}$$

riippumatta luvusta x_n. Merkitään nyt

$$s = \frac{\varphi(P) + 1}{d}.$$

Tällöin

$$x_{n+1}^s \equiv x_n \pmod{P}.$$

Jotta luku s voidaan laskea, tarvitaan tieto $\varphi(P)$. Tämän laskeminen palautuu luvun P tekijöihin jakoon, jonka suorittamiseen ei toistaiseksi tunneta tehokasta algoritmia. Siis luvusta x_{n+1} ei voi päätellä lukua x_n. Lukujen d, ja $\varphi(P)$ täytyy olla keskenään jaottomia, koska tällöin myös luvut d^k $(k = 0, 1, 2, \ldots)$ ovat. Tämä tieto tarvitaan faktaan

$$x_n \equiv x_0^{d^n} \pmod{P}.$$

BBS-generaattori on potenssigeneraattori, missä $d = 2$ ja $P = p_1 \cdot p_2$ sekä $p_1, p_2 \equiv 3 \pmod{P}$. Myös tämän generaattorin ajatellaan olevan kryptografisesti varma.

2.4 Diskreetin eksponentin generaattori

Diskreetin eksponentin generaattori on muotoa

$$x_{n+1} \equiv g^{x_n} \pmod{P}. \tag{4}$$

Tällaisella generaattorilla on eräs tärkeä erikoistapaus, kun P on suurehko alkuluku ja g on primitiivijuuri modulo P. Tällaisia tilanteita on yhtä monta kuin on alkulukuja eli äärettömästi, sillä jokaista alkulukua kohti löytyy ainakin yksi primitiivijuuri. Tämän generaattorin ajatellaan olevan kryptografisesti vahva, sillä luvun x_n ratkaiseminen luvusta x_{n+1} vaatisi niin sanotun diskreetin logaritmin ottamista luvusta x_{n+1}. Kuitenkin toistaiseksi ei ole löydetty tehokasta algoritmia kyseisen probleeman ratkaisemiseksi. Palaamme tähän generaattoriin vielä luvussa 5.

2.5 Tausworthe-generaattorit

Tausworthe generaattori on itse asiassa erikoistapaus lineaarisen kongruenssin generaattorista. Tausworthe-generaattoriksi kutsutaan sitä erikoistapausta, joka saadaan modulin P arvolla 2.

Siis generaattori on muotoa

$$x_{n+1} \equiv a_0 x_n + \cdots + a_j x_{n-j} + b \pmod{2}.$$

Erikoiseksi generaattorin tekee se, että se on helposti laskettavissa ja toteutettavissa tietokoneella tai loogisiin piireihin. Tämä johtuu siitä, että

$$x_{n+1} = a_0 \odot x_n \oplus \cdots \oplus a_j \odot x_{n-j} \oplus b.$$

Kyseessä on siis yksinkertainen j-syötteen looginen piiri.

2.6 Äärelliset kunnat generaattoreina

Äärellisiä kuntia voidaan käyttää satunnaislukujen generointiin samaan tapaan kuin lineaarisen kongruenssin generaattoria. Itse asiassa lineaarisen kongruenssin generaattori on äärellisen kunnan erikoistapaus, kun moduli P on alkuluku.

Muutama fakta äärellisistä kunnista:

Lause 2.6.1. *Äärellisten kuntien alkioiden lukumäärät ovat alkulukujen potensseja.*

Lause 2.6.2. *Samankokoiset äärelliset kunnat ovat isomorfisia. Puhummekin äärellisistä kunnista niin sanottuina Galois'n kuntina. Merkitsemme* $GF(p^k)$.

Todistukset. Sivuutetaan. Katso [5].

Muodostetaan esimerkiksi Galois'n kunta $GF(2^4)$. Valitaan ensimmäiseksi supistumaton neljännen asteen polynomi modulo 2. Tälläinen on esimerkiksi $X^4 + X + 1$. Valitaan jokin polynomin primitiivijuuri esim. X. Lasketaan $X^k \pmod{X^4 + X + 1}$, $(k = 1, 2, \ldots, 2^4 - 1)$. Huomattavaa on se, että laskutoimitukset tapahtuvat kunnassa $\mathbf{Z}_2$. Tulokseksi saamme ne $2^4 - 1 = 15$ polynomia, joktka on esitetty taulukossa 2.1. Muutamme kunkin polynomin vielä luvuksi n siten, että sijoitamme $X = 2$ ja laskemme esim. $n = X^3 + X = 2^3 + 2 = 10$.

k	$X^k \pmod{X^4 + X + 1}$	n
1	X	2
2	X^2	4
3	X^3	8
4	$X + 1$	3
5	$X^2 + X$	6
6	$X^3 + X^2$	12
7	$X^3 + X + 1$	11
8	$X^2 + 1$	5
9	$X^3 + X$	10
10	$X^2 + X + 1$	7
11	$X^3 + X^2 + X$	14
12	$X^3 + X^2 + X + 1$	15
13	$X^3 + X^2 + 1$	13
14	$X^3 + 1$	9
15	1	1

Taulukko 2.1. *Esimerkkipolynomiin liittyvät pseudosatunnaisluvut.*

Näin on saatu 15 pseudosatunnaislukua. Tällaisen generaattorin voidaan ajatella olevan myös kryptografisesti vahva, sillä myös tässä on kyse eräänlaisesta diskreetin logaritmin ongelmasta. Laskennalista problematiikka tuottaa kuitenkin supistumattoman polynomin löytäminen, sekä primitiivipolynomin löytyminen. Tässä esityksessä ei kuitenkaan paneuduta tähän ongelmaan. Äärellisiä kuntia voidaan myös käyttää suunniteltaessa Tausworthe-generaattoreita.

3 Eräiden generaattorien ominaisuuksista

3.1 Lineaarisen kongruenssin generaattori

Lineaarisen kongruenssin generaaattori on siis muotoa (1). Nyt kertoimet $a_0, a_1, \ldots, a_j, b$ ovat jaottomia luvun P kanssa. Tästä seuraa

Apulause 3.1.1. *Yhtälön (1) mukainen generaattori on syklinen ja sykli alkaa generaattorin alusta eli alkuarvoista $x_0, x_1, \ldots, x_j$.*

Todistus. Koska generaattori voi saada korkeintaan P^{j+1} erilaista arvoa, niin generaattori saa jossain vaiheessa varmasti samoja arvoja, ja on siis syklinen. Olkoon nyt syklin alkukohdan arvot $x_n, x_{n-1}, \ldots, x_{n-j}$. Väite on todistettu, kun osoitetaan vielä, että $n = j$. Tehdään vasta-oletus, että $n > j$. Oletetaan, että syklin pituus on k eli $x_{i+k} \equiv x_i \pmod P$, kun $i \geq n - j$. Lisäksi $x_{n-j-1+k} \not\equiv x_{n-j-1} \pmod P$. Nyt siis

$$x_n \equiv a_0 x_{n-1} + a_1 x_{n-2} + \cdots + a_j x_{n-j-1} + b \pmod P.$$

Toisaalta myös

$$\begin{aligned} x_{n+k} &\equiv a_0 x_{n-1+k} + a_1 x_{n-2+k} + \cdots + a_{n-j+k} + a_j x_{n-j-1+k} + b \\ &\equiv a_0 x_{n-1} + a_1 x_{n-2} + \cdots + a_{n-j} x_{n-j} + a_j x_{n-j-1+k} + b \pmod P. \end{aligned}$$

Koska $x_{n+k} \equiv x_n \pmod P$, niin $a_j x_{n-j-1+k} \equiv a_j x_{n-j-1} \pmod P$. Edelleen koska luvuilla a_j ja P ei ole yhteisiä tekijöitä, niin $x_{n-j-1+k} \equiv x_{n-j-1} \pmod P$. tämä on ristiriita, joten vastaoletus oli väärä. On siis oltava $n = j$. □

Apulauseen 3.1.1 perusteella tiedämme siis, että generaattori toistuu tietyn jakson jälkeen alusta. Käytämme seuraavaa määritelmää

Määritelmä 3.1.1. *Generaattorin pituus on pienin positiiviluku k, jolle on voimassa $x_{i+k} \equiv x_i$, kun $i \geq 0$. Merkitään tässä modulia P vastaavan generaattorin pituutta $L(P)$.*

Huomautus. $L(P)$ riippuu paitsi modulista P, niin myös kertoimista $a_0, a_1, \ldots, a_j$, b sekä alkuarvoista $x_0, x_1, \ldots, x_j$.

Olkoot meillä nyt generaattorilla kaksi eri modulia P_1 ja P_2. Olkoot näiden pituudet vastaavasti $L(P_1)$ ja $L(P_2)$. Nyt saamme modulille $P_1 \cdot P_2$ seuraavan lauseen.

