

## Generatori di Numeri Casuali

Michela Meo  
Maurizio M. Munafò

Michela.Meo@polito.it - Maurizio.Munafò@polito.it

## Copyright

Quest'opera è protetta dalla licenza *Creative Commons NoDerivs-NonCommercial*. Per vedere una copia di questa licenza, consultare:  
<http://creativecommons.org/licenses/nd-nc/1.0/>  
oppure inviare una lettera a:  
*Creative Commons, 559 Nathan Abbott Way, Stanford, California 94305, USA.*

This work is licensed under the *Creative Commons NoDerivs-NonCommercial* License. To view a copy of this license, visit:  
<http://creativecommons.org/licenses/nd-nc/1.0/>  
or send a letter to  
*Creative Commons, 559 Nathan Abbott Way, Stanford, California 94305, USA.*

## Obiettivo

- I simulatori necessitano di generatori di istanze di variabili casuali con una data distribuzione
- Si generano in questo modo
  - Le tracce sintetiche dei processi di ingresso al simulatore (es. processo degli arrivi, tempi di servizio, ...)
  - Comportamenti aggregati di porzioni del sistema non simulati in dettaglio (es. probabilità di errore sul canale, ritardi di attraversamento di porzioni della rete, ...)

## Obiettivo

- I generatori devono
  - Generare sequenze di numeri le cui proprietà statistiche approssimino accuratamente quelle ideali
  - Rendere possibile la *ripetibilità* di un esperimento
  - Generare un numero molto grande di istanze diverse e scorrelate tra loro
  - Essere efficienti
  - Essere portabili su computer e linguaggi di programmazione diversi

## Obiettivo

- Procediamo come segue:
  1. Generiamo una sequenza X di numeri casuali interi
  2. Partendo da X, generiamo una sequenza Z di istanze di variabile casuale uniforme in (0,1)
  3. Usando Z, generiamo istanze della v.c. di interesse

## Numeri pseudo-casuali

- Consideriamo generatori di sequenze *pseudo-casuali*
  - L'algoritmo di generazione è deterministico
  - Noti i parametri dell'algoritmo, si può rigenerare la stessa sequenza, ed è quindi possibile ripetere un esperimento
  - Le sequenze generate *imitano* quelle casuali, nel senso che hanno le stesse proprietà statistiche di una sequenza di istanze di una v.c. uniforme

## Generatori Congruenti Lineari

- Sono i più diffusi
- Furono inizialmente proposti da Lehmer negli anni '50
- Sono basati sull'operatore di *modulo*
- Generano una sequenza di numeri interi  $X_i$  nell'intervallo  $(0, m-1)$  tramite la relazione iterativa

$$X_{i+1} = (aX_i + c) \bmod m$$

## Generatori Congruenti Lineari

$$X_{i+1} = (aX_i + c) \bmod m$$

- $X_0$ : è detto *seme*
- $a$ : *moltiplicatore*
- $c$ : *incremento*
  - $c = 0$  nei generatori *congruenti moltiplicativi*
  - $c \neq 0$  nei generatori *misti congruenti*
- $m$ : *modulo*

## Sequenze di Uniformi

- Dato un generatore di sequenze  $X$  di numeri interi in  $[0, m-1]$ , una sequenza  $Z$  di istanze di uniformi in  $[0, 1)$  oppure in  $[0, 1]$  si ottiene da
$$Z_i = X_i / m \text{ oppure } Z_i = X_i / (m-1)$$
- A seconda del tipo di normalizzazione gli estremi della sequenza, 0 e 1, si possono includere o meno

## Generatori Congruenti Lineari

- **Le sequenze non sono casuali**  
Infatti l'algoritmo è deterministico; però la sequenza generata *sembra* casuale, sembra una sequenza di istanze indipendenti e uniformi
- **La sequenza non può assumere tutti i valori in  $(0, 1)$**   
Infatti i valori diversi da  $i/m$  non sono possibili, ma con valori di  $m$  grossi la sequenza è molto densa

## Generatori Congruenti Lineari

- La scelta dei parametri  $(X_0, a, c, m)$  del generatore influenza fortemente le proprietà della sequenza generata
- Generatori diversi hanno caratteristiche stocastiche diverse che le fanno imitare più o meno bene le caratteristiche delle variabili uniformi
  - Un generatore con modulo  $m$  dovrebbe generare interi uniformemente distribuiti nell'intervallo  $[0, m-1]$

## Generatori Congruenti Lineari: Esempio 1

- $a=7, c=0, m=11, X_0=1$
- Sequenza generata:
$$1, 7, 5, 2, 3, 10, 4, 6, 9, 8, 1, 7, 5$$

- Sequenza contiene tutti i numeri possibili in  $[1, 10]$ : è di lunghezza massima, pari a  $m-1$
- Quando viene nuovamente generato il seme, la sequenza si ripete identica

### Generatori Congruenti Lineari: Esempio 2

- $a=5, c=0, m=11, X_0=1$

- Sequenza generata:

