Simulación: Conceptos y programación

Rafael Alberto Moreno Parra

La simulación es un método por el cual se puede probar o experimentar o ver qué sucede en determinado proceso, escenario u objeto sin el riesgo que haya consecuencias reales. Esto permite comparar diferentes soluciones ante un problema, probarlas y ver cuál es la mejor, después de eso, se podría aplicar esa solución que funcionó adecuadamente en la simulación, en el mundo real y esperar que las consecuencias que el modelo de simulación muestra, sean las mismas en el mundo real.

Universidad Libre - Cali

Universidad Libre Cali - Colombia - Sur América - Diagonal 37A Nro. 3 - 29 - Santa Isabel -

PBX (+57 2) 524 0007

27/07/2012

Contenido

Dedicatoria

A mi padre, José Alberto Moreno Sánchez;

A mi madre, María del Rosario Parra Espinoza;

A mi hermana, Diana Pilar Moreno Parra;

Y no olvido a mi mascota, Sally, una gata que me ha alegrado la vida.

Agradecimientos

Las siguientes personas les agradezco su colaboración para el desarrollo de este libro:

Matemático Walter Magaña como el par evaluador del libro.

Ing. María Mercedes Sinisterra, directora de la División de Investigación de la Universidad Libre - Cali

Ing. Fabián Castillo, director del Programa de Ingeniería de la Universidad Libre - Cali

Licencia de este libro

Marcas registradas

Universidad Libre – Seccional Cali. Enlace: http://www.unilibrecali.edu.co/home/

En este libro se hace uso de las siguientes tecnologías registradas:

Microsoft ® Windows ® Enlace: http://windows.microsoft.com/en-US/windows/home

Microsoft ® Office ® Enlace: http://office.microsoft.com/en-us/excel

Oracle ® Java ® Enlace: http://www.oracle.com/technetwork/java/index.html

Introducción

La simulación es un método por el cual se puede probar o experimentar o ver qué sucede en determinado proceso, escenario u objeto sin el riesgo que haya consecuencias reales. Luego permite comparar diferentes soluciones ante un problema, probarlas y ver cuál es la mejor, posteriormente, se podría aplicar esa solución que funcionó adecuadamente en la simulación, en el mundo real y esperar que las consecuencias que el modelo de simulación mostró, sean las mismas en el mundo real.

Es por ese motivo que cuando se planea una simulación, ésta debe ser lo más realista posible o acercarse lo suficiente a lo que sucede en la realidad. La expresión: "*la solución funcionaba correctamente en el modelo, no entiendo la razón por la cual no funciona en la realidad*", significa que la simulación no fue realizada correctamente, es decir, el modelo matemático de la situación real tiene fallas como no deducir correctamente como las diferentes variables afectan el todo o en el peor de los casos, olvidar incluir variables fundamentales.

Una de las lecciones que primero se aprenden al diseñar modelos de simulación, es darse cuenta que el mundo real es indeterminista, esto significa que hay variables que no se pueden controlar, a lo sumo estimar su comportamiento; estas variables son conocidas como variables aleatorias, y funcionan con fórmulas en el que el azar es el protagonista.

Uno de los temas más llamativos es mostrar que haciendo uso de modelos de simulación, es posible solucionar problemas complejos, por ejemplo: encontrar el área bajo la curva dibujada por una función algebraica requiere de resolver integrales, una tarea que puede llegar a ser muy difícil, sin embargo, usando el método Monte carlo, encontrar el área bajo cualquier curva es muy fácil, aclarando que es un método que aproxima muy bien el área de la curva, no es un método totalmente exacto.

Muchas personas para entretenerse comienzan a resolver Sudokus, se han planteado Sudokus desde sencillos hasta increíblemente complejos que requieren horas o días de dedicación para resolverlos. ¿Cree que un método usando azar pueda resolver cualquier Sudoku? La respuesta es sí, es un algoritmo muy sencillo, corto y rápido. En este libro se muestra y explica este algoritmo.

Este libro va desarrollando el concepto de simulación, los modelos, cómo se construyen, cuáles son sus partes; entre estas partes están los números aleatorios (el representante del azar) explicando que son, como se obtienen, como se prueban; luego viene la parte de las variables aleatorias y las ecuaciones que las generan (conocida como distribuciones de probabilidad). Una vez que esta explicado el concepto de modelo, el libro continua con las aplicaciones de estos modelos, desde simular procesos industriales como inventario o filas de espera, hasta la solución de problemas en el que el enfoque determinista no es la mejor opción como el problema del viajero.

Estoy abierto a las recomendaciones de los lectores para futuros libros, pueden escribirme a ramsoftware@gmail.com o visitar mi página web en http://darwin.50webs.com

Números aleatorios

Una pieza fundamental en una simulación que requiera el uso del azar, son los números aleatorios. En una hoja de cálculo como Microsoft® Excel®, para tener un número aleatorio es escribir en una celda *=aleatorio()*

Los números aleatorios tienen las siguientes características:

1. Son números reales.
2. Se representan con la letra r.
3. Su rango es [0,1) , es decir, $0 \leq r < 1$
4. Se distribuyen uniformemente entre 0 y 1
5. En una secuencia larga de números aleatorios, el promedio debe ser de 0.5 y su varianza de $\frac{1}{12}$
6. Cada número aleatorio es independiente del anterior y del siguiente.

Ejemplo de números aleatorios:

0,102693	0,72090815	0,8416563	0,01919562	0,30854969	0,84546362
0,97588244	0,42808482	0,63139762	0,35768096	0,18634783	0,63273874
0,40099075	0,70995155	0,54840686	0,77633856	0,23492947	0,39072164
0,6494215	0,9643229	0,56011635	0,32169894	0,20957046	0,80023535
0,3739077	0,05570935	0,3702134	0,69996839	0,29223249	0,56032819
0,26775066	0,1263963	0,48318402	0,61046046	0,64424586	0,23648002
0,97170821	0,06160593	0,62256281	0,85188234	0,54230602	0,9501602
0,49521218	0,57720142	0,25788505	0,06383283	0,40111375	0,48474157
0,78211251	0,59561591	0,29332582	0,56729874	0,61722938	0,10060332
0,35971964	0,39351285	0,702667	0,47877622	0,5915599	0,50977597
0,94311542	0,2871632	0,0423875	0,18247436	0,39593878	0,56613832
0,63109173	0,87552863	0,84786695	0,72808404	0,14191931	0,09556429
0,9967313	0,35229998	0,20860891	0,05803163	0,88940381	0,80649234

Tabla 1: Ejemplo de números aleatorios

Números aleatorios vs números pseudo-aleatorios

Tanto las hojas electrónicas como los lenguajes de programación tienen instrucciones para generar números aleatorios, pero esto requeriría una fuente real de azar puro en el computador y esta no existe dentro de la máquina (los PCs son deterministas). ¿Qué son entonces esos números "aleatorios" generados? Realmente son números generados usando ecuaciones recursivas, por lo tanto, no son números aleatorios, sino pseudo-aleatorios. De todos modos, estos números pseudo-aleatorios cumplen con las características que deben poseer los números aleatorios reales y por consiguiente son válidos para ser utilizados en simulaciones que requieran azar.

Generando números pseudo-aleatorios

Existen varios algoritmos para generar números pseudo-aleatorios. Se exploran varios métodos.