Lause 3.1.2. *Olkoon luvut P_1, P_2 keskenään jaottomia. Tällöin $L(P_1 \cdot P_2) = \mathrm{LCM}(L(P_1), L(P_2))$.*

Todistus. Oletuksen mukaan on $x_{i+k \cdot L(P_1)} \equiv x_i \pmod{P_1}$ ja $x_{i+k \cdot L(P_2)} \equiv x_i \pmod{P_2}$, kun $i, k \geq 0$. Merkitään $k = \mathrm{LCM}(L(P_1), L(P_2))$. Siis

$$x_{i+k} \equiv x_i \pmod{P_1}$$

ja

$$x_{i+k} \equiv x_i \pmod{P_2}.$$

Koska luvuilla P_1, P_2 ei ole yhteisiä tekijöitä, niin

$$x_{i+k} \equiv x_i \pmod{P_1 \cdot P_2}.$$

Siis $L(P_1 \cdot P_2) = k$. □

Jos siis tiedämme generaattorin pituuden moduleilla P_1 ja P_2, niin saamme tietää lauseen 2 perusteella generaattorin pituuden modulille $P_1 \cdot P_2$.

Huomautus. Lauseen 3.1.2 tulos yleistyy useammallekin modulille $P_1, P_2, \ldots, P_k$. Jos luvut $P_1, \ldots, P_k$ ovat keskenään jaottomia, niin

$$L(P_1 P_2 \cdots P_k) = \mathrm{LCM}(L(P_1), L(P_2), \ldots, L(P_k)).$$

Huomautus. Riittää tarkastella tilannetta, kun moduli on muotoa $P = p^k$, missä p on alkuluku.

Tarkastelemme seuraavaksi tilannetta, jossa modulina on alkuluvun p potenssi. Saamme seuraavan lauseen

Lause 3.1.3. *Olkoon generaattorin pituus $L(p^k)$ modulilla p^k. Tällöin generaattorin pituus $L(p^{k+1})$ modulilla p^{k+1} on joko $L(p^k)$ tai $p \cdot L(p^k)$.*

Todistus. ([4] ss.34–35) Osoitamme, että $L(p^{k+1}) = l \cdot L(p^k)$, missä $l \mid p$ eli $l = 1$ tai $l = p$, mistä seuraa väitös. Koska

$$x_{i+L(p^{k+1})} \equiv x_i \pmod{p^{k+1}},$$

niin myös

$$x_{i+L(p^{k+1})} \equiv x_i \pmod{p^k}.$$

Siis $L(p^{k+1}) \geq L(p^k)$. Tiedetään siis, että $L(p^{k+1}) = q \cdot L(p^k) + r$, missä $0 \leq r < L(p^k)$. Nyt siis $x_{i+q \cdot L(P^k)+r} \equiv x_{i+r} \equiv x_i \pmod{p^k}$. Näin ollen $r = 0$. Siis $L(p^{k+1})$ on muotoa $l \cdot L(p^k)$. Vielä osoitetaan, että $l \mid p$. Jos nyt

$$x_{i+L(p^k)} \equiv x_i \pmod{p^{k+1}} \ (i = 0, 1, \ldots, j),$$

niin $L(p^{k+1}) = L(p^k)$ eli tässä tapauksessa väite on tosi. Oletetaan nyt, että on olemassa sellainen $i \in \{0, 1, \ldots j\}$, että

$$x_{i+L(p^k)} \not\equiv x_i \pmod{p^{k+1}}.$$

Tällöin

$$x_{i+L(p^k)} \equiv x_i + c_i p^k \pmod{p^{k+1}},$$

missä $c_i \not\equiv 0 \pmod{p}$. Tästä seuraa, että

$$x_{i+v \cdot L(p^k)} \equiv x_i + v c_i p^k \pmod{p^{k+1}}.$$

Koska $c_i \not\equiv 0 \pmod{p}$, niin $v c_i \not\equiv 0 \pmod{p}$, $(v = 1, 2, \ldots, p-1)$. Siis

$$x_{i+v \cdot L(p^k)} \not\equiv x_i \pmod{p^{k+1}}, \ (v = 1, 2, \ldots, p-1).$$

Toisaalta $p c_i \equiv 0 \pmod{p}$, joten

$$x_{i+p \cdot L(p^k)} \equiv x_i + p c_i p^k \equiv x_i \pmod{p^{k+1}}.$$

Olemme todistaneet, että $l|p$. □

Olemme todistaneet, että modulin ollessa alkuluvun potenssi eksponentin kasvatus yhdellä joko pitää generaattorin pituuden ennallaan tai kasvattaa sen pituuden p-kertaiseksi.

Saamme vielä seuraavan lauseen

Lause 3.1.4. *Mikäli* $L(p^{k+1}) = p \cdot L(p^k)$, *niin tällöin* $L(p^{k+m}) = p^m \cdot L(p^k)$, *kun* $m \geq 0$.

Todistus. ([4], ss.34–35) Todistamme lauseen induktiolla luvun m suhteen. Perusaskel $m = 0$ tai $m = 1$ on triviaali. Todistamme vielä tapauksen $m = 2$. Kuten lauseessa 3.1.3, koska $c_i \not\equiv 0 \pmod{p}$, niin $vc_i \not\equiv 0 \pmod{p}$, $(v = 1, 2, \ldots, p-1)$. Samoin, koska $c_i \not\equiv 0 \pmod{p}$, niin $vpc_i \not\equiv 0 \pmod{p^2}$, $(v = 1, 2, \ldots, p-1)$. Toisaalta pätee $p^2 c_i \equiv 0 \pmod{p^2}$. Näin ollen

$$x_{i+vp\cdot L(p^k)} \not\equiv x_i \pmod{p^{k+2}},\ (v = 1, 2, \ldots, p-1)$$

ja

$$x_{i+p^2\cdot L(p^k)} \equiv x_i + p^2 c_i p^k \equiv x_i \pmod{p^{k+2}}.$$

Siis $L(p^{k+2}) = p \cdot L(p^{k+1}) = p^2 \cdot L(p^k)$. Olkoon nyt $m \geq 1$. Tehdään induktio-oletus, että $L(p^{k+n}) = p^n \cdot L(p^k)$, kun $n \leq m$. Siis
$L(p^{k+m-1+1}) = p \cdot L(p^{k+m-1})$. Saadaan siis

$$\begin{aligned} L(p^{k+m+1}) &= L(p^{k+m-1+2}) \\ &= p^2 \cdot L(p^{k+m-1}) \\ &= p^2 \cdot p^{m-1} \cdot L(p^k) \\ &= p^{m+1} \cdot L(p^k). \end{aligned}$$

Induktioperiaatteen mukaan väite on todistettu. □

Nyt meillä on keinot konstruoida pitkiä lineaarisen kongruenssin generaattoreita. Tavoitteena on löytää sellaiset kertoimet generaattoriin, että modulin kasvatus pidentää generaattorin pituutta. Tällöin voimme valita modulin kuinka suureksi tahansa ja kasvattaa generaattorin pituutta rajatta. Idea on etsiä generaattoreita, jotka ovat täysmittaisia eli modulin P pituisia. Tällöin toteutuu ainakin se kriteeri, että generaattori on tasan jakautunut.

3.2 Esimerkkejä

Tarkastelemme seuraavaksi esimerkkeinä erilaisia muotoa $x_{n+1} \equiv ax_n + b \pmod{2^k}$ olevia generaattoreita. Valitsemme ensimmäiseksi moduliksi $2^1 = 2$. Tällöin

$a, b, x_0 \equiv 0, 1 \pmod 2$. Tutkimme generaattorin pituutta lukujen a, b, x_0 eri arvoilla. Huomattavaa on kuitenkin, että tapaukset $a, b \equiv 0 \pmod 2$ eivät ole mielenkiintoisia, koska generaattorin oletuksissa päätettiin, että kertoimilla ja modulilla ei ole yhteisiä tekijöitä. Tulokset on esitetty taulukossa 3.2.1.

$a \bmod 2$	$b \bmod 2$	$x_0 \bmod 2$	$L(2)$
1	1	0	2
1	1	1	2

Taulukko 3.2.1. *Esimerkkigeneraattorin arvoja modulilla 2.*

Siis saamme täyspituisen generaattorin aina, kun $a, b \equiv 1 \pmod 2$ ja modulina on 2. Tulos on riippumaton alkuarvosta x_0.

Seuraavaksi pyrimme parantamaan tulosta ja tarkastelemme tilannetta, kun $a, b \equiv 1 \pmod 2$ ja modulina on 4. Tällöin siis $a, b \equiv 1, 3 \pmod 4$. Saamme taulukon 3.2.2.

$a \bmod 4$	$b \bmod 4$	$L(4)$
1	1	4
1	3	4
3	1	2
3	3	2

Taulukko 3.2.2. *Esimerkkigeneraattorin arvoja modulilla 4.*

Siis täysipituinen generaattori saadaan, kun $a \equiv 1 \pmod 4$ ja $b \equiv 1 \pmod 2$. Koitamme parantaa tulosta vielä ja tutkimme tilannetta, kun modulina on 8. Tällöin siis $a \equiv 1, 5 \pmod 8$, $b \equiv 1, 3, 5, 7 \pmod 8$. Saamme taulukon 3.2.3.

