Generadores de números aleatorios

Patricia Kisbye

FaMAF

23 de marzo, 2010

¿Para qué se utilizan?

- Simulación.
- Muestreo.
- Análisis numérico.
- Testeo de programas.
- Juegos de azar.
- Toma de decisiones.

Secuencias de números aleatorios

En simulación las secuencias con distribución uniforme en $[0, 1]$ se utilizan:

- en forma directa,
- para generar distribuciones discretas y continuas,
- para generar valores de variables aleatorias dependientes.

Un poco de historia

Antes de las computadoras, existieron diferentes métodos para generar secuencias de números aleatorios:

- Procedimientos físicos (monedas, dados, bolilleros, . . .).
- (1927) Tipett: tabla de 40000 dígitos aleatorios (no resultaron con distribución uniforme).
- (1939) Kendall y Babbington: dispositivo mecánico. Tabla de 100.000 números aleatorios.
- (1955) Rand Corporation: ruido electrónico. Tabla de 1 millón de números aleatorios.

Inconvenientes de métodos físicos

1. No puede repetirse una misma secuencia.
2. No hay velocidad computacional.
3. Incorporar una tabla a la computadora implica gran costo de almacenamiento en relación a la cantidad de números.

Con la aparición de las computadoras, surgen métodos de generación de secuencias de números aleatorios.

Propiedades deseables de un generador

Aleatorio: Distribución uniforme

Un generador de números aleatorios razonable debe cumplir:

1. repetibilidad
   Al repetir los parámetros del generador, se repite la secuencia.
2. portabilidad,
   La secuencia no debe depender del lenguaje computacional ni de la computadora utilizada.
3. velocidad computacional.
   Puede ser conveniente utilizar lenguajes de bajo o medio nivel.

Principios generales de un buen generador

- La secuencia generada debe ser intuitivamente aleatoria.
- Esa aleatoriedad debe ser establecida teóricamente o, al menos, debe pasar ciertos tests de aleatoriedad.
- Debe conocerse algo sobre las propiedades teóricas del generador.

Un ejemplo

Secuencia de von Neumann (1946):

1. X_0: número de 5 dígitos. (03001)
2. X_i^2: escrito con diez dígitos. (0009006001)
3. X_{i+1}: 5 dígitos centrales. (09006)
4. Volver a 2

$$3001, 9006, 81108, 78507, 63349, 13095, 71479, 92474, 51440$$

$$21000, 41000, 81000, 61000, 21000 \ldots$$

con 4 dígitos: $3792, 3792, 3792, 3792, \ldots$

Desventajas del Gen. de von Neuman

1. No se conocen propiedades teóricas del generador: (semilla conveniente, número de dígitos en cada término).
2. Forsythe: (con 4 dígitos) De 16 semillas, 12 degeneraban en 6100, 2100, 4100, 6100,.... y 4 degeneraban en 0.
3. Metropolis: Secuencias de 20 bits degeneran en uno de 13 ciclos, con longitud máxima 142.
4. En algunos casos el primer tramo es satisfactoriamente aleatorio y luego degenera.

Generador congruencial lineal

$$\boxed{y_i = a y_{i-1} + c \mod M, \qquad 1 \leq i,}$$

$$x_i = \frac{y_i}{M}, \qquad \text{secuencia en el } [0, 1).$$

- y_0: semilla
- a: multiplicador
- c: incremento
- M: módulo
- generador *mixto*: $c \neq 0$
- generador *multiplicativo*: $c = 0$.

Ejemplo

¿Cuál es el siguiente número en la secuencia, entre 0 y 15?

$$0, 1, 6, 15, 12, 13, 2, 11, 8, 9, 14, ..$$

$$y_i = 5y_{i-1} + 1 \mod 16, \qquad y_0 = 0.$$

Secuencia intuitivamente no aleatoria:

$$1, 12, 1, 12, 1, 12, 1, 12 \ldots$$

Propiedades

- El menor número K tal que $y_{n+K} = y_n$ es el período de la secuencia.
- Todo generador congruencial genera secuencias de período finito.
- El período de una secuencia está acotado por M.
- Repetibilidad.
- ¿Portabilidad?

Elección de a, c y M

Las buenas propiedades dependen de una elección apropiada de a, c y M, y en algunos casos y_0.

- La elección de M se relaciona con: longitud de la secuencia y velocidad computacional.
- La elección de a y c, en función de M, se relacionan con la aleatoriedad.

Generadores mixtos

$$y_{i+1} = a\,y_i + c \mod M, \qquad c \neq 0$$

tiene período M si y sólo si

1. $m.c.d.(c, M) = 1$
2. $a \equiv 1 \mod p$, para cualquier factor primo p de M.
3. Si $4 \mid M$, entonces $a \equiv 1 \mod 4$.

Corolario: Si M es primo, el período máximo ocurre sólo si $a = 1$.