1, 5, 3, 4, 9, 1, 5, 3



- La sequenza (lunga 5) ha una lunghezza minore di  $m-1=10$
- Alcuni numeri non compaiono mai

### Generatori Congruenti Lineari: Esempio 3

- $a=5, c=0, m=11, X_0=6$

- Sequenza generata:

6, 8, 7, 2, 10, 6, 8,



- Partendo da un seme non contenuto nella precedente sequenza, si ottengono tutti numeri non appartenenti alla sequenza precedente

### Caratteristiche delle Sequenze Generate

- Dato  $m$ , negli esempi precedenti ci si poteva aspettare un valor medio della sequenza pari a

$$m(m+1)/2=5.5$$

- Esempio 1: media = 5.5 (periodo massimo)
- Esempio 2: media = 4.4
- Esempio 3: media = 6.6

### Caratteristiche delle Sequenze Generate

- La lunghezza della sequenza è detta *periodo*
- Un generatore con modulo  $m$  ha un *periodo massimo* pari a
  - $m$  se  $c \neq 0$
  - $m-1$  se  $c=0$  (numero 0 non è possibile)
- Non tutti i generatori hanno periodo massimo

### Criteri per la Scelta dei Parametri

Caso  $m = 2^b, c \neq 0$

- il periodo massimo è  $m$  se
  - $c$  e  $m$  sono primi tra loro (l'unico fattore in comune è 1)
  - $a = 1 + 4k$ , con  $k$  intero

Caso  $m = 2^b, c = 0$

- il periodo massimo è  $m/4 = 2^{b-2}$  se
  - $X_0$  è dispari
  - $a = 3 + 8k$  oppure  $a = 5 + 8k$ , con  $k$  intero

### Criteri per la Scelta dei Parametri

Caso  $m$  primo e  $c = 0$

- il periodo massimo è  $m-1$  se
  - Il più piccolo intero  $k$  per cui  $a^k-1$  è divisibile per  $m$  è il numero  $k=m-1$

## Caratteristiche delle Sequenze Generate

- Negli esempi precedenti  $m$  è primo e  $c=0$ , il periodo massimo vale
  - Esempio 1:  $m-1=10$ , perché con  $a=7$  il più piccolo  $k$  tale per cui  $a^k - 1$  è divisibile per  $m=11$  è  $k=10$ , che è pari a  $m-1$
  - Esempi 2 e 3: periodo non è massimo perché con  $a=5$  il più piccolo  $k$  tale per cui  $a^k - 1$  è divisibile per  $m=10$  è  $k=5$  che è minore di  $m-1$

## Caratteristiche delle Sequenze Generate

- I criteri per la scelta dei parametri sono dettati, anche, da esigenze di efficienza e velocità di calcolo
  - L'operazione di modulo può essere più efficiente se  $m$  è una potenza di 2 (se  $m=2^b$  è dato dalle  $b$  cifre meno significative)

## Un Buon Generatore

- Linguaggio C
  - $a = 7^5 = 16,807$  --  $c = 0$   
 $m = 2^{31} - 1 = 2,147,483,647$  (primo)
- JAVA
  - $a=25,214,903,917$  --  $c=11$  --  $m=2^{48}$
- Visual Basic
  - $a=1,140,671,485$  --  $c=12,820,163$  --  $m=2^{24}$

## Problemi di calcolo

- I generatori congruenti lineari possono avere i seguenti problemi
  - Il linguaggio di programmazione impiegato deve svolgere le operazioni in aritmetica intera, senza arrotondamenti
  - Il prodotto  $aX_i$  può causare overflow

## Problemi di calcolo

- Per evitare l'overflow si può sfruttare una proprietà dell'operazione di modulo
- Si vuole eseguire
$$ax \bmod m$$
- Si definisce
  - $q = m \operatorname{div} a$  (l'operatore  $\operatorname{div}$  fornisce il quoziente troncato)
  - $r = m \bmod a$

## Problemi di calcolo

- Si può dimostrare che
$$ax \bmod m = g(x) + mh(x)$$
$$g(x) = a(x \bmod q) - r(x \operatorname{div} q)$$
con  $h(x)=0$  se  $g(x)$  è positivo, 1 altrimenti

### Altri Generatori Congruenti

- Alcune tecniche permettono di allungare la periodicità della sequenza
- Generatori congruenti più generali sono del tipo

$$X_i = g(X_{i-1}, X_{i-2}, \dots) \text{ mod } m$$

con  $g()$  una funzione generica

### Altri Generatori Congruenti

- Esempio 1: generatore *quadratico*

$$X_i = (a_2 X_{i-1}^2 + a_1 X_{i-1} + c) \text{ mod } m$$

con  $a_1 = a_2 = 1, c = 0, m = 2^b$  ha buone proprietà

- Esempio 2: generatore di *Fibonacci*

$$X_i = (X_{i-1} + X_{i-2}) \text{ mod } m$$

ha un periodo maggiore di  $m$  ma pessime caratteristiche stocastiche

### Generatori Congruenti Lineari Combinati

- Usano più generatori congruenti lineari combinandoli tra loro in diversi modi
- Uno di questi è basato sulla combinazione di  $k$  generatori congruenti lineari
- Siano  $X_{i,1}, X_{i,2}, \dots, X_{i,k}$  le  $i$ -esime istanze di  $k$  diversi generatori congruenti, con modulo primo e periodo massimo