$a \bmod 8$	$b \bmod 8$	$L(8)$
1	1	8
1	3	8
1	5	8
1	7	8
5	1	8
5	3	8
5	5	8
5	7	8

Taulukko 3.2.3. *Esimerkkigeneraattorin arvoja modulilla 8.*

Siis $L(4) = 4$ aina, kun $a \equiv 1 \pmod 4$ ja $b \equiv 1 \pmod 2$. Samoilla ehdoilla myös $L(8) = 8$. Nyt siis lauseen 3.1.4 perusteella lineaarisen kongruenssin generaattorin $x_{n+1} \equiv ax_n + b \pmod{2^k}$ pituus $L(2^k) = 2^k$ aina, kun $a \equiv 1 \pmod 4$ ja $b \equiv 1 \pmod 2$.

Olemme löytäneet kokonaisen perheen täysipituisia lineaarisen kongruenssin generaattoreita. Tämä analyyttinen tarkastelu takaa meille sen, että pseudosatunnaislukumme ovat todellakin tasajakaumasta. Itse asiassa jokaisilla parametrin arvoilla generaattori on lukujen $0, 1, 2, \ldots, 2^k - 1$ jokin permutaatio. Näin ei kuitenkaan ole mahdollista saavuttaa näiden lukujen kaikkia mahdollisia permutaatioita, koska erilaisia generaattoreita on $2^k/4 \cdot 2^k/2 = 2^{2k-3}$ kappaletta. Kun taas erilaisia permutaatioita on $(2^k)!$ kappaletta.

Tieteenkään generaattorin muodostamat satunnaisluvut eivät ole riippumattomia. Kysymys siitä, että kuinka pahasti satunnaisluvut ovat riippuvuussuhteessa, palautuu parametrien arvojen valintaan ja siihen sovellukseen, johin generaattoria aiotaan käyttää.

On huomattavaa mitä bittejä satunnaisluvusta aiotaan käyttää. Esimerkiksi, jos otetaan kustakin satunnaisluvusta vähiten merkitsevä bitti, niin tällöin saamme uuden bittijonon, jonka pituus on vain 2. Jos taas otetaan 2 vähiten merkitsevää bittiä kustakin satunnaisluvusta, niin uuden bittijonon syklin pituus on 4 ja niin edelleen.

Nyrkkisääntönä lineaarisen kongruenssin generaattoreissa onkin käyttää vain ylimpiä eli eniten merkitseviä bittejä.

Tutkitaan seuraavaksi Fibonacci-generaatoria $x_n \equiv x_{n-1} + x_{n-2} \pmod{2^k}$. Muodostetaan taulukot kuten edellisen generaattorin tapauksessa. Nyt riittää tosin tutkia alkuarvojen x_0, x_1 valintaa. Nyt joudumme käyttämään oletusta, että alkuarvot ja moduli ovat keskenään jaottomat. Tutkitaan siis ensin tilanne $P = 2$. Tällöin $x_0, x_1 \equiv 1 \pmod 2$. Täten saamme pseudosatunnaislukujonon $1, 1, 0, 1, 1, 0, \ldots$. Siis $L(2) = 3$. Tutkitaan seuraavaksi tilanne, kun $P = 4$. Siis $x_0, x_1 \equiv 1, 3 \pmod 4$. Täten saadaan taulukko 3.2.4.

$x_0 \bmod 4$	$x_1 \bmod 4$	$L(4)$
1	1	6
1	3	6
3	1	6
3	3	6

Taulukko 3.2.4. *Fibonacci-generaattorin arvoja modulilla 4.*

Jälleen lauseen 3.1.4 perusteella, alkuarvojen x_0, x_1 ollessa parittomia, generaattorin pituus tuplaantuu aina, kun moduli tuplaantuu. Eli, kun $x_0, x_1 \equiv 1 \pmod 2$, niin generaattorin $x_n \equiv x_{n-1} + x_{n-2} \pmod{2^k}$ pituus $L(2^k)$ on $3 \cdot 2^k$.

Vastaavalla tavalla voidaan käsitellä myös kaikkia muita lineaarisen kongruenssin generaattoreita, mutta täytyy huomata, että sopivien ehtojen löytäminen ei ole aina helppoa. Lisäksi tällaisissa generaattoreissa täytyy olla tarkkana myös hilarakenteen kanssa, jotta satunnaisluvut eivät korreloisi liikaa.

Lineaarisen kongruenssin generaattorit eivät tosin ole kryptografisessa mielessä hyviä, sillä osittaisistakin biteistä voidaan selvittää esimerkiksi alkuarvot, joilla generaattori on alustettu. Kryptografisesti vahvojen generaattorien idea palautuukin juuri siihen seikkaan, että ei pystytä selvittämään polynomisessa ajassa generaattorin alkuarvoja tai edes alkuarvojen osaa.

4 Generaattoreiden tilastollisesta testaamisesta

Tavallisesti generaattoreiden tutkiminen matemaattisilta ominaisuuksiltaan on vaikeaa eivätkä ne anna kuvaa generaattorin käyttäytymisestä käytettävässä sovelluksessa. Täytyy siis turvautua tilastollisiin menetelmiin. Käytännössä tämä tarkoittaa sitä, että voidaan suorittaa sarja tilastollisia testejä ja tutkia, voidaanko olettaa haluttujen ominaisuuksien olevan voimassa valitulla riskitasolla. Tärkeimmät ominaisuudet ovat

- Generaattori on halutusta jakaumasta, yleensä tasa-jakaumasta.
- Generaattori on tilastollisesti riippumaton.

Ensimmäiseen kriteeriin tavallisesti luokitellaan aineistoa ja tehdään tilastolliset testit, joista tavallisimmat ovat χ^2-testit sekä Kolmogorov-Smirnov-testi.

Jälkimmäinen kriteeri on hankalampi todeta, vaikka kyse olisi "oikeasti" satunnaisita luvuista, mutta jotkut testipatterit tunnistavat tavallisimpien generaattorityyppien riippuvuudet, mikäli parametrejä ei ole valittu huolella. Tällaisia ovat muun muuassa autokorrelaatio-testit, Gap-testit sekä Run-testit.

Vaikka generaattori läpäisikin testit saamme vain jonkin asteisen varmuuden sen käyttökelpoisuudesta. Toisaalta vaikka generaattori ei läpäisikään testipatteria, niin ei se tarkoita, että generaattori olisi huono. Itse asiassa joillakin otoksilla, ehkäpä harvinaisilla, generaattori pitäisi hylätä kaikkien testien perusteella. Tämän pitäisi tapahtua, mikäli generaattori on erinomainen.

Tarkastellaan seuraavaksi joitakin tilastollisia testejä.

4.1 χ^2-testit

Tarkastellaan aluksi χ^2-jakaumaa. Mikäli meillä on n kappaletta normaalijakautuneita riippumattomia satunnaismuuttujia $x_1, x_2, \ldots, x_n$, niin silloin

$$\sum_{i=1}^{n} x_i^2$$

noudattaa χ^2-jakaumaa parametrinaan n. Oletetaan, että satunnaismuuttuja $x \sim \chi^2(n)$. Tällöin sen odotus-arvo $\mathrm{E}(x) = n$ ja varianssi $\mathrm{Var}(x) = 2n$.

Lause 4.1.1. *Olkoon joukko luokiteltu k osajoukon kokoelmaksi ja olkoon näiden mitatut frekvenssit $f_1, f_2, \ldots, f_k$ sekä odotusarvoiset frekvenssit $e_1, e_2, \ldots, e_k$. Tällöin*

$$\chi_s^2 = \sum_{i=1}^{k} \frac{(f_i - e_i)^2}{e_i} \sim \chi^2(k-1)$$

Todistus. Katso [2].

Lause 4.1.1 antaa oivan työkalun tutkia, onko tutkittava generaattori halutusta jakaumasta. Oletetaan, että generaattorin pitäisi tuottaa satunnaislukuja jakaumasta $\mathrm{Tas}(0,1)$. Tuotetaan suurehko määrä pseudosatunnaislukuja generaattorilla. Luokitellaan ne k osajoukkoon, esim. väleihin $[0, 1/k), [1/k, 2/k), \ldots, [(k-1)/k, 1)$. Lasketaan kunkin joukon frekvenssit $f_1, f_2, \ldots, f_k$. Lasketaan myös odotettavat frekvenssit $e_1, e_2, \ldots, e_k$. Esimerkki tapauksessa $e_1 = e_2 = \cdots = e_k = 1/k$. Lasketaan lauseen 4.1.1 testisuure. Nyt meillä on nollahypoteesina

H0: Otos on tasajakaumasta eli $\chi_s^2 = 0$.