Generadores multiplicativos

- a es raíz primitiva de M si $a^{(M-1)/p} \not\equiv 1 \mod (M)$ para cualquier factor primo p de $M-1$.

Ejemplo: $M = 7$.

$a = 2$	$a = 3$
$2^1 \equiv 2 \mod 7$	$3^1 \equiv 3 \mod 7$
$2^2 \equiv 4 \mod 7$	$3^2 \equiv 2 \mod 7$
$2^3 \equiv 1 \mod 7$	$3^3 \equiv 6 \mod 7$
$2^4 \equiv 2 \mod 7$	$3^4 \equiv 4 \mod 7$
$2^5 \equiv 4 \mod 7$	$3^5 \equiv 5 \mod 7$
$2^6 \equiv 1 \mod 7$	$3^6 \equiv 1 \mod 7$
$2^7 \equiv 2 \mod 7$	$3^7 \equiv 3 \mod 7$

Generadores multiplicativos

Para un generador multiplicativo $y_{i+1} = a\,y_i \mod M$, la longitud K de la secuencia verifica

1. Si $K = M - 1$ entonces M es primo.
2. K divide a $M - 1$.
3. $K = M - 1$ si y sólo si a es raíz primitiva de M.

Problema: Encontrar raíces primitivas.

Propiedad útil: Si a es raíz primitiva y $(k, M - 1) = 1$, entonces a^k es raíz primitiva.

Un ejemplo con M primo

$$M = 2^{31} - 1 \qquad a = 16807$$

$$M - 1 = 2^{31} - 2 = 2 \cdot 3^2 \cdot 7 \cdot 11 \cdot 31 \cdot 151 \cdot 331,$$

- M: es un primo de Mersenne.
- 7: raíz primitiva, $(5, M - 1) = 1$, implica que $7^5 = 16807$ es raíz primitiva.
- secuencia de longitud máxima $M - 1 = 2\,147\,483\,646$.

Módulo potencia de 2

Si $M = 2^k$, $c = 0$, tomar módulo es computacionalmente sencillo.

$$y_j = a^j y_0 \mod (2^k)$$

- secuencia de longitud máxima = 2^{k-2}, para a raíz primitiva.
- facilita cálculos (desplazamiento de bits).

- fenómeno de no aleatoriedad en bits menos significativos.
- RANDU: $M = 2^{31}$, $a = 2^{16} + 3 = 65539$.

Desventaja de un generador congruencial

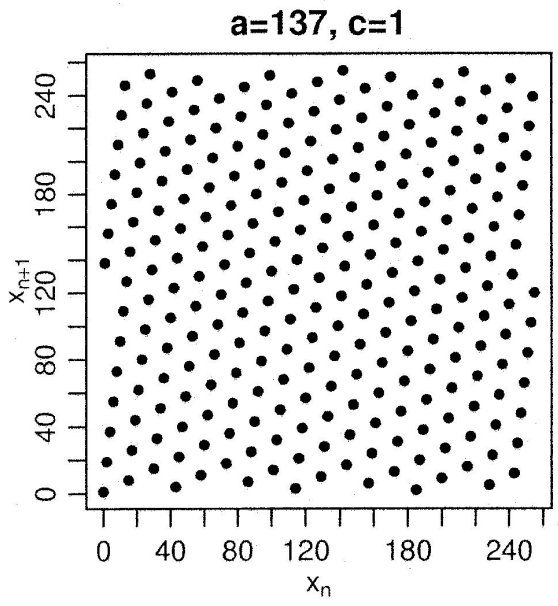
En una secuencia $y_1, y_2, \ldots$ dada por un generador congruencial cualquiera, los puntos

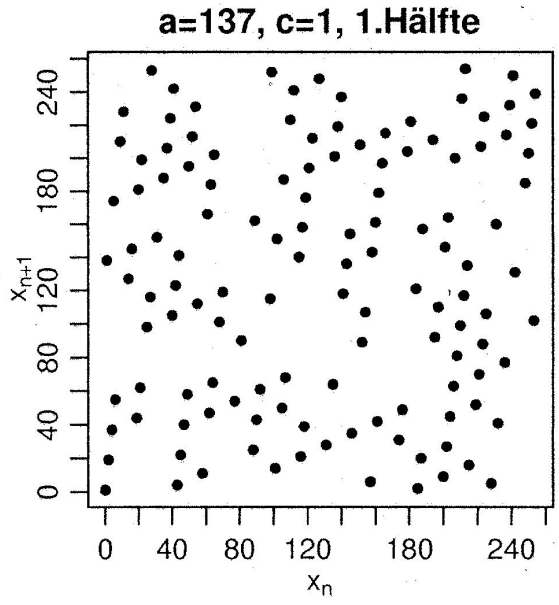
$$(y_j, y_{j+1}, \ldots, y_{j+k-1}), \quad j = 0, 1, 2, \ldots$$