Método congruencial

Hace uso de la siguiente fórmula:

$$X_1 = (A * X_0 + B)\ mod\ N \quad r_1 = X_1/N$$

$$X_2 = (A * X_1 + B)\ mod\ N \quad r_2 = X_2/N$$

$$X_3 = (A * X_2 + B)\ mod\ N \quad r_3 = X_3/N$$

$$X_n = (A * X_{n-1} + B)\ mod\ N$$

A, B, X_0 y N son valores llamados "semillas" que son ingresados por el usuario, a partir de ese punto el algoritmo genera los números pseudo-aleatorios. Y la operación `mod` retorna el residuo de la división.

Este es un ejemplo de ejecución

Xn	A	B	N	Xn+1	r
111	78	45	77	2	0,02597403
2				47	0,61038961
47				15	0,19480519
15				60	0,77922078
60				28	0,36363636
28				73	0,94805195
73				41	0,53246753

41				9	0,11688312

Tabla 2: Ejecución de un generador congruencial

En las celdas A2, B2, C2, D2 se ingresan los valores semilla, en la celda E2 se escribe la operación *=RESIDUO(A2*$B$2+$C$2;$D$2)* y en la celda F2 se escribe la operación *=E2/$D$2* , luego en la celda A3 se escribe la operación *=E2* (de esa forma la ecuación se retroalimenta).

	A	B	C	D	E	F	G
1	Xn	A	B	N	Xn+1	r	
2	111	78	45	77	=RESIDUO(A2*$B$2+$C$2;$D$2)		
3	2				47	0,61038961	
4	47				15	0,19480519	
5	15				60	0,77922078	
6	60				28	0,36363636	

Figura 1: Valores semilla y cálculo del siguiente valor

	A	B	C	D	E	F
1	Xn	A	B	N	Xn+1	r
2	111	78	45	77	2	=E2/$D$2
3	2				47	0,61038961
4	47				15	0,19480519
5	15				60	0,77922078

Figura 2: Cálculo del número pseudo-aleatorio

	A	B	C	D	E	F
1	Xn	A	B	N	Xn+1	r
2	111	78	45	77	2	0,02597403
3	=E2				47	0,61038961
4	47				15	0,19480519

Figura 3: Se instruye la hoja de cálculo para realimentarse del resultado anterior

El generador implementado en Java

```
public class Congruencial
{
  public static void main(String[] args)
  {
    int valX0, valA, valB, valN;

    valX0 = 111;
    valA = 78;
    valB = 45;
    valN = 77;

    for (int genera=1; genera<=100; genera++)
    {
      valX0 = (valX0 * valA + valB) % valN;
      double valR = (double) valX0 / valN;
      System.out.println("r = " + valR);
    }
  }
}
```

Generaría lo siguiente:

```
r = 0.025974025974025976
r = 0.6103896103896104
r = 0.19480519480519481
r = 0.7792207792207793
r = 0.36363636363636365
r = 0.948051948051948
```

Una característica de los números pseudo-aleatorios es el período, como se observa en la imagen:

	A	B	C	D	E	F
1	Xn	A	B	N	Xn+1	r
2	90	23	46	13	10	0,76923077
3	10				3	0,23076923
4	3				11	0,84615385
5	11				0	0
6	0				7	0,53846154
7	7				12	0,92307692
8	12				10	0,76923077
9	10				3	0,23076923

Período

Figura 4: Período de un generador de números pseudo-aleatorios

Finalmente, la realimentación genera un número que ya se había calculado anteriormente por lo que la secuencia se repite nuevamente. En el ejemplo de la imagen el período es demasiado corto por lo que es inútil ese generador con esos valores

semillas para una simulación debido a que si los números aleatorios repiten secuencia, se repiten resultados en el modelo de simulación. Lo ideal es un período muy largo, que supere el número de aleatorios que se necesitan para una simulación, así que se está hablando de periodos con una longitud que se mida en decenas o cientos de millones de números.

Algoritmo de cuadrados medios

Paso 1: Valor Inicial X (semilla)
Paso 2: Se eleva al cuadrado ese valor X
Paso 3: Se extrae los dígitos centrales
Paso 4: Se convierte a un valor entre 0 y 1
Paso 5: Usar el nuevo valor como X y volver al Paso 2

Ejemplo
Semilla = 4561

X_n	X^2	X_{n+1}	r
4561	20**8027**21	8027	0,8027
8027	64**4327**29	4327	0,4327
4327	18**7229**29	7229	0,7229
7229	52**2584**41	2584	0,2584

Tabla 3: Ejecución de un generador de cuadrados medios

Implementar este generador requeriría manipular las cifras del número elevado al cuadrado, lo cual exigiría una serie de cálculos en la CPU y aquí hay otra regla que debe cumplir los generadores de números pseudo-aleatorios: deben ser extremadamente rápidos.

El infame generador RANDU

El generador de números aleatorios llamado RANDU fue escrito por IBM en Fortran, este se popularizó y dio origen a otros generadores con la misma base. Durante décadas se utilizó para variadas pruebas estadísticas, hasta que se descubrió su error.

El algoritmo que genera números enteros era el siguiente:

$$V_{j+1} = 65539 * V_j \; mod \; 2^{31}$$

La secuencia del generador RANDU usando como semilla 1 es el siguiente

1	65539
65539	393225
393225	1769499
1769499	7077969
7077969	26542323
26542323	95552217
95552217	334432395
334432395	1146624417
1146624417	1722371299
1722371299	14608041
14608041	1766175739
1766175739	1875647473
1875647473	1800754131

Tabla 4: Ejecución del generador RANDU

RANDU fue un grave error. Se seleccionó como divisor un número no primo para facilitar la división modular y el multiplicador fue seleccionado por su simplicidad en su representación binaria (10000000000000011_2). Falló en la prueba aleatoria 3D [1]

Mersenne Twister

Desarrollado en 1997 por Makoto Matsumoto y Takuji Nishimura, es un generador de números pseudo-aleatorios [2] con un enorme periodo: 2^{19937}-1. En 2007 tuvo una mejora que lo hacía más rápido (el doble), mejor distribuido y con mejor capacidad para recuperarse de un estado inicial cero [3].

Multiply-with-carry

Es un conjunto de métodos desarrollados por George Marsaglia para generar una serie de números enteros pseudo-aleatorios con enormes periodos: 2^{60} a $2^{20000000}$. Existen implementaciones de este generador en lenguajes de programación como C# [4].

Well Equidistributed Long-period Linear (WELL)

Desarrollado en 2006 por F. Panneton, P. L'Ecuyer, y M. Matsumoto. Se caracteriza por recuperarse mejor de una inicialización de estado cero y tener mejor desempeño en comparación al Mersenne Twister. [5].

Probando los generadores de números aleatorios

Lo sucedido con RANDU deja la enseñanza que hay que hacer una gran serie de pruebas a los generadores de números aleatorios para validar que:

1. Cumplen lo de estar uniformemente distribuidos entre 0 y 1 (o un rango determinado por ejemplo cuando se generan números enteros).

2. Hay independencia entre un número aleatorio con su predecesor y su sucesor.

Se documentan cinco pruebas en este libro.

Prueba del promedio

El promedio de los números aleatorios entre [0,1) debe ser cercano o igual a 0.5

Prueba de varianza

La varianza de los números aleatorios, debe ser cercano o igual a 1/12.