### Generatori Congruenti Lineari Combinati

- Si esegue l'operazione seguente

$$X_i = \left( \sum_{j=1}^k (-1)^{j-1} X_{i,j} \right) \text{ mod } (m_i - 1)$$

$$Z_i = \begin{cases} \frac{X_i}{m_i}, & \text{con } X_i > 0 \\ \frac{m_i - 1}{m_i}, & \text{con } X_i = 0 \end{cases}$$

### Generatori Congruenti Lineari Combinati

- Il massimo possibile periodo è pari a

$$p = \frac{(m_1 - 1)(m_2 - 1) \dots (m_k - 1)}{2^{k-1}}$$

- Si ha periodo dell'ordine di  $10^{18}$  con
  - $k=2$
  - $a_1=40014, c_1=0, m_1=2147483562$
  - $a_2=40692, c_2=0, m_2=2147483399$

### Generatori Congruenti Lineari Combinati

- Un'altra tecnica è basata su *shuffling*
- Si usano due generatori:  $G_1$  per generare un insieme di numeri,  $G_2$  per scegliere all'interno dell'insieme
  - Genera il vettore  $V=(V_1, V_2, \dots, V_k)$  con  $G_1$
  - Genera un intero  $I$  in  $[1, k]$  con  $G_2$
  - Scegli  $V_I$  come istanza generata
  - Sostituisci  $V_I$  con il prossimo numero generato da  $G_1$

## Generatori Congruenti Lineari Combinati

- Questa tecnica migliora molto le proprietà di generatori cattivi, perché riduce le correlazioni
- Permette di estendere facilmente il periodo anche usando generatori con periodi brevi (per es., in macchine con aritmetica a pochi bit)
- Inizialmente era suggerito  $k=128$
- Anche con  $k=2$  si ottengono ottime prestazioni
- E' difficile ottenere punti della sequenza che siano lontani tra loro

## Generatori di Tausworthe

- Derivano dalle tecniche di crittografia, e lavorano direttamente sui bit (servono chiavi casuali lunghe, anche 512 bit)
- Si ricava la sequenza di numeri binari
$$b_i = (c_1 b_{i-1} + c_2 b_{i-2} + \dots + c_q b_{i-q}) \bmod 2$$
- con  $c_j$  costanti binarie
- Il massimo periodo è  $2^q - 1$
- Si raccolgono le cifre binarie a sequenze di lunghezza  $l$  per avere dei numeri interi

## Generatori di Tausworthe

- Un generatore spesso usato è del tipo

$$b_i = (b_{i-r} + b_{i-q}) \bmod 2$$

che è equivalente all'operazione di xor tra i bit (efficiente)

$$b_i = \begin{cases} 0 & \text{se } b_{i-r} = b_{i-q} \\ 1 & \text{se } b_{i-r} \neq b_{i-q} \end{cases}$$

## Scelta del Seme

- Quando basta un'unica sequenza di numeri casuali
  - La scelta del seme è indifferente se il simulatore ha periodo massimo
  - Alcuni generatori non accettano il seme 0 (per es. i congruenti moltiplicativi e i Tausworthe)
  - Per i moltiplicativi con  $m=2^k$  il seme deve essere dispari (altrimenti, il periodo non è massimo)

## Scelta del Seme

- A volte servono diverse sequenze
  - Si vogliono eseguire alcuni run di simulazione indipendenti
- Sequenze diverse tratte dalla stessa sequenza casuale dovrebbero non sovrapporsi, in modo da essere scorrelate

## Scelta del Seme

- La scelta di semi diversi per generare sequenze che non si sovrappongono può essere fatta generando, *prima della simulazione*, un gran numero di istanze
- Scegliendo  $M$  sufficientemente grande, si possono poi usare
$$X_0, X_M, X_{2M}, \dots$$
come semi di sequenze diverse

## Scelta del Seme

- Nel caso di generatori congruenti lineari, l'istanza  $M$ -esima può anche essere calcolata dalla formula

$$X_M = \left( a^M X_0 + \frac{c(a^M - 1)}{a - 1} \right) \bmod m$$

- Quando si fanno simulazioni successive, conviene usare, come semi di nuove sequenze, le ultime istanze generate nelle precedenti simulazioni

## Smentire alcuni luoghi comuni...

- Operazioni complesse portano a risultati difficilmente predicibili ma non per questo casuali (indipendenti e uniformi)
- I numeri casuali sono spesso facilmente predicibili ma apparentemente casuali (non sono adatti per crittografia)
- L'implementazione è importante, troncamenti e arrotondamenti vanno evitati