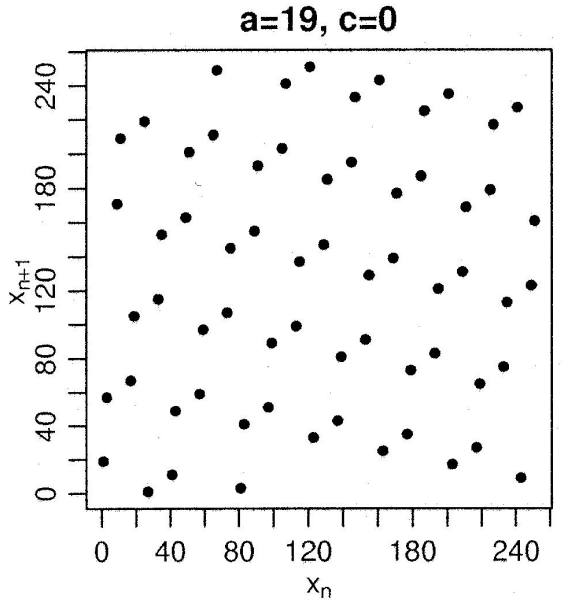
están ubicados en no más de $(k!M)^{1/k}$ hiperplanos paralelos.

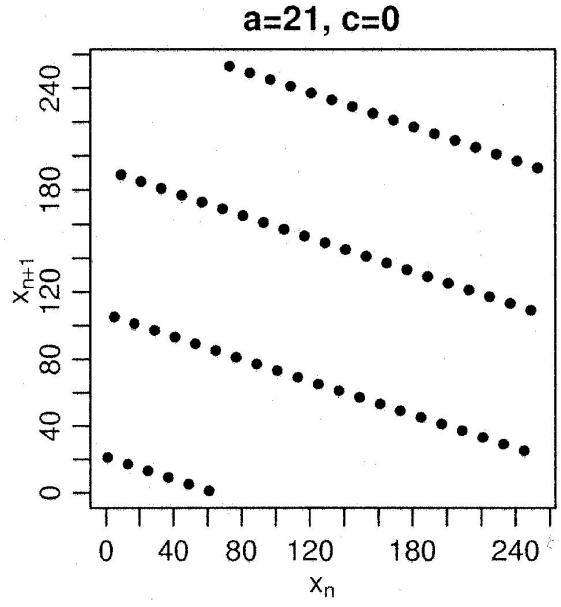
- Cota máxima: $(k!M)^{1/k}$: Estructura de red
- Generador RANDU: Ternas ubicadas en 15 planos paralelos.