Para calcular la varianza se hace uso de la siguiente fórmula

$$Varianza = \frac{\sum_{i=1}^{n}(Xi - Promedio)^2}{n}$$

Ejemplo

	r	Promedio	Diferencial cuadrado	
	0,67	0,43	0,05818293	
	0,47	0,43	0,00211498	
	0,93	0,43	0,25107677	
	0,12	0,43	0,09562926	
	0,04	0,43	0,15379429	
	0,04	0,43	0,1526288	
	0,31	0,43	0,01278217	
	0,10	0,43	0,10741861	
	0,71	0,43	0,07758388	
	0,38	0,43	0,0023943	
	0,55	0,43	0,01517245	
	0,21	0,43	0,04776856	
	0,02	0,43	0,16620733	
	0,95	0,43	0,27544766	
	0,23	0,43	0,04060265	
	0,41	0,43	0,00029022	
	0,87	0,43	0,19426676	
	0,88	0,43	0,2015445	
	0,08	0,43	0,11854597	
	0,59	0,43	0,02768634	
Promedio	**0,43**			
Total números	**20**			
Varianza			0,10005692	
Varianza ideal			0,08333333	1/12

Tabla 5: Cálculo de la varianza en una secuencia de números aleatorios

Prueba de uniformidad Ji-Cuadrada

La prueba de Ji-cuadrada es una prueba para demostrar, estadísticamente hablando, que los números que produce un generador de números aleatorios tienen (o no) una distribución uniforme.

Se necesitan por lo menos 100 números aleatorios.

Paso 1

Como los números están en el rango 0 y 1 entonces se divide este rango en un número finito de subrangos (mínimo deben ser 5), en el ejemplo se usarán 10 subrangos.

Rango	**Significa**
de 0 a 0,1	0 <= n < 0,1
de 0,1 a 0,2	0,1 <= n < 0,2
de 0,2 a 0,3	0,2 <= n < 0,3
de 0,3 a 0,4	0,3 <= n < 0,4
de 0,4 a 0,5	0,4 <= n < 0,5
de 0,5 a 0,6	0,5 <= n < 0,6
de 0,6 a 0,7	0,6 <= n < 0,7
de 0,7 a 0,8	0,7 <= n < 0,8
de 0,8 a 0,9	0,8 <= n < 0,9
de 0,9 a 1,0	0,9 <= n < 1

Paso 2

Se lanza el generador de números aleatorios. En el ejemplo se hace unas mil veces. Luego se cuenta cuantos números caen en cada subrango.

Rango	Números aleatorios en cada rango
de 0 a 0,1	91
de 0,1 a 0,2	102
de 0,2 a 0,3	113
de 0,3 a 0,4	110
de 0,4 a 0,5	87
de 0,5 a 0,6	95
de 0,6 a 0,7	98
de 0,7 a 0,8	104
de 0,8 a 0,9	117
de 0,9 a 1,0	83

Paso 3

En una distribución uniforme ideal la cantidad de números que caen en cada subrango debería ser = Total Números / # SubRangos, es decir, 1000 / 10 = 100

Rango	frecuencia obtenida FO	frecuencia esperada FE
de 0 a 0,1	91	100
de 0,1 a 0,2	102	100
de 0,2 a 0,3	113	100
de 0,3 a 0,4	110	100
de 0,4 a 0,5	87	100
de 0,5 a 0,6	95	100
de 0,6 a 0,7	98	100
de 0,7 a 0,8	104	100
de 0,8 a 0,9	117	100
de 0,9 a 1,0	83	100

Paso 4

Se aplica la siguiente operación ((FE-FO)^2)/FE

Rango	frecuencia obtenida FO	frecuencia esperada FE	((FE-FO)^2)/FE
de 0 a 0,1	91	100	((100-91)^2)/100 = 0,81
de 0,1 a 0,2	102	100	0,04
de 0,2 a 0,3	113	100	1,69
de 0,3 a 0,4	110	100	1
de 0,4 a 0,5	87	100	1,69
de 0,5 a 0,6	95	100	0,25
de 0,6 a 0,7	98	100	0,04
de 0,7 a 0,8	104	100	0,16
de 0,8 a 0,9	117	100	2,89
de 0,9 a 1,0	83	100	2,89
		Sumatoria	**11,46**

Paso 5

Nivel de confianza Alpha (entre más alto, más confiable). Alpha = 0,05

Paso 6

Como son 10 rangos, se tiene 10-1=9 grados de libertad.

Paso 7

Se busca en la tabla cruzando Alpha y los grados de libertad

	Alpha	0,005	0,01	0,025	0,05	0,10
Grados de Libertad						
1		7,88	6,63	5,02	3,84	2,71
2		10,60	9,21	7,38	5,99	4,61
3		12,84	11,34	9,35	7,81	6,25
4		14,96	13,28	11,14	9,49	7,78
5		16,7	15,1	12,8	11,1	9,2
6		18,5	16,8	14,4	12,6	10,6
7		20,3	18,5	16,0	14,1	12,0
8		22,0	20,1	17,5	15,5	13,4
9		23,6	21,7	19,0	**16,9**	14,7
10		25,2	23,2	20,5	18,3	16,0
11		26,8	24,7	21,9	19,7	17,3
12		28,3	26,2	23,3	21,0	18,5
13		29,8	27,7	24,7	22,4	19,8
14		31,3	29,1	26,1	23,7	21,1
15		32,8	30,6	27,5	25,0	22,3
16		34,3	32,0	28,8	26,3	23,5
17		35,7	33,4	30,2	27,6	24,8
18		37,2	34,8	31,5	28,9	26,0
19		38,6	36,2	32,9	30,1	27,2
20		40,0	37,6	34,2	31,4	28,4
21		41,4	38,9	35,5	32,7	29,6
22		42,8	40,3	36,8	33,9	30,8
23		44,2	41,6	38,1	35,2	32,0
24		45,6	43,0	39,4	36,4	33,2
25		49,6	44,3	40,6	37,7	34,4
26		48,3	45,6	41,9	38,9	35,6
27		49,6	47,0	43,2	40,1	36,7
28		51,0	48,3	44,5	41,3	37,9
29		52,3	49,6	45,7	42,6	39,1
30		53,7	50,0	47,0	43,8	40,3
40		66,8	63,7	59,3	55,8	51,8
50		79,5	76,2	71,4	67,5	63,2
60		92,0	88,4	83,3	79,1	74,4
70		104,2	100,4	95,0	90,5	85,5
80		116,3	112,3	106,6	101,9	96,6
90		128,3	124,1	118,1	113,1	107,6
100		140,2	135,8	129,6	124,3	118,5

Valores tomados del libro "Discrete Events System Simulation" de J. Banks. Ed. Prentice Hall [6]

Explicación

Alfa es el nivel de significación utilizado para calcular el nivel de confianza. El nivel de confianza es igual a 100*(1 - alfa)%, es decir, un alfa de 0.05 indica un nivel de confianza de 95%.