Vaihtoehtoisen hypoteesin ollessa

H1: Otos ei ole tasajakaumasta eli $\chi_s^2 > 0$.

Seuraavaksi valitaan riskitaso, jolla H0 hylätään. Tavallisimmin täksi valitaan 0.01. Seuraavaksi katsotaan $\chi^2(k-1)$-jakaumasta se arvo, joka on suurempi 99% luvuista. Esimerkiksi, jos $k = 10$, niin $\chi^2(9)$- jakaumassa tämä kohta olisi $21,67$. Siis, jos laskemamme testisuure $\chi_s^2 > 21,67$, niin nollahypoteesi hylätään, ja tällöin voimme olettaa, että generaattori ei tuota satunnaislukuja tasajakaumasta. Virheen mahdollisuus on tällöin 1%. Monet valmisohjelmistot, kuten SAS, SPSS, Statistica, jne. laskevat suoraan tällaiset testit.

Sitä suuremman varmuuden jakaumasta saadaan, mitä useammalla tavalla joukko jaetaan erilaisiin osajoukkoihin ja suoritetaan χ^2-testejä. Huo-

mattavaa on kuitenkin se, että myös hyvän generaattorin olettaisi joissain testeissä tulevan hylätyksi.

Myös Kolmogorov-Smirnov-testiä voi käyttää tähän tarkoitukseen. Myös tämä testi löytyy useimmista valmisohjelmistoista. Sivuutamme kuitenkin sen tarkastelun.

4.2 Autokorrelaatio testit

Autokorrelaatioita laskemalla voidaan tutkia onko olemassa korrelaatiota generoidun satunnaisluvun ja edeltävien satunnaislukujen välillä. Tällöin rakennetaan regressiomalli

$$\mathbf{X}_k = (\mathbf{1} : \mathbf{X}_1 : \mathbf{X}_2 : \cdots : \mathbf{X}_{k-1})\mathbf{b} + \mathbf{e},$$

missä vektori $\mathbf{X}_i = (x_i \; x_{i+1} \cdots x_{i+n})^T$ $(i = 1, \ldots, k)$, luvut $x_1, \ldots, x_{n+k}$ ovat generoidut satunnaisluvut ja $\mathbf{1} = (1\,1 \cdots 1)_n^T$. Kerroinvektori $\mathbf{b} = (b_0 \; b_1 \cdots b_{k-1})^T$ on estimoitava kerroinvektori, joka saadaan pienimmän neliösumman menetelmällä. Kun satunnaisluvut ovat riippumattomia, pätee virhevektorille $\mathbf{e} \sim N(0, \mathbf{I}_{n\times n})$ likimain. Merkitään vielä lyhyesti $\mathbf{A} = (\mathbf{1} : \mathbf{X}_1 : \mathbf{X}_2 : \cdots : \mathbf{X}_{k-1})$. Siis ratkaistaan estimaatti $\hat{\mathbf{b}}$ pienimmän neliösumman menetelmällä, jolloin saadaan

$$\hat{\mathbf{b}} = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{X}_k.$$

Estimoituja kertoimia analysoidaan tilastollisin menetelmin. Oletetaan, että satunnaislukumme olivat $N(0,1)$ normaalijakaumasta. Tasajakaumasta päästään helposti normaalijakaumaan muunnoksella $y = \Phi^{-1}(x)$, missä $x \sim Tas(0,1)$ ja Φ on normaalijakauman kertymäfunktio. Nyt hypoteesit ovat

H0: $\hat{\mathbf{b}} = \mathbf{0}$,
H1: $\hat{\mathbf{b}} \neq \mathbf{0}$.

Käytettävä tilastollinen testi on niin sanottu Studentin t-testi. Myös nämä löytyvät tilasto-ohjelmistoista, kuten SPSS. Eräs tapa tutkia autokorrelaatiota on muodostaa korrelaatiokertoimia eri välein otetuille satunnaisluvuille ja tutkia ovatko, tilastollisesta näkökulmasta, kertoimet lähellä nollaa.

Myös muita aikasarja-analyysien menetelmiä voi käyttää tutkimaan onko riippuvuutta olemassa.

4.3 Gap-testit

Gap-testejä on monenlaisia, mutta yhteistä niille on, että tutkitaan eripituisia satunnaisluvuissa esiintyviä välejä ja verrataan näitä teoreettisiin todennäköisyyksiin. Tarkastellaan muutamia esimerkkejä. Tutkitaan satunnaislukugeneraattorilla saatua bittijonoa siten, että lasketaan kahden saman bitin välisiä etäisyyksiä. Nämä välit voivat saada arvoja $0, 1, 2, \ldots$. Nyt kahden saman bitin välisen etäisyyden r todennäköisyys saadaan laskettua seuraavasti

$$p_r = \left(1 - \frac{1}{2}\right)^r \cdot \frac{1}{2} = \left(\frac{1}{2}\right)^{r+1}.$$

Nyt lasketaan bittijonosta kunkin etäisyyden frekvenssit ja verrataan teoreettisiin todennäköisyyksiin p_r. Vertailuun voidaan taas käyttää lausetta 4.1.1.

Eräs tapa on tutkia välien pituuksia siten, että tutkitaan sellaisten lukujen välistä etäisyyttä, joiden vieressä molemmin puolin on pienempi luku. Tässä tutkitaan siis satunnaislukuja tasajakaumasta $Tas(0, 1)$.

Myös muunlaisia Gap-testejä on olemassa, tässä rajoituksena on vain mielikuvitus. Kysymys kuuluukin, onko sovelluksen kannalta välttämätöntä tutkia kokonaisia pattereita Gap-testejä.

4.4 Run-testit

Run-testeissä tutkitaan erilaisia muutoksia satunnaislukujonossa ja näiden muutosten pituuksia ja verrataan taas teoreettisiin todennäköisyyksiin. Tarkastellaan muutamia esimerkkejä.

Tutkitaan jonosta, miten satunnaisluvut suhtautuvat toisiinsa. Eli tutkitaan, onko $x_i > x_{i+1}$ ja taas onko $x_{i+1} > x_{i+2}$ ja niin edelleen. Kiinnostuksen kohteena on tilanne $x_i > x_{i+1} > \cdots > x_{i+k} < x_{i+k+1}$. Nyt tutkitaan lukujen $k = 1, 2, \ldots$ esiintymistiheyttä ja verrataan teoreettisiin todennäköisyyksiin. Vertailu voidaan taas suorittaa käyttämällä χ^2-testiä. Oletuksena meillä on siis, että luvut $x_i \sim Tas(0, 1)$. Teoreettiset esiintymistodennnäköisyydet voidaan muodostaa seuraavasti

$$\mathrm{P}(k = 1) = \mathrm{P}(x_1 > x_2) = \int_0^1 \int_0^{x1} 1 \, \mathrm{d}x_1 \mathrm{d}x_2 = \frac{1}{2} \quad \text{ja}$$

$$\mathrm{P}(k = r) = \mathrm{P}(x_1 > x_2 > \cdots > x_r) = \int_0^1 \int_0^{x_1} \cdots \int_0^{x_{r-1}} 1 \, \mathrm{d}x_r \mathrm{d}x_{r-1} \cdots \mathrm{d}x_1 = \frac{1}{r!}.$$

Nyt siis tarkastellaan satunnaislukujoukon tilannetta ja lasketaan otoksesta frekvenssit ja verrataan niitä edellä oleviin teoreettisiin todennäköisyyksiin.

Eräs Run-testi perustuu siihen, että koodataan satunnaislukujono uudestaan bittijonoksi seuraavalla funktiolla $f : [0, 1) \to \{0, 1\}$

$$f(x) = 1, \text{ kun } x < a, \quad f(x) = 0, \text{ kun } x \geq a,$$

missä luku a voidaan valita vapaasti väliltä $[0, 1)$. Tavanomaisesti tosin luvuksi a valitaan otoksen keskiarvo tai mediaani. Nyt satunnaisluvuista $x_1, x_2, \ldots$ on saatu bittijono $f(x_1), f(x_2), \ldots$ ja tähän bittijonoon voidaan soveltaa edellisessä kohdassa esiteltyä Gap-testiä.

Kuten Gap-testeissä, niin myös Run-testeissä vain mielikuvitus on rajana erilaisille testeille. Täytyy kuitenkin arvioida, onko testi sovelluksen kannalta tarpeellinen ja pystytäänkö muodostamaan teoreettiset todennäköisyydet tai approksimoimaan ne riittävän tarkasti.

4.5 Pokeri-testit

Pokeritesteissä sananmukaisesti simuloidaan korttipeliä nimeltä pokeri tai tämän muunnoksia. Eli satunnaislukujen oletetaan muodostavan erilaisia käsiä peliin. Sitten lasketaan erilaisten käsien esiintymisfrekvenssi ja verrataan teoreettisiin todennäköisyyksiin saada kyseisiä käsiä. Vertailu toteteutaan, kuten ennenkin, χ^2-testein.