Generadores congruenciales, m=256

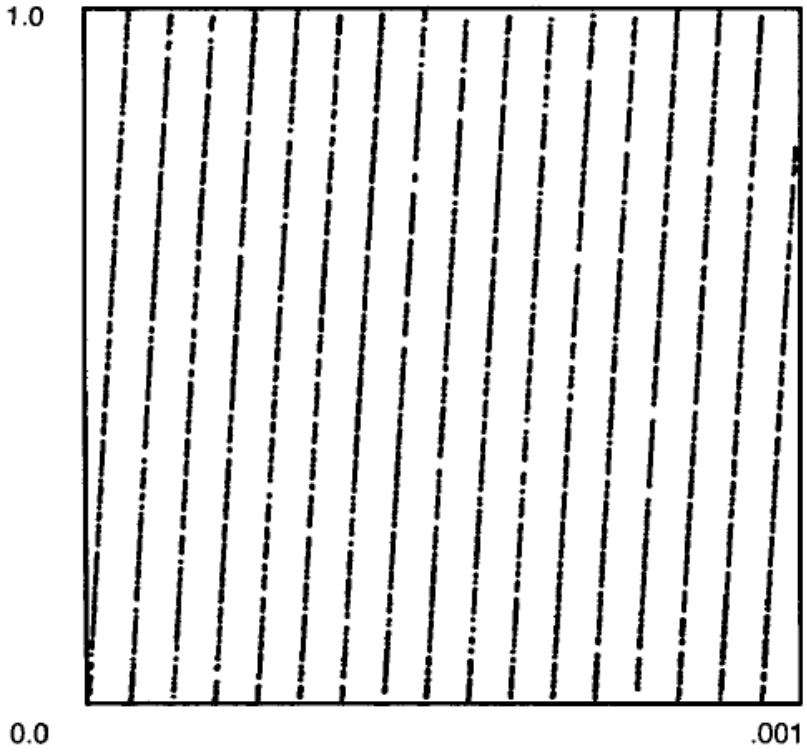




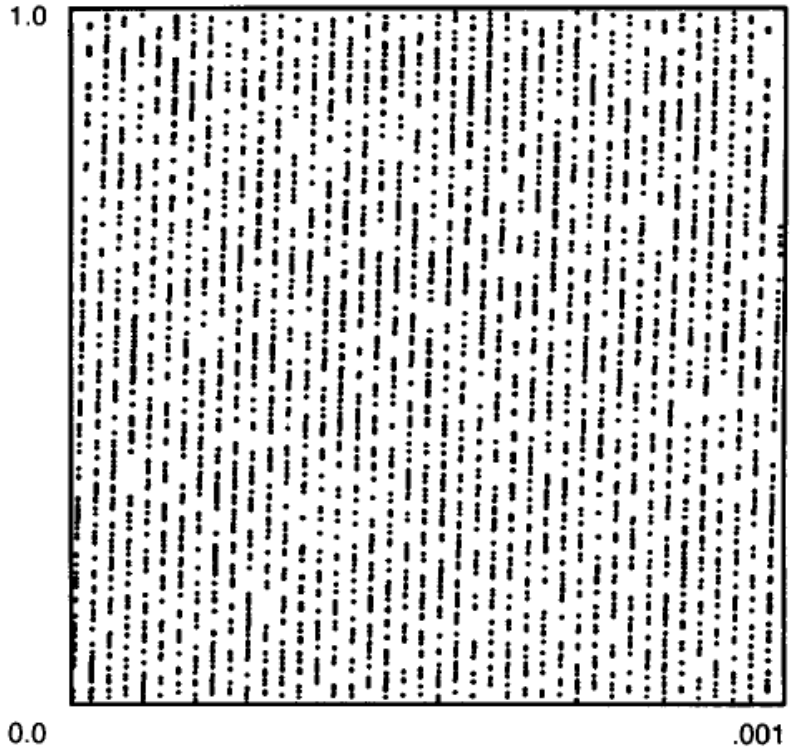




Generadores congruenciales

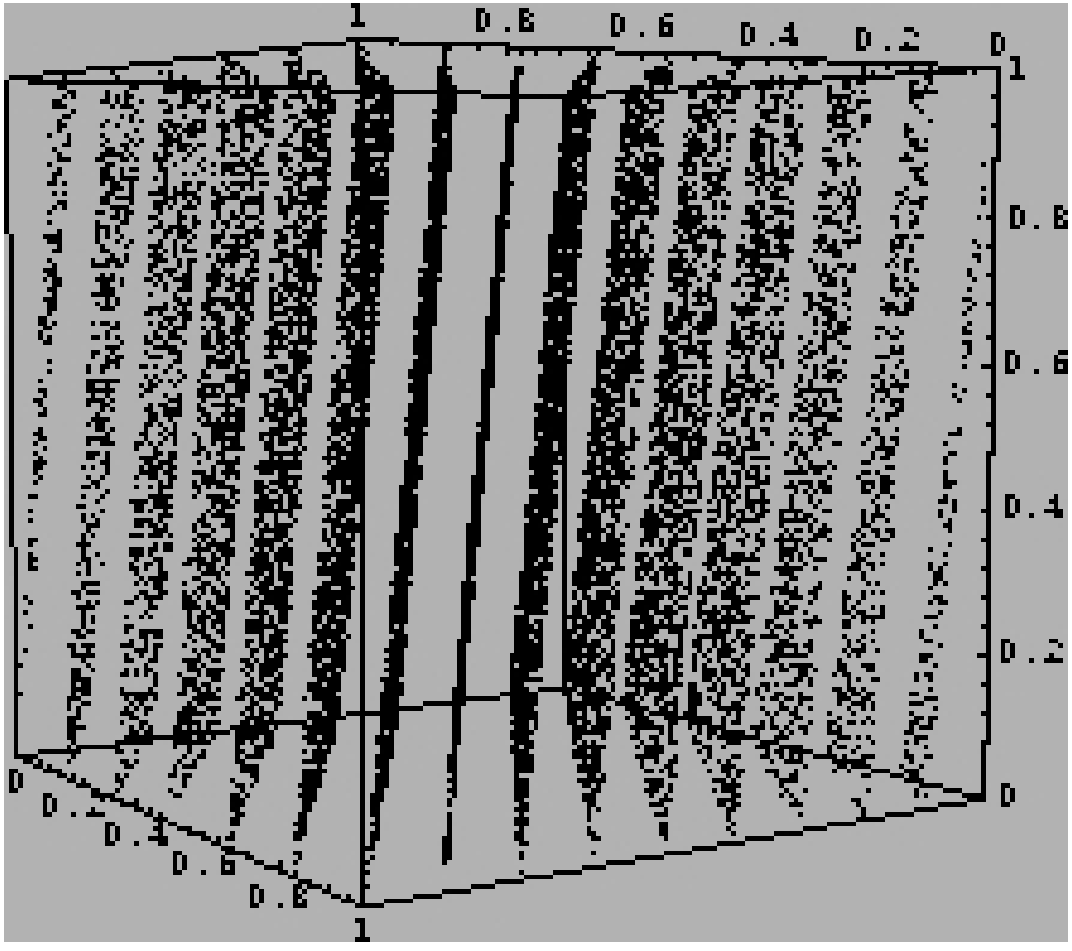


(a) The MLCG with $m = 2^{31} - 1$ and $a = 16807$.