Suponga que se observa que en la muestra de 50 personas que realizan diariamente un trayecto, la distancia media de viaje es 30 minutos con una desviación estándar de la población de 2,5. Con alfa = 0.05, INTERVALO.CONFIANZA(0,05; 2,5; 50) devuelve 0,69291. El intervalo de confianza correspondiente será entonces 30 ± 0,69291

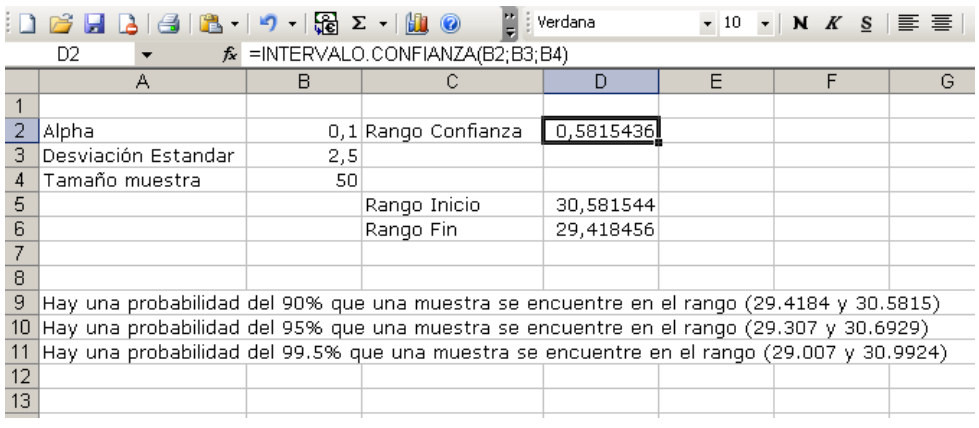

Figura 5: Rango de confianza

D2 =INTERVALO.CONFIANZA(B2;B3;B4)

	A	B	C	D	E	F	G
1							
2	Alpha	0,005	Rango Confianza	0,9924363			
3	Desviación Estandar	2,5					
4	Tamaño muestra	50					
5			Rango Inicio	30,992436			
6			Rango Fin	29,007564			
7							
8							
9	Hay una probabilidad del 90% que una muestra se encuentre en el rango (29.4184 y 30.5815)						
10	Hay una probabilidad del 95% que una muestra se encuentre en el rango (29.307 y 30.6929)						
11	Hay una probabilidad del 99.5% que una muestra se encuentre en el rango (29.007 y 30.9924)						
12							

Figura 6: Rango de confianza.

Paso 8

Comparación. Como 11.46 (lo obtenido en la sumatoria) < 16.9 (el valor constante de la tabla) entonces se acepta la hipótesis que el generador pasa esta prueba de uniformidad.

Prueba de Uniformidad de Kolmogorov-Smirnov

Es importante el tamaño de la muestra, en el ejemplo se generarán los cien primeros números aleatorios con un generador congruencial con estos valores

X_0=178
A=791
B=107
N=1000

Tiene un periodo de 250.

X	A	B	N	R	Aleatorio
178	791	107	1000	905	0,905
905				962	0,962
962				49	0,049
49				866	0,866
866				113	0,113
113				490	0,49
490				697	0,697
697				434	0,434
434				401	0,401
401				298	0,298
298				825	0,825
825				682	0,682
682				569	0,569
569				186	0,186
186				233	0,233

Paso 1

Como los números están en el rango 0 y 1 entonces se divide este rango en un número finito de subrangos, en el ejemplo se usarán 10 subrangos.

Rango	Significa
de 0 a 0,1	0 <= n < 0,1
de 0,1 a 0,2	0,1 <= n < 0,2
de 0,2 a 0,3	0,2 <= n < 0,3
de 0,3 a 0,4	0,3 <= n < 0,4
de 0,4 a 0,5	0,4 <= n < 0,5
de 0,5 a 0,6	0,5 <= n < 0,6
de 0,6 a 0,7	0,6 <= n < 0,7
de 0,7 a 0,8	0,7 <= n < 0,8
de 0,8 a 0,9	0,8 <= n < 0,9
de 0,9 a 1,0	0,9 <= n < 1

Paso 2

Se lanza el generador de números aleatorios, en el ejemplo se hace unas mil veces. Luego se cuenta cuantos números caen en cada subrango.

Rango	Total de números pseudo-aleatorios en cada rango
de 0 a 0,1	12
de 0,1 a 0,2	14
de 0,2 a 0,3	9

de 0,3 a 0,4	6
de 0,4 a 0,5	14
de 0,5 a 0,6	11
de 0,6 a 0,7	9
de 0,7 a 0,8	10
de 0,8 a 0,9	8
de 0,9 a 1,0	7

Paso 3

Se deduce la frecuencia obtenida acumulada

Rango	frecuencia obtenida FO	frecuencia obtenida acumulada	Probabilidad obtenida acumulada
de 0 a 0,1	12	12	0,12
de 0,1 a 0,2	14	26	0,26
de 0,2 a 0,3	9	35	0,35
de 0,3 a 0,4	6	41	0,41
de 0,4 a 0,5	14	55	0,55
de 0,5 a 0,6	11	66	0,66
de 0,6 a 0,7	9	75	0,75
de 0,7 a 0,8	10	85	0,85
de 0,8 a 0,9	8	93	0,93
de 0,9 a 1,0	7	100	1

Paso 4

Se compara con la probabilidad esperada (como son diez rangos y debe ser uniforme, lo ideal es que hubiesen el mismo número de elementos por rango).

Rango	frecuencia esperada FE	frecuencia esperada acumulada	Probabilidad esperada acumulada
de 0 a 0,1	10	10	0,1
de 0,1 a 0,2	10	20	0,2
de 0,2 a 0,3	10	30	0,3
de 0,3 a 0,4	10	40	0,4
de 0,4 a 0,5	10	50	0,5
de 0,5 a 0,6	10	60	0,6
de 0,6 a 0,7	10	70	0,7
de 0,7 a 0,8	10	80	0,8
de 0,8 a 0,9	10	90	0,9
de 0,9 a 1,0	10	100	1

Paso 5

Se compara la probabilidad obtenida acumulada y la probabilidad esperada acumulada y se busca cual tuvo la mayor diferencia.

Rango	probabilidad obtenida acumulada	probabilidad esperada acumulada	Diferencia
de 0 a 0,1	0,12	0,1	0,02
de 0,1 a 0,2	0,26	0,2	0,06
de 0,2 a 0,3	0,35	0,3	0,05
de 0,3 a 0,4	0,41	0,4	0,01
de 0,4 a 0,5	0,55	0,5	0,05
de 0,5 a 0,6	0,66	0,6	0,06
de 0,6 a 0,7	0,75	0,7	0,05
de 0,7 a 0,8	0,85	0,8	0,05
de 0,8 a 0,9	0,93	0,9	0,03
de 0,9 a 1,0	1	1	0
		Mayor diferencia	0,06