4.6 Lopuksi tilastollisista testeistä

Tässä oli pintaraapaisu tavanomaisista tilastollista testeistä, joita käytetään satunnaislukugeneraattorien hyvyyden testaamiseen. Löytyy vielä paljon muitakin testejä ja tässä esitetyille testeille löytyy lukemattomia muunnoksia.

Selvää on, että kaikkia testejä ei ole mahdollista suorittaa eikä se olisi myöskään järkevää. Nyrkkisääntö on se, että testit valitaan sen mukaan, mitä ominaisuuksia generaattorilla täytyy olla. Tämä on täysin sovelluskohtaista. Eli generaattori, joka on erinomainen toisessa sovelluksessa, voi olla äärimmäisen huono toisessa.

Ei ole olemassa universaalia testiä, joka korvaisi kaikki muut testit, mutta aina voidaan kehittää testejä, jotka kattavat mahdollisimman suuren joukon muita testejä.

Huomautus. Tässä sanottiin, että ei ole universaaleja testejä. Tämä pitää paikkansa, mutta on olemassa monia testejä, jotka on nimetty universaaleiksi esimerkiksi *Ueli Maurerin universaalitesti*.

Siinä pyritään approksimoimaan bittijonon entropiaa. Mikäli entropia on tilastollisesti lähellä muistittoman satunnaislähteen entropiaa

$$H = -\frac{1}{2}\log\frac{1}{2},$$

niin tällöin kyse on riippumattomasta tasajakautuneesta bittijonosta. Tällaisia testejä pyritään kehittämään erityisesti kryptologian tarpeisiin.

5 Satunnaislukugeneraattorit kryptografiassa

5.1 Yksisuuntaiset funktiot

Tässä luvussa käsitellään niin sanottuja yksisuuntaisia funktioita. Aluksi on huomautettava, että tätä tutkielmaa kirjoitettaessa ei ole vielä selvitetty, onko tällaisia edes olemassa. Aloitetaan yksisuuntaisen funktion määritelmästä.

Määritelmä 5.1.1. *Funktiota* $f : \{0,1\}^* \to \{0,1\}^*$ *sanotaan yksisuuntaiseksi, mikäli*
(1) On olemassa sellainen $c \geq 1$, *että* $\|x\|^{1/c} < \|f(x)\| < \|x\|^c$,
(2) $f(x)$ *on polynomisessa ajassa laskettavissa,*
(3) Jokaista satunnaistettua polynomisessa ajassa laskettavaa algoritmia $A : \{0,1\}^* \to \{0,1\}^*$ *kohti*

$$\mathrm{P}(f(A(f(y))) = f(y)) = \mathrm{NEGL}(n),$$

missä y *on satunnaisesti valittu joukon* $\{0,1\}^n$ *tasajakaumasta.*

Määritelmä 5.1.2. *Sanomme, että yksisuuntainen funktio on yksisuuntainen permutaatio, mikäli funktio on injektiivinen sekä*

$$\|f(x)\| = \|x\|, \ \forall x \in \{0,1\}^*.$$

Tarkastellaan aluksi näitä määritelmiä. Määritelmän 5.1.1 kohta (1) takaa sen, että merkkijonon kuvan pituus on rajoitettu merkkijonon pituuden mukaisesti. Erityisesti erikoistapaus $c = 1$ on mielenkiintoinen, sillä silloin merkkijonot kuvautuvat samanpituisiksi merkkijonoiksi.

Kohta (2) on tärkeä vaatimus laskennan teorian kannalta. Mikäli ei olisi kyse polynomisessa ajassa laskettavasta funktiosta, niin funktiolla ei käytännön hyötyä juuri olisikaan.

Kohta (3) kertoo, että kyseinen todennäköisyys on *mitätön* eli pienempi kuin luvun n minkä tahansa polynomin käänteisluku. Kohdassa (3) määritellään todennäköisyyden sille, että mikäli valitaan sattumanvaraisesti merkkijono y ja lasketaan $f(y)$ sekä yritetään tämän jälkeen algoritmilla A laskea merkkijonon $f(y)$ alkukuva, niin tässä onnistutaan. Huomautan, että alkukuvia voi olla siis useita, ellei funktio f ole injektio. Lisäksi funktio ei välttämättä

ole surjektio; tästä syystä ei voida kirjoittaa suoraan $\mathrm{P}(f(A(y)) = y) = \mathrm{NEGL(n)}$.

Yksisuuntainen permutaatio on bijektiivinen, joten tässä tapauksessa kohdan (3) kaava voidaan kirjoittaa

$$\mathrm{P}(f(A(y)) = y) = \mathrm{NEGL}(n).$$

Yksisuuntaisten funktioiden tai permutaatioiden olemassaolo on vielä avoin ongelma. Jos $\mathcal{P} = \mathcal{NP}$, niin tällöin ei voi olla olemassa yksisuuntaisia funktioita. Tämä seuraa tietysti suoraan siitä, että olisi olemassa sellainen polynomisessa ajassa laskettava funktio A, että $f(A(f(y)) = f(y)$ kaikilla syötteillä y. Tällöin kohdan (3) todennäköisyys olisi siis 1 ja $1 \neq \mathrm{NEGL}(n)$.

Toisaalta avoin ongelma on se, että mikäli $\mathcal{P} \subset \mathcal{NP}$, niin onko tällöin myös olemassa yksisuuntaisia funktioita vai ei.

Eroavaisuus näissä kahdessa ongelmassa on se, että ongelma $\mathcal{P} = \mathcal{NP}$? käsittelee pahimman tapauksen tilannetta, kun taas yksisuuntaisen funktion olemassaolo käsittelee keskimääräisen tapauksen tilannetta.

Kryptografisessa mielessä yksisuuntaisen funktion idea on siinä, että vaikka merkkijonosta on helppo laskea merkkijonon kuva, niin merkkijonon kuvasta on (liian) vaativaa laskea merkkijono.

Tähän saakka yksisuuntaisista funktioista on onnistuttu konstruoimaan yksisuuntaisia permutaatioita ja toisinpäin. Tällöin tosin kohdan (3) todennäköisyys yleensä tapaa suurentua.

Mielenkiintoinen ongelma olisi myös konstruoida $\mathcal{NP}$-täydellisestä ongelmasta yksisuuntainen funktio. Tämä jo yllä todettiinkin avoimeksi ongelmaksi.

Tarkastellaan nyt konjekturoituja yksisuuntaisia funktioita.

Tekijöihin jaon ongelma. Olkoot p, q alkulukuja, joiden pituus binäärijärjestelmässä on n. Olkoon $f(n, x, y) = (n, xy)$. Toistaiseksi ei tunnetta polynomisessa ajassa laskettavaa funktiota $A(n, xy) = (n, x, y)$, joten funktion f ajatellaan olevan yksisuuntainen.

Diskreetin logaritmin ongelma. Olkoot p alkuluku, g primitiivijuuri modulo p, ja $0 < i < p$. Olkoon $f(p, g, i) = (p, g, g^i \bmod p)$. Toistaiseksi ei tunneta polynomisessa ajassa laskettavaa funktiota $A(p, g, g^i \bmod p) = (p, g, i)$. Lukujen p, g ei tarvitse olla satunnaisesti valittu, mutta luku p valitaan yleensä sellaiseksi, että luvulla $p - 1$ on suuria tekijöitä.

Diskreetin neliöjuuren ongelma. Olkoot $m \in \mathbf{Z}_+$, ja $x < m$. Olkoon $f(m, x) = (m, x^2 \bmod m)$. Taaskaan ei tunneta polynomisessa ajassa laskettavaa funktiota $A(m, x^2 \bmod m) = (m, x)$.

RSA-funktion ongelma. Olkoot p, q alkulukuja. Olkoon luku $e \in \mathbf{Z}_{p \cdot q}$ sellainen, että sillä ei ole yhteisiä tekijöitä luvun $(p-1)(q-1)$ kanssa. Olkoon $y \in \mathbf{Z}_{p \cdot q}$ ja edelleen $f(p, q, e, y) = (p \cdot q, e, y^e \bmod p \cdot q)$. Jälleen ei tunneta polynomisessa ajassa laskettavaa funktiota $A(p \cdot q, e, y^e \bmod p \cdot q) = y$.

Kaikissa näissä ongelmissa löytyy erikoistapauksia, jotka ratkeavat polynomisessa ajassa. Käytännön kryptografiassa nämä funktion f syötteet valitaan suuriksi, yleensä suuremmaksi kuin 2^{128}.

5.2 Turvalliset satunnaisluvut

Määrittelemme nyt kryptografisesti turvalliset pseudosatunnaislukugeneraattorit.