(b) The MLCG with $m = 2147483399$ and $a = 40692$.

RANDU



Generadores portables

$$a = 7^5 = 16807$$

$$M = 2^{31} - 1 = 2147483647$$

- "Buen generador".
- Las multiplicaciones superan el rango de 32 bits.
- Algoritmo de Schrage.

Algoritmo de Schrage

$$M = aq + r$$

$$q = [M/a], \qquad r = M \mod a$$

Si $r < q$ y $0 < z < M - 1$ se puede probar que para todo z, $0 < z < M$:

- $0 \leq a(z \mod q) \leq M - 1$
- $0 \leq r[z/q] \leq M - 1$
-

$$az \mod M = \begin{cases} a(z \mod q) - r[z/q] & \text{si es } \geq 0 \\ a(z \mod q) - r[z/q] + M & \text{c.c.} \end{cases}$$

El generador ran0

$$y_{j+1} = a y_j \mod M$$

$$a = 7^5 = 16807,\ M = 2^{31} - 1 = 2147483647$$

Schrage: se utiliza $q = 127773$ y $r = 2836$
Desventajas:

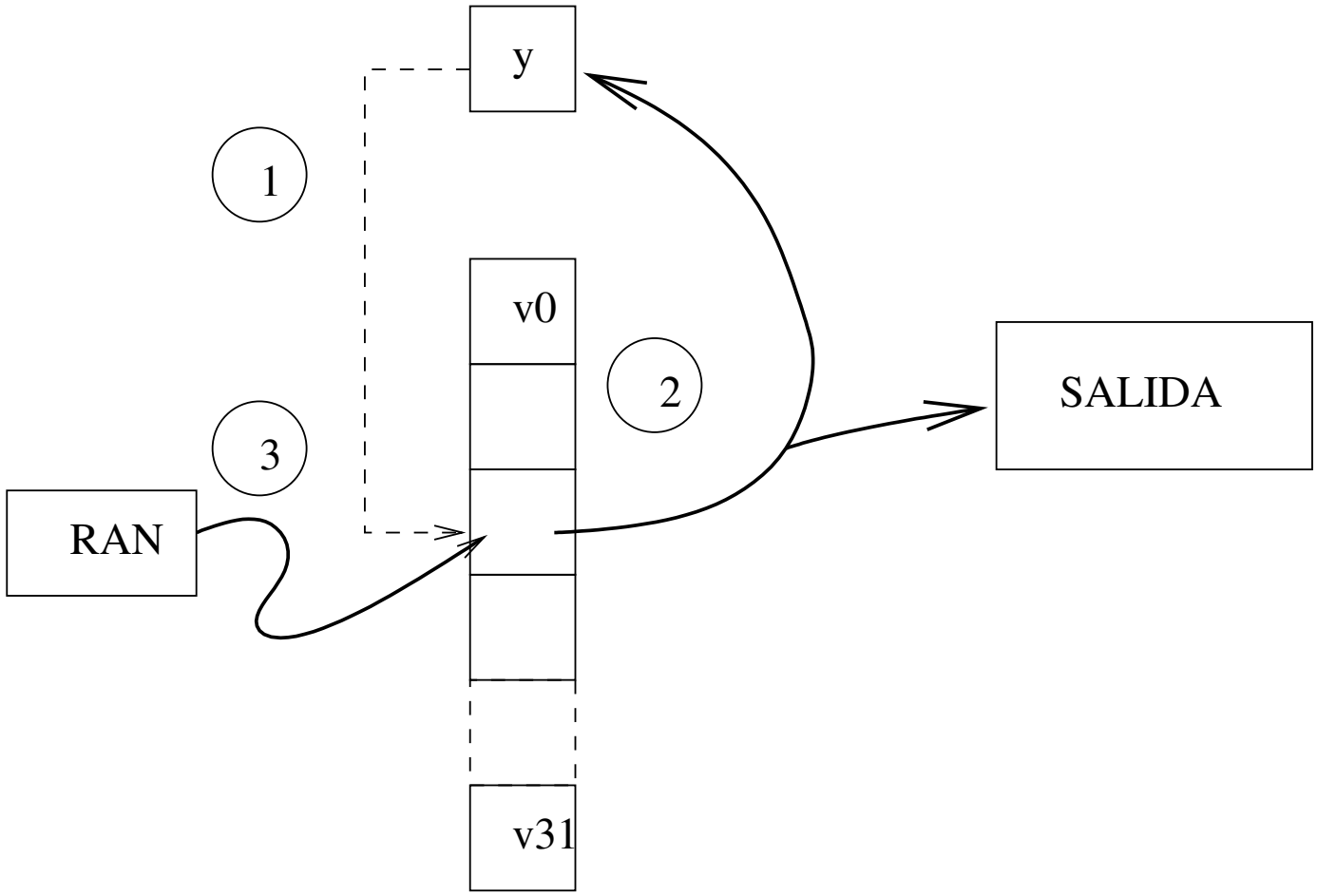
- Sucesiones de números muy pequeños.
- Inconvenientes en el plano: (y_i, y_{i+1}): el test χ^2 falla para $N \approx O(10^7) \leq M - 2$

Shuffling

Se ”almacenan” los últimos valores generados en una tabla, y la salida se obtiene eligiendo aleatoriamente un elemento de dicha tabla y reponiéndolo por un nuevo valor generado.