Paso 6

Nivel de confianza Alpha (entre más alto, más confiable). Alpha = 0,05

Paso 7

Se debe recordar que el tamaño de la muestra es 100

Paso 8

E busca en la tabla cruzando Alpha y el tamaño de la muestra

Prueba de Smirnov-Kolmogorov. Valores críticos $D_{max p}(\alpha,n)$					
	Nivel de significancia α				
Tamaño de la muestra	.20	.15	0.10	0.05	0.01
1	.900	.925	.950	.875	.995
2	.684	.726	.776	.842	.929
3	.565	.597	.642	.708	.828
4	.494	.525	.564	.624	.733
5	.446	.474	.510	.565	.669
6	.410	.436	.470	.521	.618
7	.381	.405	.438	.486	.577
8	.358	.381	.411	.457	.543
9	.339	.360	.388	.432	.514
10	.322	.342	.368	.410	.490
11	.307	.326	.352	.391	.468
12	.295	.313	.338	.375	.450
13	.284	.302	.325	.361	.433
14	.274	.292	.314	.349	.418
15	.266	.283	.304	.338	.404
16	.258	.274	.295	.328	.392
17	.250	.266	.286	.318	.381
18	.244	.259	.278	.309	.371
19	.237	.252	.272	.301	.363
20	.231	.246	.264	.294	.356
25	.210	.220	.240	.270	.320
30	.190	.200	.220	.240	.290
35	.18	.190	.201	.230	.270
≥ 35	$1.07/\sqrt{N}$	$1.14/\sqrt{N}$	$1.22/\sqrt{N}$	$1.36/\sqrt{N}$	$1.63/\sqrt{N}$

Como el tamaño de la muestra es 100 y alpha es 0,05 entonces el número buscado es 1.36 / RaizCuadrada(100) = 0.136

Paso 9

Comparando la mayor diferencia con el número obtenido en la tabla. Como 0.06 < 0.136 entonces se acepta la hipótesis que los datos tienen la distribución U(0,1), luego el generador es bueno en uniformidad.

Pruebas de Independencia

Supóngase que un generador de números aleatorios ha dado 400 números (del 1 al 1000), uniformemente repartidos pero del primer número al 200avo número está justo por debajo de 500 y del 201avo número al 400avo están por encima de 500. Eso es raro, no suena a azar. Luego se debe hacer la prueba de independencia.

Prueba de Póker

Esta es una prueba de independencia que se basa en la frecuencia con que se repiten los dígitos en los aleatorios generados, por ejemplo: en los aleatorios: 0.345, 0.777, 0.945, 0.003, 0.478

Se pueden observar los siguientes casos:

a) Los tres dígitos son iguales (0.777)

b) Los tres son diferentes (0.478, 0.345, 0.945)

c) Existe un par de iguales (0.003)

Si se examina una muestra independiente se espera que los dígitos que componen los números estén repartidos al azar, de manera similar a cuando se reparte una "mano" de Póker donde se espera que las cartas estén distribuidas al azar en un juego legal, por eso el nombre de la prueba.

Para aplicar la prueba es necesario estos pasos:

1. Saber la cantidad de dígitos que formarán los aleatorios que se desean probar.

2. Clasificar los casos posibles que se pueden formar (pares de iguales, tercias, etc.).

3. Calcular las probabilidades de que en esos números se presenten los casos que se determinaron.

4. Generar una muestra de aleatorios con el generador a probar y clasificar la frecuencia que presentaron los casos en la muestra.

5. Efectuar una prueba ji-cuadrada para verificar si existe evidencia estadística para afirmar que las frecuencias observadas son diferentes a las esperadas. En caso contrario, no se rechazará la hipótesis de que el generador produce aleatorios independientes.

Ejemplo

Determine si hay independencia en los aleatorios generados para el siguiente caso (con un 5% de significancia y aplicando la prueba de Póker).

Hay 1000 números de tres dígitos y después de analizarlos se tienen los siguientes resultados:

700 con tres dígitos diferentes.

273 con un par de dígitos iguales.

27 con dígitos iguales.

Caso	Frecuencia obtenida FO	Frecuencia Esperada FE	((FE-FO)^2)/FE
3 dígitos diferentes	700	720	0,555
2 dígitos iguales	273	270	0,033
dígitos iguales	27	10	28,9
		Sumatoria	29,4888

Grados de libertad: 3 - 1 = 2 y Alpha=0.05 consultando en la tabla el cruce de estos dos valores se obtiene el valor 5.99

Como 29,4888 > 5,99 se rechaza la independencia.

Nota: Hay que determinar las frecuencias esperadas cuando hay dos, cuatro, cinco o más dígitos.

Prueba de Wald-Wolfowitz

Supóngase que se tiene la siguiente secuencia FFHHHFFHFFFFFHHF , el número de secuencias es R=7 que resulta de la siguiente agrupación FF HHH FF H FFFFF HH F

Si fuese la cadena así: HHHHHHFFFFFFFFFF entonces R=2 porque hay dos agrupaciones HHHHHH FFFFFFFFFF

Para probar que un generador de números pseudoaleatorios pasa esta prueba debe seguir estos pasos:

Paso 1

Se genera los números aleatorios (mínimo 20). Por ejemplo estos generados por la función Aleatorio() de Excel.

0,13115648
0,15542105
0,86586089
0,87441265
0,2733703
0,84422255
0,18411069
0,72996966
0,40422589
0,61401405
0,187506
0,71442666
0,829512
0,35673842
0,64474038
0,82786857
0,28769922
0,84599494
0,69417319
0,77368306

Paso 2

Se calcula la media (promedio) de esos números, en el ejemplo es 0,561955333

Paso 3

Se compara el valor de la media con cada número, si es superior se pone un +, si es inferior se pone un -

0,13115648 -
0,15542105 -
0,86586089 +
0,87441265 +
0,27337030 -
0,84422255 +
0,18411069 -
0,72996966 +
0,40422589 -

```
0,61401405 +
0,18750600 -
0,71442666 +
0,82951200 +
0,35673842 -
0,64474038 +
0,82786857 +
0,28769922 -
0,84599494 +
0,69417319 +
0,77368306 +
```

En otras palabras es --++-+-+-+-++-++-+++ luego R=14 Con N1=8 (número de -), N2=12 (número de +) y N=20 (todos los elementos)

Paso 4

Se calcula la media de R

$$Media = \frac{2 * N_1 * N_2}{N + 1} = \frac{2 * 8 * 12}{20 + 1} = 10.6$$

Paso 5

Se calcula la varianza de R

$$Varianza = \frac{(Media - 1) * (Media - 2)}{N - 1} = \frac{(10.6 - 1) * (10.6 - 2)}{20 - 1} = 4.345263158$$

Paso 6

Se calcula Z

$$Z = \frac{R - Media}{\sqrt{Varianza}} = \frac{14 - 10.6}{\sqrt{4.345263158}} = 1.631063523$$

Paso 7

Nivel de confianza de 100*(1- Alpha)% . Para el ejemplo Alpha=0.05

Paso 8

Se busca en la tabla de distribución normal de Z para Alpha/2 (es decir 0.025) y se observa los números -1.96 (izquierda) y 1.96 (derecha). Como el resultado de Z fue 1.63106352 que está entre esos dos valores (-1.96 y 1.96) entonces se acepta que es aleatorio.

Variables Aleatorias

En un modelo de simulación, lo que no es controlable (la variable independiente) se le conoce como variable aleatoria. Durante la simulación, esta variable recibe valores al azar, pero no cualquier azar, sino dependiendo de una distribución. En otras palabras, se genera un número al azar entre 0 y 1, con ese número se deduce un valor para esa variable aleatoria utilizando para ello la función de distribución que tenga.

Por ejemplo, en la experiencia de lanzar monedas, los posibles resultados son {cara, cruz}, y sus probabilidades son {1/2, 1/2}. En la experiencia de lanzar dados, los resultados posibles son {1, 2, 3, 4, 5, 6} y sus probabilidades respectivas son {1/6, 1/6, 1/6, 1/6, 1/6, 1/6}.