Määritelmä 5.2.1. *Olkoon $f : \{0,1\}^* \to \{0,1\}^*$ polynomisessa ajassa laskettava funktio, jolle on voimassa*

$$\|x\| \leq \|f(x)\| < \|x\|^c,$$

*jollain vakiolla c. Sanomme tällaista funktiota **generaattoriksi**.*

Määritelmä 5.2.2. *Olkoon $A : \{0,1\}^* \to \{0,1\}$ satunnaistettu polynomisessa ajassa laskettavissa oleva funktio. Olkoon $n \in \mathbf{Z}_+$. Olkoon $x \in \{0,1\}^n$ valittu satunnaisesti. Olkoon $y \in \{0,1\}^N$ valittu satunnaisesti, missä $N = \|f(x)\|$. Nyt sanomme, että funktio A on **onnistunut**, jos joko $A(f(x)) = 0$ tai $A(y) = 1$.*

Tällainen funktio A siis tunnistaa, onko annettu syöte sattumanvaraisesti valittu vai jonkin alkion x kuva $f(x)$. Funktio A saa arvon 0, kun kyseessä on funktion f jokin kuva, ja funktio f saa arvon 1, kun syötteenä on sattumanvaraisesti valittu merkkijono.

Määritelmässä 5.2.2 funktio A saa syötteekseen siis joko merkkijonon $f(x)$ tai merkkijonon y. Kummankin todennäköisyys olla syöte on $1/2$.

Määritelmä 5.2.3. *Sanomme, että generaattori f on* ***turvallinen satunnaislukugeneraattori****, mikäli jokaiselle satunnaistetulle polynomisessa ajassa laskettavalle funktiolle A on voimassa*

$$\mathrm{P}(A \text{ onnistuu}) \leq 1/2 + \mathrm{NEGL(n)}.$$

Satunnaislukugeneraattori on siis turvallinen, mikäli ei ole polynomisessa ajassa laskettavaa funktiota, joka tehokkaasti tunnistaisi todelliset satunnaisluvut generaattorilla tuotetuista.

Turvallisten satunnaislukujen olemassaolo on avoin kysymys. Mutta mikäli yksisuuntaisia permutaatioita on, niin silloin on myös turvallisia satunnaislukugeneraattoreita.

Määritelmä 5.2.4. *Olkoon $n \geq 1$. Olkoon x valittu satunnaisesti yhtä suurin todennäköisyyksin joukosta $\{0,1\}^n$. Olkoon $f(x) = F_1F_2 \ldots F_N$. Mikäli jokaiselle satunnaistetulle polynomisessa-ajassa laskettavalle algoritmille $A : \{0,1\}^i \to \{0,1\},\ (i \leq N)$ on voimassa*

$$\max_i \mathrm{P}(A(F_1F_2 \ldots F_i) = F_{i+1}) = 1/2 + \mathrm{NEGL}(n),$$

niin sanomme generaattorin f olevan ***ennustamaton****.*

Toisin sanoen, generaattori on ennustamaton, mikäli edellisistä biteistä ei kyetä laskemaan polynomisessa ajassa seuraavaa bittiä tai onnistumisen todennäköisyys ei ole kuin mitättömästi suurempi kuin 1/2.

Lause 5.2.1. *Generaattori f on turvallinen satunnaislukugeneraattori, jos ja vain jos se on ennustamaton.*

Todistus. ([6] ss.50-52.) Osoitamme vain, että turvallinen satunnaislukugeneraattori on ennustamaton ja sivuutamme todistuksen toiseen suuntaan. Oletetaan nyt, että generaattori f on turvallinen satunnaislukugeneraattori. Tehdään vasta-oletus, että generaattori ei olisikaan ennustamaton. Tällöin on olemassa sellaiset satunnaistettu polynomisessa ajassa laskettava algoritmi A ja vakio $k > 0$, että on olemassa äärettömän monta sellaista lukua n, että jokaista lukua n kohti on olemassa sellainen $i \leq N$, että

$$\mathrm{P}(A(F_1F_2 \ldots F_i) = F_{i+1}) > \frac{1}{2} + \frac{1}{n^k}.$$

Tässä siis $x \in \{0,1\}^n$ on satunnaisesti valittu ja $f(x) = F_1F_2 \dots F_N$. Olkoon

$$A_2 : \{0,1\}^* \to \{0,1\},$$

$$A_2(y) = 0, \text{ kun } A(y_1y_2 \dots y_i) = y_{i+1} \text{ ja } A_2(y) = 1 \text{ muutoin.}$$

Olkoon nyt $R_1R_2 \dots R_N$ satunnaisesti valittu joukosta $\{0,1\}^N$. Nyt todennäköisyys, että algoritmi A_2 tunnistaa satunnaisen merkkijonon merkkijonosta $f(x)$ on

$$\begin{aligned} \frac{1}{2} \cdot \mathrm{P}(A(R_1R_2 \dots R_i) \neq R_{i+1}) + \frac{1}{2} \cdot \mathrm{P}(A(F_1F_2 \dots F_i) = F_{i+1})) &\geq \\ \frac{1}{2} \cdot \frac{1}{2} + \frac{1}{2}\left(\frac{1}{2} + \frac{1}{n^k}\right) &= \\ \frac{1}{2} + \frac{1}{2n^k}. \end{aligned}$$

Koska $\frac{1}{2} + \frac{1}{2n^k} \neq \frac{1}{2} + \mathrm{NEGL}(n)$, niin algoritmi f ei ole turvallinen satunnaislukugeneraattori, mikä on ristiriita. Näin ollen vasta-oletus on väärä ja generaattori on ennustamaton. □

Katsotaan seuraavaksi, mitä yhteistä on turvallisilla satunnaisluku generaattoreilla ja yksisuuntaisilla funktioilla.

Lause 5.2.2. *Olkoon f turvallinen satunnaislukugeneraattori. Olkoon lisäksi $\|f(x)\| \geq 2 \cdot \|x\|$. Tällöin f on yksisuuntainen funktio.*

Todistus. (Vrt. lause 5.2.1.) Oletetaan, että generaattori f ei ole yksisuuntainen funktio. Tällöin on olemassa vakio $k > 0$ ja satunnaistettu polynomisessa-ajassa laskettava algoritmi A sekä äärettömän monta sellaista lukua n, että satunnaisesti valitulle merkkijonolle $y \in \{0,1\}^n$ on voimassa

$$\mathrm{P}(f(A(f(y))) = f(y)) > \frac{1}{n^k}.$$

Olkoon nyt

$$A_2 : \{0,1\}^N \to \{0,1\},$$

$$A_2(y) = 0, \text{ kun } f(A(y)) = y, \text{ ja } A_2(y) = 1 \text{ muutoin.}$$

Olkoon $r \in \{0,1\}^N$ satunnaisesti valittu. Nyt algoritmin A_2 onnistumisen todennäköisyys, kun syötteeksi annetaan todennäköisyydellä $1/2$ joko r tai $f(x)$, on

$$\frac{1}{2} \cdot \mathrm{P}(g(A(r)) \neq r) + \frac{1}{2} \cdot \mathrm{P}(g(A(g(x))) = g(x))).$$

Todetaan aluksi, että $\frac{1}{n^k} > \frac{1}{2^n}$, kun luku n on riittävän suuri. Koska

$$\mathrm{P}(f(A(r) = r) \leq \frac{2^n}{2^N} \leq \frac{2^n}{2^{2n}} = \frac{1}{2^n},$$

niin

$$\mathrm{P}(f(A(r)) \neq r) \geq \left(1 - \frac{1}{2^n}\right).$$

Näin ollen

$$\begin{aligned} \frac{1}{2} \cdot \mathrm{P}(g(A(r)) \neq r) + \frac{1}{2} \cdot \mathrm{P}(g(A(g(x))) = g(x))) &\geq \\ \frac{1}{2} \cdot \left(1 - \frac{1}{2^n}\right) + \frac{1}{2} \cdot \frac{1}{n^k} &= \\ \frac{1}{2} + \frac{1}{2n^k} - \frac{1}{2^{n+1}} &> \\ \frac{1}{2} + \frac{1}{4n^k}. \end{aligned}$$

Koska $\frac{1}{2} + \frac{1}{4n^k} \neq \mathrm{NEGL}(n)$, niin generaattori ei ole turvallinen satunnaislukugeneraattori, mikä on ristiriita. Siis generaattori f on yksisuuntainen funktio. □

Voidaan myös osoittaa, että yksisuuntaisten funtioiden olemassa-olo takaa turvallisten satunnaislukujen olemassa-olon.

Lause 5.2.3. *Olkoon f yksisuuntainen permutaatio. Tällöin on olemassa turvallinen satunnaislukugeneraattori.*

Todistus. Sivuutetaan. Katso [6].

Yksisuuntaista funtioista voidaan muodostaa yksisuuntaisia permutaatioita, mutta yleensä turvallisuudessa hävitään. Tällä ei tosin olisi merkitystä, mikäli voitaisiin todistaa, että oletetut yksisuuntaiset funktiot olisivat todella yksisuuntaisia.

5.3 Esimerkki

Tarkastellaan seuraavaksi konkreettista esimerkkiä kryptografisesti vahvasta satunnaislukugeneraattorista.