- El generador ran1.
- $a = 7^5 = 16807$, $M = 2^{31} - 1 = 2147483647$
- Tabla de 32 posiciones.

El generador ran1



Combinación de generadores

Teorema

Sean $W_1, W_2, \ldots, W_n$ variables aleatorias discretas, tales que $W_1 \sim U([0, d-1])$. Entonces

$$W = (\sum_{j=1}^{n} W_j) \mod d$$

es una v.a. uniforme discreta en $[0, d-1]$.

Ejemplo: tirar 2 dados, y sumar módulo 6.

Combinación de congruenciales

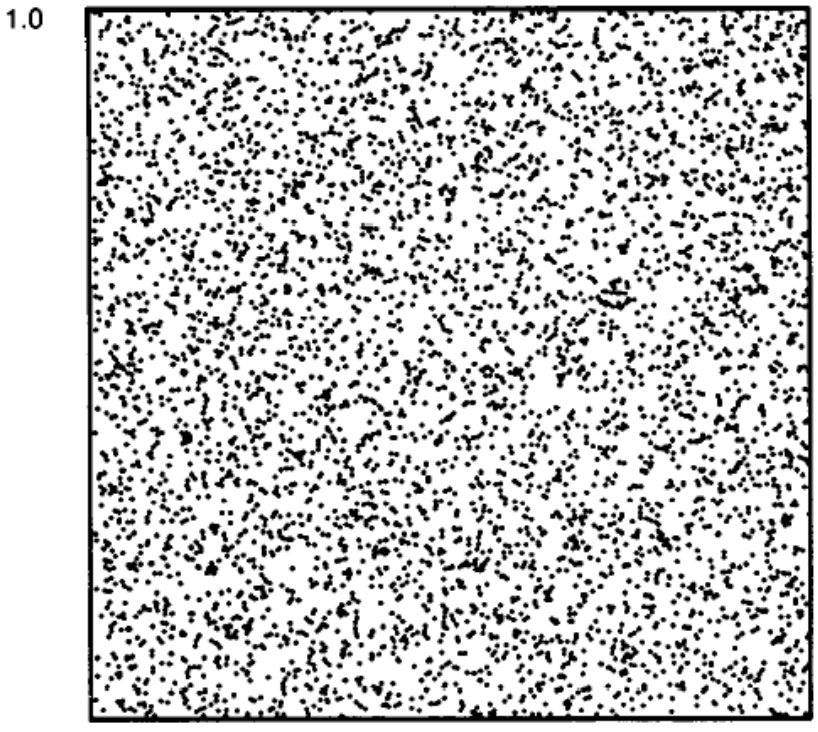
- Combinar secuencias de generadores congruenciales. Sugerencia: restar.
- Se obtiene un generador de v.a. uniformes.
- La longitud de la secuencia es mayor = mínimo común múltiplo de los generadores.

$$x_n = 40014 x_{n-1} \mod 2^{31} - 85$$

$$y_n = 40692 y_{n-1} \mod 2^{31} - 249$$

El 2 es el único factor común.
Período $\approx 2.3 \times 10^{18}$

Combinación de congruenciales



(c) The proposed combined generator.