Se hace girar una ruleta y se apunta el número del sector que coincide con la flecha. En la ruleta de la izquierda de la figura los resultados posibles son {0, 1, 2, 3, 4, 5, 6, 7}, y la probabilidad de cada resultado es 1/8. En la ruleta de la derecha de la figura los posibles resultados son {0, 1, 2, 3}, y las probabilidades respectivas {1/4, 1/2, 1/8, 1/8}, proporcionales al ángulo del sector.

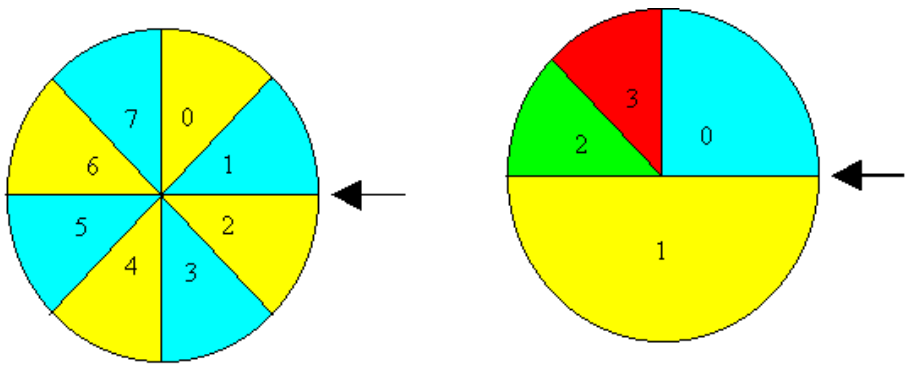

Figura 7: Ruletas con diferentes probabilidades

En los tres primeros ejemplos, la variable aleatoria X se dice que está uniformemente distribuida, ya que todos los resultados tienen la misma probabilidad. Sin embargo, en el último ejemplo, la variable aleatoria X, no está uniformemente distribuida.

Variable aleatoria discreta

Para simular la ruleta situada a la derecha de la figura, se procede del siguiente modo: se hallan las probabilidades de cada resultado, proporcionales al ángulo de cada sector y se apuntan en la segunda columna, la suma total debe de dar la unidad. En la tercera columna, se escriben las probabilidades acumuladas.

Resultado	Probabilidad	Probabilidad acumulada
0	0.25	0.25
1	0.5	0.75
2	0.125	0.875
3	0.125	1

Se sortea un número aleatorio r uniformemente distribuido en el intervalo [0, 1), el resultado del sorteo se muestra en la figura. En el eje X se sitúan los distintos resultados que son nombrados X_0, X_1, X_2, X_3. En el eje vertical, las probabilidades en forma de segmentos verticales de longitud igual a la probabilidad Pi de cada uno de los resultados. Dichos segmentos se ponen unos a continuación de los otros, encima su respectivo resultado Xi. Se obtiene así una función escalonada. Cuando se sortea una variable aleatoria g, se traza una recta horizontal cuya ordenada sea g. Se busca el resultado cuya abscisa sea la intersección de dicha recta horizontal y del segmento vertical, tal como se señala con flechas en la figura. Si el número aleatorio r está comprendido entre 0.25 y 0.75 se obtiene el resultado denominado X1.

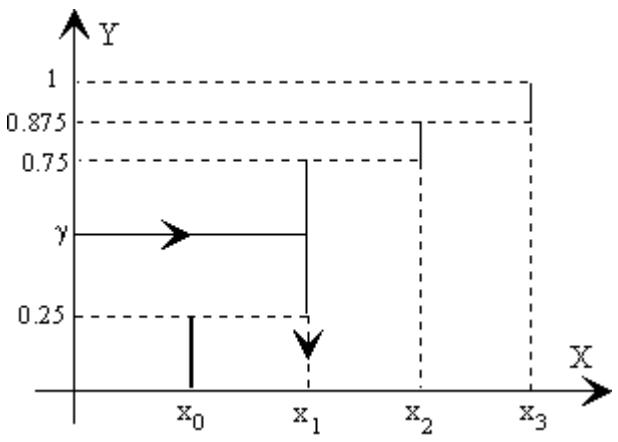

Figura 8: Obteniendo los valores de x en una distribución discreta

La tabla describe el sorteo de una variable discreta, siendo g una variable aleatoria uniformemente distribuida en el intervalo [0,1).

Condición	Resultado
0 <= r <0.25	0
0.25 <= r < 0.75	1
0.75 <= r < 0.875	2
0.875 <= r < 1	3

Variable aleatoria continua

La probabilidad de esta variable está dada por una ecuación, luego al sortear un número aleatorio **Y** uniformemente distribuido en el intervalo [0, 1), se obtiene la **X** correspondiente.

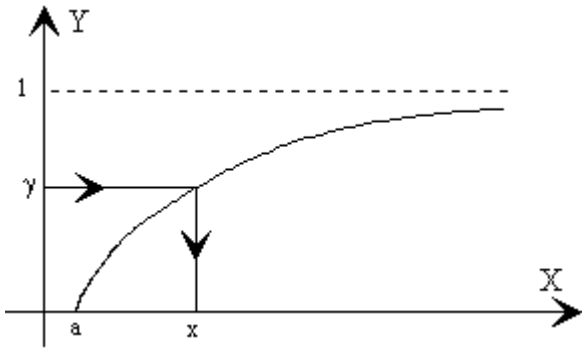

Figura 9: Obteniendo los valores de x en una distribución continua

Una de las ecuaciones más usadas es la exponencial (utilizada en simulación de colas) que es así:

$$F(X) = 1 - e^{-\alpha * x}$$

El problema aquí radica en que se da el valor de F(X) y se debe encontrar x, luego la tarea es despejar x.

$$x = \frac{In(1 - F(x))}{-\alpha}$$

Distribuciones de variables aleatorias continuas

Transformada Inversa

Si X es una variable aleatoria continua con función de densidad dada por f(x), la función de distribución acumulativa F(x) está dada:

$$F(x) = \int_{-\infty}^{\infty} f(x)dx$$

F(x) = r Hacer que la función distributiva F(x) sea igual a un número aleatorio.

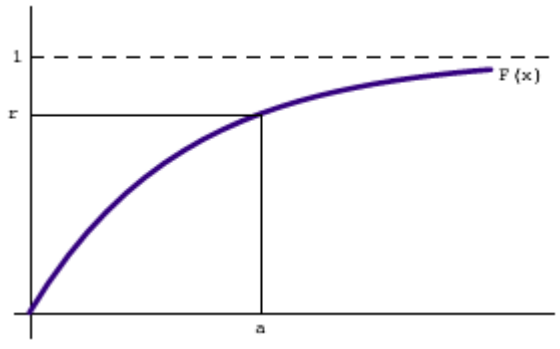

Figura 10: Distribución acumulativa

Ejemplo:

Un viaje entre la ciudad A y la ciudad B puede demorarse entre:

0 y 2 horas con una probabilidad de 1/6

2 y 3 horas con una probabilidad de 1/3

3 y 7 horas con una probabilidad de 1/12

Se quiere simular ese viaje, luego se genera un número al azar entre 0 y 1, dando por ejemplo 0.3 ¿Cuánto tarda el viaje? (cabe recordar que es una función continua).