Lause 5.3.2. *Olkoon* $f : \{0,1\}^n \to \{0,1\}^n$ *yksisuuntainen permutaatio. Olkoot* $\|x\| = 2n$, $N = n^k$ $(k > 0)$ *ja* $y(t) = f^t(x_1, x_2, \ldots, x_n)$ $(t = 1, \ldots, N)$. *Olkoot edelleen*

$$g_t = y_1(t) \odot x_{n+1} \oplus y_2(t) \odot x_{n+2} \oplus \cdots \oplus y_n(t) \odot x_{2n}$$

ja $g(x) = g_N g_{N-1} \ldots g_1$. *Tällöin funktio* g *on turvallinen satunnaislukugeneraattori.*

Todistus. Sivuutetaan. Katso [6].

Tarkastellaan lauseen 5.3.2 menetelmää turvallisen satunnaislukugeneraattorin muodostamiseksi. Diskreetin logaritmin

$$f(z) = g^z \bmod p$$

on konjekturoitu olevan yksisuuntainen permutaatio.

Valitaan nyt moduliksi alkuluku $p = 65537$. Tälle löytyy primitiivijuuri $g = 75$. Valitaan alkuarvo $z_1 \in \{1, 2, \ldots 65536\}$. Valitaan toinen luku satunnaisesti $w \in \{0, 1, 2, \ldots, 65535\}$. Tällöin $\|z_1 - 1\| = \|w\| = 16$. Valitaan satunnaisbittien lukumäärä, joka on nyt siis muotoa 16^k. Valitaan esimerkin vuoksi $16^2 = 256$. Suoritetaan seuraava pseudokielinen algoritmi:

1. Iteroi 256 kertaa sijoitusta $z_{i+1} = 75^{z_i} \bmod 65537$.

2. Laske $y_i = z_i - 1$ luvun i arvoilla $1, 2, \ldots, 256$.

3. Laske $x_i = (y_i \text{ AND } w)$ luvun i arvoilla $1, 2, \ldots, 256$.

4. Laske luvun x_i pariteetti ja sijoita muuttujaan g_i. Tee tämä luvun i arvoilla $1, 2, \ldots, 256$.

5. Tulosta bitit g_{256-i} luvun i arvoilla $1, 2, \ldots, 256$.

6. Lopeta.

Valitsin satunnaisesti luvut $z = 55$, $w = 46774$ ja tuloksena tuli taulukon 5.3.1 psedosatunnaisbittijono.

1-64	65-129	130-193	193-255
01010110	11110001	10110110	00001101
01001001	10000100	01110100	01011111
10000010	10110101	01011011	10110000
00101110	10111110	11110000	01001010
00001001	10011100	00110001	10000000
01100100	10000100	10010110	00101010
10111100	01101001	01010001	01011101
01111110	11010011	01011001	1010011

Taulukko 5.3.1. *Esimerkkiin liittyvä pseudosatunnaisbittijono.*

Kovin satunnaisellta näyttää! Idea on kuitenkin siinä, että 16-bittisestä luvusta saatiin laskettua tämä 256-bittinen luku. Lisäksi tästä pidemmästä 256-bittisestä luvusta ei voida polynomisessa ajassa laskea alkuperäistä 16-bittistä lukua. Tämä pitää paikkansa mikäli kyseessä siis todella on yksisuuntainen funktio.

Tietysti kaikki 16-bittiset luvut voidaan nopeasti tarkistaa tietokoneella, koska niitä ei ole kuin 65536 kappaletta. Asia on toisin, jos lähdettäisiin esimerkiksi 256-bittisestä luvusta. Tällöin pitäisi tarkistaa 2^{256} lukua. Tähän ei nykyteknologialla enään pystyttäisikään.

6 Satunnaislukugeneraattorit ja Monte Carlo menetelmä

6.1 Monte Carlo menetelmä

Simuloinnissa pyritään tarkastelemaan matemaattisen mallin käyttäytymistä eri parametrien arvoilla ja tämän jälkeen valitsemaan edullisimman tuloksen omaavat parametrit. Kuitenkin useissa malleissa oletetaan olevan mukana satunnaistekijöitä tai kohinaa, joita ei tunneta. Tällöin malliin pitäisi lisätä satunnaistekijät. Jos nyt kuitenkin käytettäisiin oikeita satunnaislukuja, niin simulointia ei voisi samanlaisena totetuttaa uusille parametrin arvoille, koska tulokset saattavat pahimmassa tapauksessa johtua kokonaan satunnaistekijöistä. Tämän vuoksi pseudosatunnaislukujen käyttö on oleellinen osa simulointia. Pseudosatunnaisluvuilla on se ylivoimainen piirre, että ne voidaan toistaa.

Puhumme suorasta simuloinnista silloin, kun malliin lisätään satunnaisluvuista riippuvia muuttujia.

Mateemaattisen ongelman ratkaisu, jolle ei löydy tai ei tunneta analyyttistä ratkaisua, voidaan usein löytää tai approksimoida niin sanotulla Monte Carlo menetelmällä.

Monte Carlo menetelmä perustuu siihen, että tulosta approksimoidaan satunnaisesti valituilla yritteillä.

Itse asiassa voidaan olettaa, että Monte Carlo menetelmässä on kyse moninkertaisen integraalin laskemisesta eli integraalin

$$V = \int_0^1 \cdots \int_0^1 R(x_1, x_2, \ldots, x_n)\ \mathrm{d}x_1 \mathrm{d}x_2 \cdots \mathrm{d}x_n$$

laskemisesta, missä funktio R on ongelmaan valittu sopiva funktio.

Monte Carlo-menetelmä perustuu siihen, että valitaan satunnaisluvut tasajakaumasta $y_1, y_2, \ldots y_n \sim \mathrm{Tas}(0,1)$. Lasketaan satunnaismuuttuja

$$S = R(y_1, y_2, \ldots, y_n).$$

Nyt tämä satunnaismuuttuja S on yllä olevan integraalin harhaton estimaattori. Siis $\mathrm{E}(S) = V$. Odotusarvon harhaton estimaattori on tunntetusti aritmeettinen keski-arvo. Eli lasketaan useita satunnaismuuttujia $S_1, S_2, \ldots$ ja lasketaan näistä keski-arvo $\overline{S}$. Funktio R täytyy kuitenkin pyrkiä valitsemaan siten, että $\mathrm{Var}(\overline{S}) = 1/n \cdot \mathrm{Var}(R)$ on mahdollisimman pieni, koska tällöin voidaan olla varmempi tuloksen oikeellisuudesta ja tarkkuudesta.

Tarkastellaan esimerkkinä legendaarinen luvun π likiarvon laskeminen Monte Carlo menetelmällä.

Luku π on yksikköympyrän pinta-ala. Yksikköympyrä toteuttaa tunnetusti yhtälön

$$x^2 + y^2 = 1$$

eli $y = \sqrt{1 - x^2}$. Olkoot nyt satunnaisluvut $e_1, e_2, \ldots, e_{2n} \sim \text{Tas}(0, 1)$.

Tutkitaan ensin niin sanottua Hit-or-Miss tekniikkaa. Koska yksikköympyrä on ensimmäisessä neljänneksessä joukon $[0, 1] \times [0, 1]$ osajoukko, niin voimme tarkastella seuraavaa funktiota

$$R(x, y) = \begin{cases} 1, \text{ kun } \sqrt{1 - x^2} \leq y \\ 0, \text{ muulloin.} \end{cases}$$

tai yhtäpitävästi

$$R(x, y) = \begin{cases} 1, \text{ kun } x^2 + y^2 \leq 1 \\ 0, \text{ muulloin.} \end{cases}$$

Funktio R ilmaisee, kuuluuko piste (x, y) yksikköympyrään vai ei. Näin ollen $\text{E}(R) = \frac{\pi}{4}$ ja tämä on myös samalla todennäköisyys pisteen kuulumiselle yksikköympyrään. Täten kyseessä on binomijakauma yhdellä alkiolla (Bernoullijakautuma) ja näin ollen $\text{Var}(R) = \frac{\pi}{4}\left(1 - \frac{\pi}{4}\right)$. Nyt lasketaan

$$\overline{S} = \frac{1}{n} \sum_{i=1}^{n} R(e_{2i-1}, e_{2i}).$$

Nyt $\text{E}(\overline{S}) = \frac{\pi}{4}$ ja $\text{Var}(\overline{S}) = \frac{1}{n} \cdot \frac{\pi}{4}\left(1 - \frac{\pi}{4}\right)$ eli $\text{Var}(\overline{S}) \approx \frac{0.16855}{n}$.