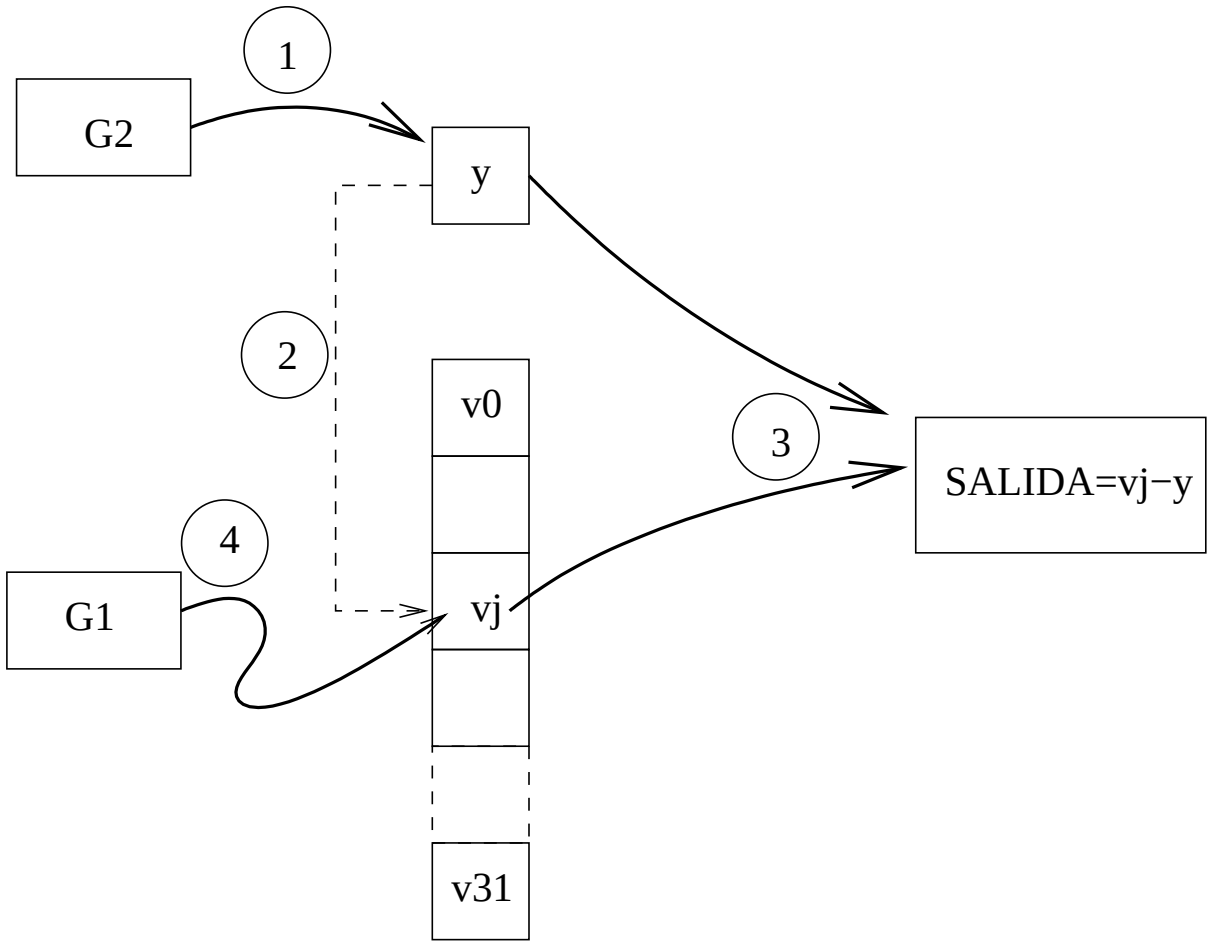
El generador ran2

- Utiliza 2 generadores congruenciales de enteros de 32 bits.
- Tabla de 32 posiciones: *shuffling*.
- Salida = combinación de x e y.

$$G_1 \mapsto \quad x_n = 40014 x_{n-1} \mod 2^{31} - 85$$

$$G_2 \mapsto \quad y_n = 40692 y_{n-1} \mod 2^{31} - 249$$

ran2

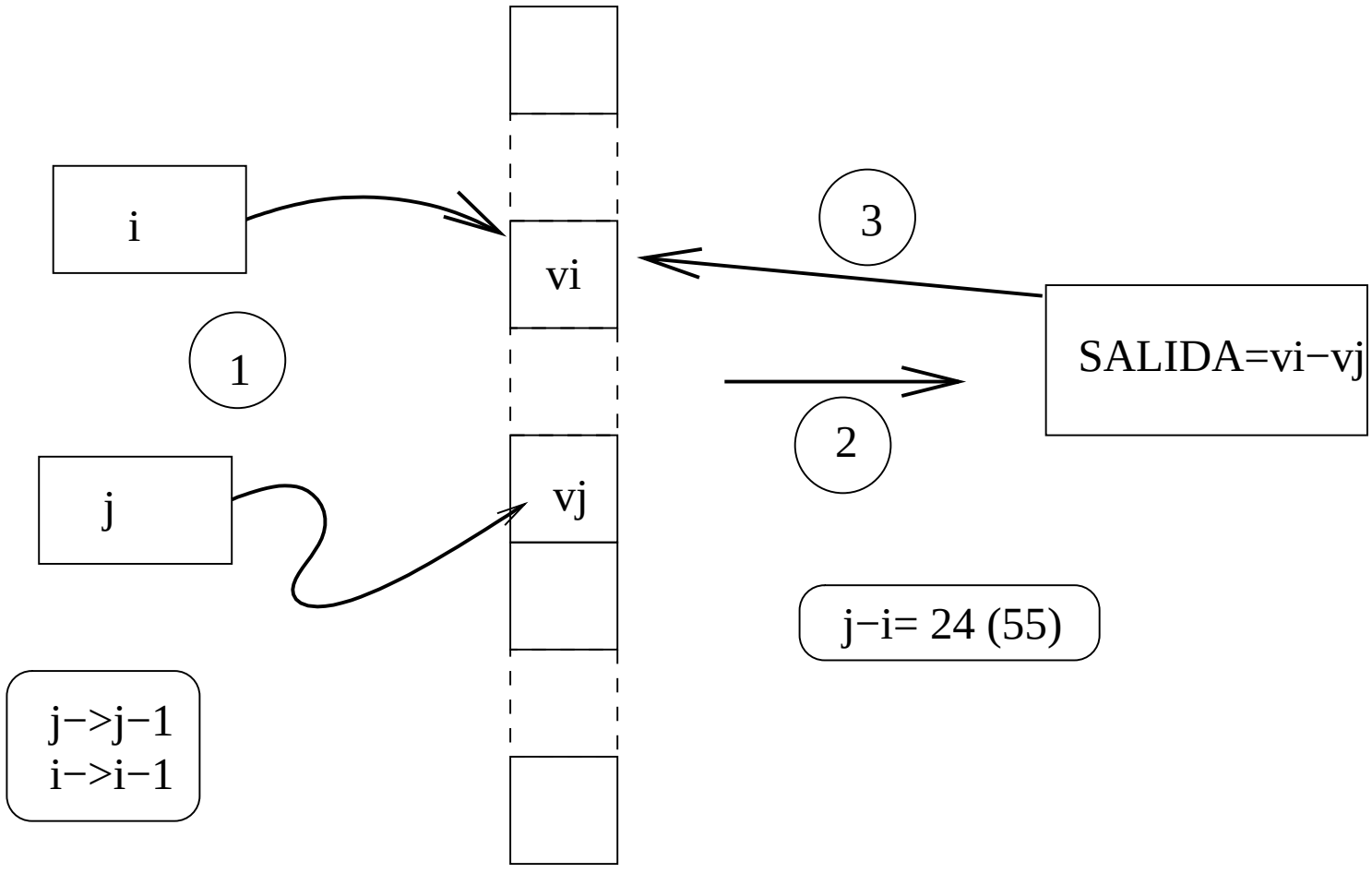


El generador ran3

Ver en Knuth, *Seminumerical Algorithms*, sección 3.2.1.

- Se construye inicialmente una tabla de 55 posiciones, con números aleatorios.
- i y j recorren cíclicamente la tabla, separados por 24 posiciones.
- Salida: diferencia entre los elementos i y j, que reemplaza a su vez el lugar i.
- El ciclo que se obtiene es de longitud 2^{32}.

ran3



Marsaglia - Zaman

Some portable very-long-period random number generators, George Marsaglia and Arif Zaman.

- Sugerencias sobre otros generadores.
- Fibonacci
- Resta con préstamo
- Suma con acarreo
- El generador mzran2

Consideraciones sobre ran2

- La combinación de generadores produce mejoras. Sin embargo, conviene combinar estructuras diferentes.

- Utilizar módulo 32 bits en lugar de 31 bits.
- Utilizar ”primos seguros” (*safeprimes*), con $a = 2^k + \alpha$.

$$2^{31} - 69 \qquad 2^{31} - 535$$

$$2^{32} - 1409, \qquad 2^{32} - 209$$

Consideraciones sobre ran2

- ¿Por qué no un rango de enteros?
- La subrutina iran2( ).

UNI( )=.5+.232830E-9*iran2()

VNI( )=.4656613E-9*iran2()

Algunos ejemplos

Módulo	Secuencia	Período
2^{32}	$x_n = 69069\, x_{n-1} + \text{impar}$	2^{32}
2^{32}	$x_n = x_{n-1} * x_{n-2}$	2^{31}
2^{32}	$x_n = x_{n-1} + x_{n-2} + C$	2^{58}
$2^{31} - 69$	$x_n = x_{n-3} - x_{n-1}$	2^{62}
$2^{32} - 18$	$x_n = x_{n-2} - x_{n-3} - C$	2^{95}

El generador mzran( )

Combina los generadores:

$$x_n = 69069 x_{n-1} + 1013904243 \mod 2^{32}$$

Período: $\approx 2^{32}$

$$x_n = x_{n-3} - x_{n-1} \mod 2^{31} - 69$$

Período: $\approx 2^{94}$
El período es mayor a 2^{94}, o 10^{28}.

El generador mzran13( )

Combina los generadores:

$$x_n = 69069 x_{n-1} + 1013904243 \mod 2^{32}$$

Período: $\approx 2^{32}$

$$x_n = x_{n-2} - x_{n-3} - c' \mod 2^{32} - 18$$

Período: $\approx 2^{95}$
El período es del orden de 2^{125}, o 10^{36}.

Rutina propuesta

```
typedef unsigned long int unlong;
unlong x=521288629, y=362436069, z=16163801,
       c=1, n=1131199209;
unlong mzran13()
{ long int s;
  if (y>x+c) (s=y-(x+c); c=0;}
  else { s=y-(x+c)-18; c=1; }
  x=y; y=z;
  return (z=s) + (n=69069*n+1013904243);
};
void ran13set(unlong xx, unlong yy, unlong zz,
  { x=xx; y=yy; z=zz; n=nn; c=y>z; }
```