Tutkitaan seuraavaksi toisenlaista tekniikkaa: raakaa Monte Carlo menetelmää. Tullemme huomaamaan, yllättävää kyllä, että tämä menetelmä on hieman edellistä parempi. Käytämme saman määrän satunnaislukuja ja valitsemme funktioksi

$$R(x) = \sqrt{1 - x^2}.$$

Nyt

$$\text{E}(R) = \int_0^1 \sqrt{1 - x^2} \text{d}x = \frac{\pi}{4}, \ \text{Var}(R) = \int_0^1 \left(\sqrt{1 - x^2} - \frac{\pi}{4}\right)^2 \text{d}x = \frac{2}{3} - \left(\frac{\pi}{4}\right)^2.$$

Jälleen lasketaan

$$\overline{S} = \frac{1}{2n} \sum_{i=1}^{2n} R(e_i).$$

Nyt $\mathrm{E}(\overline{S}) = \frac{\pi}{4}$ ja $\mathrm{Var}(\overline{S}) = \frac{1}{2n} \cdot \left(\frac{2}{3} - \left(\frac{\pi}{4}\right)^2\right)$. Eli $\mathrm{Var}(\overline{S}) \approx \frac{0.0249}{n}$.

Tästä huomaamme, että funktion R valinnalla on merkitystä. Jälkimmäisen funktion varianssi on huomattavasti pienempi. Myös tehokkaampia menetelmiä on kehitetty, joista monet ovat varsin sovelluskohtaisia. Emme tässä esityksessä kuitenkaan käy läpi erilaisia menetelmiä vähentää varianssia vaan keskitymme Monte Carlo menetelmään satunnaislukugeneraattorien näkökulmasta.

6.2 Satunnaislukugeneraattorin valinta

Satunnaislukugeneraattorin valinta tiettyyn Monte Carlo ongelmaan ei ole aivan yksikäsitteistä. Mutta eräitä sääntöjä, jotka olemme jo aiemminkin todenneet, voidaan soveltaa:

- Generaattorin täytyy olla riittävän pitkä
- Generaattorin täytyy olla tasajakutunut avaruudessa $[0, 1)^k$, missä luku k on mahdollisimman suuri.

Varsinkin jälkimmäinen vaatimus on tärkeä.

Otetaan esimerkiksi generaattori

$$x_{n+1} \equiv 5x_n + 1 \pmod{65536}.$$

Tämä generaattori on tasajakautunut joukossa $\{0, 1, 2, \ldots 65535\}$. Näin ollen $y_n = x_n/65536$ on tasan jakautunut välillä $[0, 1)$. Itse asiassa generaattorin pituus on juuri 65536. Tämä tiedetään luvun 3 perusteella.

Oletetaan, että olemme päättäneet toteuttaa integraalin

$$\int_0^1 \int_0^1 R(x, y)\mathrm{d}x\mathrm{d}y,$$

missä

$$R(x, y) = \begin{cases} 1, \text{ kun } x, y \leq 0.5 \text{ tai } x, y > 0.5 \\ -1, \text{ muulloin} \end{cases}$$

laskemisen Monte Carlo tekniikalla.

Tälläinen tapaus voisi tulla kyseeseen esimerkiksi simuloitaessa rahapeliä, missä pelaajat asettavat kolikon panokseksi, ja toinen voittaa, mikäli kolikot ovat sama puoli päällä, muutoin voittaa toinen pelaaja. Tällöin itse asiassa olisi kyse reilusta nollasummapelistä, jonka odotusarvo on 0.

Siis esimerkiksi luvut ≤ 0.5 edustaisivat klaavoja ja luvut > 0.5 edustaisivat kruunuja. Lasketaan pseudosatunnaisluvut $e_1, e_2, \ldots, e_{2n}$ generaattorilla. Seuraavaksi laskemme

$$\overline{S} = \frac{1}{n} \sum_{i=1}^{n} R(e_{2i-1}, e_{2i}).$$

Oletetaan, että olemme valinneet $n = 500$. Tällöin $\overline{S}$ saa liian suuria positiivisia arvoja. Tämä ei johdu huonosta Monte Carlo menetelmästä vaan satunnaislukugeneraattorin valinnasta, sillä kyseessä oleva generaattori ei ole tasan jakautunut avaruudessa $[0, 1)^2$. Taulukosta 6.2.1 käykin ilmi, miten 500 kappaleen satunnaislukuparit jakautuvat.

$x \backslash y$	$y \leq 0.5$	$y > 0.5$
$x \leq 0.5$	153	87
$x > 0.5$	90	170

Taulukko 6.2.1. *Esimerkkiin liittyvien satunnaislukujen jakautuminen.*

Täten laskettuna $\overline{S} = 1/500 \cdot (153 + 170 - 87 - 90) = 0.292$, kun taas $\mathrm{E}(R) = 0$ ja $\mathrm{Var}(R) = 1/(4 \cdot 500) = 0.0005$. Eli tilastollisesti tarkasteltuna voidaan suurella varmuudella sanoa, että $\overline{S}$ on reilusti suurempi kuin 0.

Tällä satunnaislukugeneraattorilla voisi tulla ongelmia samasta syystä esimerkiksi yksikköneliöön mahtuvan sopivan suuntaisen ellipsin pinta-alan laskemisessa.

Aiemmin mainitut kriteerit ovat paras yleinen kriteeri valita satunnaislukugeneraattori. Mutta mikäli mallin käyttäytyminen tunnetaan jossain tilanteessa, niin tätä voidaan käyttää myös testaamaan, soveltuuko kyseinen generaattori juuri tämän mallin testaamiseen vai pitäisikö mahdollisesti etsiä parempia.

Yleisesti ottaen on kovin vaikea sanoa generaattorista, onko se tasajakautunut, erityisesti jos vaaditaan tasajakautumista useamissa ulottuvuuksissa.

Paras teoreettinen testi tälle onkin niin sanottu spektraalitesti (ks. [1]), mutta tämä soveltuu vain lineaarisen kongruenssin generaattoreihin.

Lopuksi

Hyvän psedosatunnaislukugeneraattorin löytäminen on vaikeaa, erityisesti jos toivotaan generaattorin olevan kryptografisesti vahva satunnaislukugeneraattori. Tämä on ongelmallista jo siitäkin syystä, ettei tällaisten olemassaolosta ole minkäänlaista varmuutta. Ei tunneta esimerkiksi sopivaa ehdokasta, jonka tiedettäisiin olevan $\mathcal{NP}$-täydellinen. Tosin $\mathcal{NP}$-täydellisistä ongelmista voidaan johtaa generaattoreita, mutta näissä on huomattu olevan suuria puutteita.

Lisäksi satunnaislukugeneraattori pitää aina valita silmälläpitäen sovellusta. Sovellushan voisi olla itsessään satunnaislukugeneraattori. Tällöin esimerkiksi siemenluvun generoiminen toisella generaattorilla voisi olla ongelmallista, jos siemenluvulla pitää olla tietyt ominaisuudet. Esimerkiksi se, että tätä lukua ei voida laskennallisesti (polynomisessa ajassa) toinen osapuoli laskea.

Kehitys tietysti jatkuu generaattoreiden ja niille kehitettyjen testien osalta. Saadaan yhä pitempiä generaattoreita, joiden tiedetään olevan tasajakautuneita yksikköhyperkuutioissa, joiden dimensio lähentelee jo tuhatta.

Esimerkiksi Makoto Matsumoton kehittämä generaattori Mersenne Twister on sellainen, että sen pituus on huima $2^{19937} - 1$ ja tasajakautunut ainakin $[0, 1)^{623}$, ja tietysti myös sitä alemissa dimensioissa.

Tasajakautumisen teoreettiseen tarkasteluun sopiva lupaava testi on niin sanottu diskrepanssin laskeminen satunnaislukujonosta. Diskrepanssi lähenee lukua 0, mikäli jonon voi olettaa olevan tasajakautunut. Tosin diskrepanssi on tarkoitettu äärettömien jonojen tarkasteluun. Mutta sopivin oletuksin sitä voisi soveltaa myös äärellisiin jonoihin.

Empiiristä testeistä mielenkiintoisin on entropian approksimoiminen satunnaislukujonosta. Tälläisiä testejä voisivat olla erilaisten tiedontiivistysalgoritmien soveltaminen satunnaislukujonoon ja sen tutkiminen, kuinka tiivistyssuhde eroaa symmetrisen binäärilähteen tiivistyksen jakaumasta. Ongelmana on tietysti se, että tällaisia jakaumia ei yleensä tunneta, paitsi, kokeellisesti saadut.

Kirjallisuus

[1] Bratley, Paul: A Guide to Simulation, Springer 1983.

[2] Feller, W.: An Introduction to Probability Theory and Its Applications, Vol. 2, Wiley 1971.

[3] Hammersley, J.M.: Monte Carlo Methods, Fletcher & Son 1964.

[4] Jansson, Birger: Random Number Generators, Victor Pettersons 1966.

[5] Lidl, R.: Introduction to Finite Fields and Their Applications, Cambridge 1988.

[6] Luby, Michael: Pseudorandomness and Cryptographic Applications, Princeton Academic Press 1996.