$$f(x) = \begin{cases} \frac{1}{6}\ si\ 0 \leq x \leq 2 \\ \frac{1}{3}\ si\ 2 \leq x \leq 3 \\ \frac{1}{12}\ si\ 3 \leq x \leq 7 \end{cases}$$

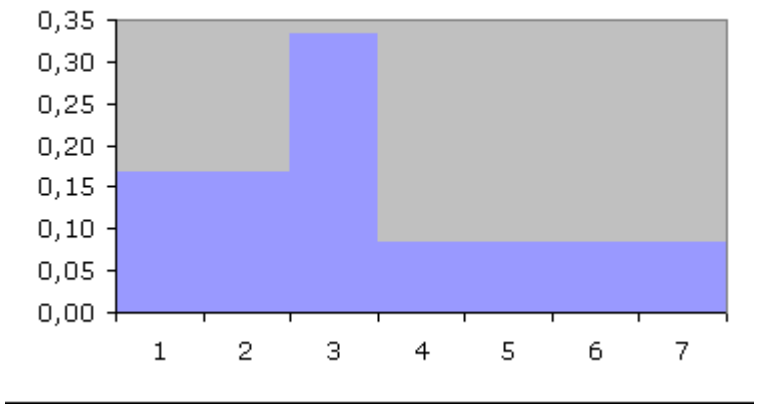

Figura 11: El área en azul suma 1

$$F(x) = \int_{-\infty}^{\infty} f(x)dx = \int_{-\infty}^{0} f(x)dx + \int_{0}^{\infty} f(x)dx$$

$$F(x) = 0 + \int_{0}^{\infty} f(x)dx \ = \int_{0}^{\infty} f(x)dx$$

Porque se evalúa el área entre 0 y x, luego

$$0 \leq x < 2 \ \Rightarrow \ F(x) = \int_{0}^{x} f(x)dx = \int_{0}^{x} \frac{1}{6}dx = \frac{x}{6}$$

Evaluando para F(2) = $\frac{1}{3}$, entonces se sabe que $0 \leq F(x) < \frac{1}{3}$

$$2 \leq x < 3 \Rightarrow F(x) = \int_0^2 f(x)dx + \int_2^x g(x)dx$$

$$F(x) = \frac{1}{3} + \int_2^x g(x)dx = \frac{1}{3} + \int_2^x \frac{1}{3}dx = \frac{1}{3} + \frac{x}{3}\Big|_0^x = \frac{1}{3} + \frac{x}{3} - \frac{2}{3} = \frac{x-1}{3}$$

Evaluando para F(3) = $\frac{2}{3}$, entonces se sabe que $\frac{1}{3} \leq F(x) < \frac{2}{3}$

$$3 \leq \mathrm{x} < 7 \Rightarrow \mathrm{F(x)} = \int_0^3 \mathrm{p(x)dx} + \int_3^{\mathrm{x}} \mathrm{q(x)dx} = \frac{2}{3} + \int_3^{\mathrm{x}} \frac{1}{12}\mathrm{dx} = \frac{2}{3} + \frac{x}{6}\Big|_3^x = \frac{x+5}{12}$$

Evaluando F(7) = 1, entonces se sabe que $\frac{2}{3} \leq F(x) < 1$

Entonces ya se obtiene la función acumulativa expresada así:

$$F(x) = \begin{cases} \frac{x}{6}\ si\ 0 \leq x \leq 2 & 0 \leq F(x) < \frac{1}{3} \\ \frac{x-1}{3}\ si\ 2 \leq x \leq 3 & \frac{1}{3} \leq F(x) < \frac{2}{3} \\ \frac{x+5}{12}\ si\ 3 \leq x \leq 7 & \frac{2}{3} \leq F(x) < 1 \end{cases}$$

Se iguala F(x)=r (un número aleatorio entre 0 y 1)

1. Si $0 \leq r < \frac{1}{3}$ entonces $\frac{x}{6} = r$, luego $x = 6 * r$
2. Si $\frac{1}{3} \leq r < \frac{2}{3}$ entonces $\frac{x-1}{3} = r$ luego $x = 3 * r + 1$
3. Si $\frac{2}{3} \leq r < 1$ entonces $\frac{x+5}{2} = r$ luego $x = 12 * r - 5$

Volviendo al inicio, en la simulación se generó un número al azar que fue r=0.3, entonces es el caso 1, luego x=6*0.3=1.8 horas

Distribución Normal

La distribución normal es una de las más utilizadas porque ciertos fenómenos tienen un comportamiento que sigue esta distribución:

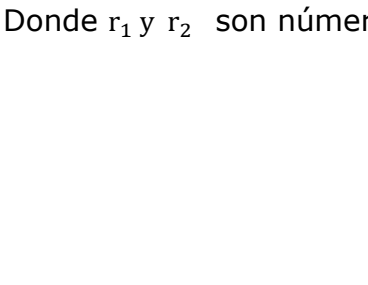

Figura 12: Distribución normal

Los parámetros recibidos para poder usar esta distribución son:

μ es la media y σ la desviación

Para generar la variable aleatoria que utilice estos parámetros se usa la ecuación:

$$X = \mu + \sigma * c$$

Donde c es un valor aleatorio que se deduce con la siguiente fórmula:

$$c = \cos(2\pi r_2) * \sqrt{-2 * In(r_1)}$$

Donde $\mathrm{r_1}$ y $\mathrm{r_2}$ son números aleatorios uniformemente distribuidos entre 0 y 1.

Un ejemplo de ejecución es:

Para un $\mu = 600$ que es la media y $\sigma = 80$ que es la desviación, este el resultado de la ejecución.

r_1	r_2	c	X
0,7426171	0,0689937	0,70010015	656,00801
0,7726637	0,7887022	0,17293273	613,83462
0,5599427	0,9631719	1,04825508	683,86041
0,4260246	0,4137198	-1,1190339	510,47729
0,5133995	0,2838823	-0,2439766	580,48187
0,0369807	0,3154899	-1,0271334	517,82933
0,2473558	0,0174322	1,66146658	732,91733

Si se lanza más números y se clasificaran cuántos caen entre determinados rangos (por ejemplo 600<=X<620, 620<=X<640, 640<=X<660), al graficar la frecuencia se obtendría la siguiente gráfica:

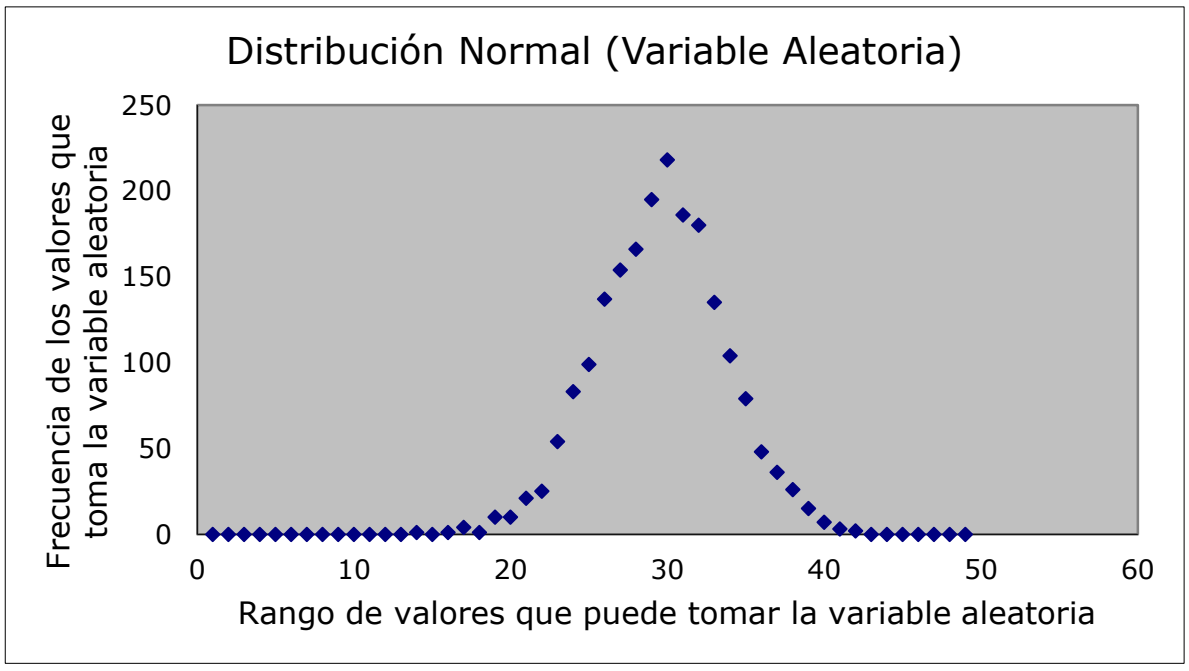

Figura 13: Distribución normal

Distribución Triangular

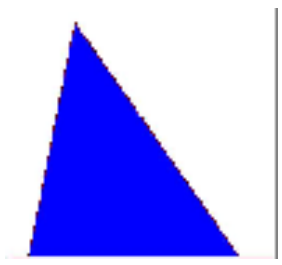

Figura 14: Distribución triangular

Los parámetros recibidos para poder usar esta distribución son:

Un valor mínimo (a), un valor más probable (b) y un valor máximo (c)

Para calcular el valor de la variable aleatorio está el siguiente algoritmo:

Si $r < \frac{b-a}{c-a}$ entonces

$$X = a + \sqrt{r * (c-a) * (b-a)}$$

De lo contrario

$$X = c - \sqrt{(1-r) * (c-a) * (c-b)}$$

Donde r es un número aleatorio.

Un ejemplo de ejecución

Mínimo: 5000

Más Probable: 7800

Máximo: 8000

r	condición	Valor 1	Valor 2	Valor variable aleatoria dependiendo de la condición
0,2534766	0,9333333	6459,1791	7330,7362	6459,1791
0,9251694		7787,7272	7788,1076	7787,7272
0,5270771		7104,1502	7467,3146	7104,1502
0,3167106		6631,0638	7359,7082	6631,0638
0,3338556		6674,6304	7367,7922	6674,6304
0,7954867		7584,9735	7649,703	7584,9735
0,7846921		7567,3749	7640,5772	7567,3749

Si se lanzase más números y se clasificaran cuántos caen entre determinados rangos (por ejemplo 7000<=X<7200, 7200<=X<7400, 7400<=X<7600), al graficar la frecuencia se obtiene la siguiente gráfica:

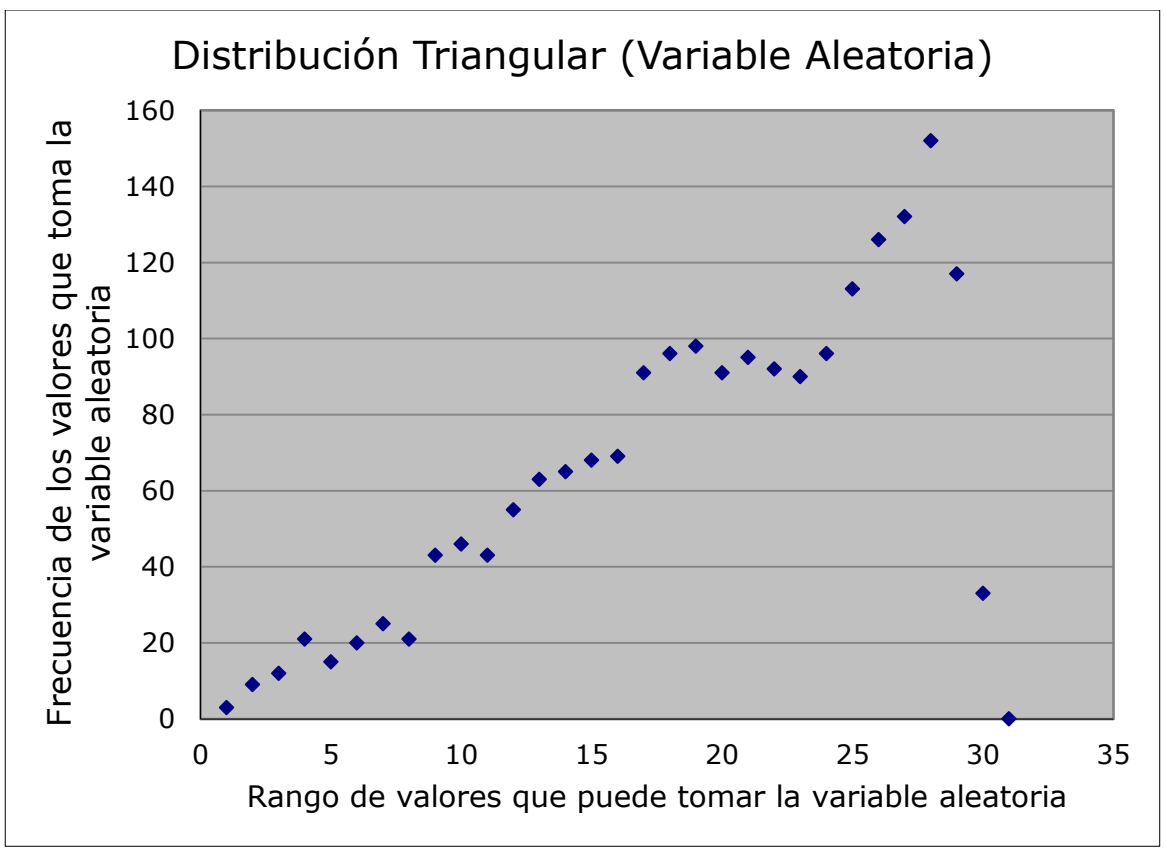

Figura 15: Distribución triangular

Distribución Exponencial

La función de distribución acumulada F(x) de la distribución exponencial está dada por:

$$F(x) = 1 - e^{-\alpha * x}$$

Como F(x) se iguala a un número aleatorio r, entonces se debe despejar X

$$F(x) = r$$

$$1 - e^{-\alpha * x} = r$$

$$x = -\frac{1}{\alpha} * In(1 - r)$$

Ejemplo

Dado un α =1.2 generar los diferentes valores

r	X
0,3941224	0,2505386
0,6148782	0,4770978
0,5109731	0,3576689
0,6400596	0,5109084
0,7265719	0,6483582
0,9674186	1,7120068
0,2039011	0,1140159

Si se lanzara más números y se clasificara cuántos caen entre determinados rangos (por ejemplo 0,1<=X<0,2; 0,2<=X<0,3; 0,3<=X<0,4), al graficar la frecuencia se obtiene la siguiente gráfica:

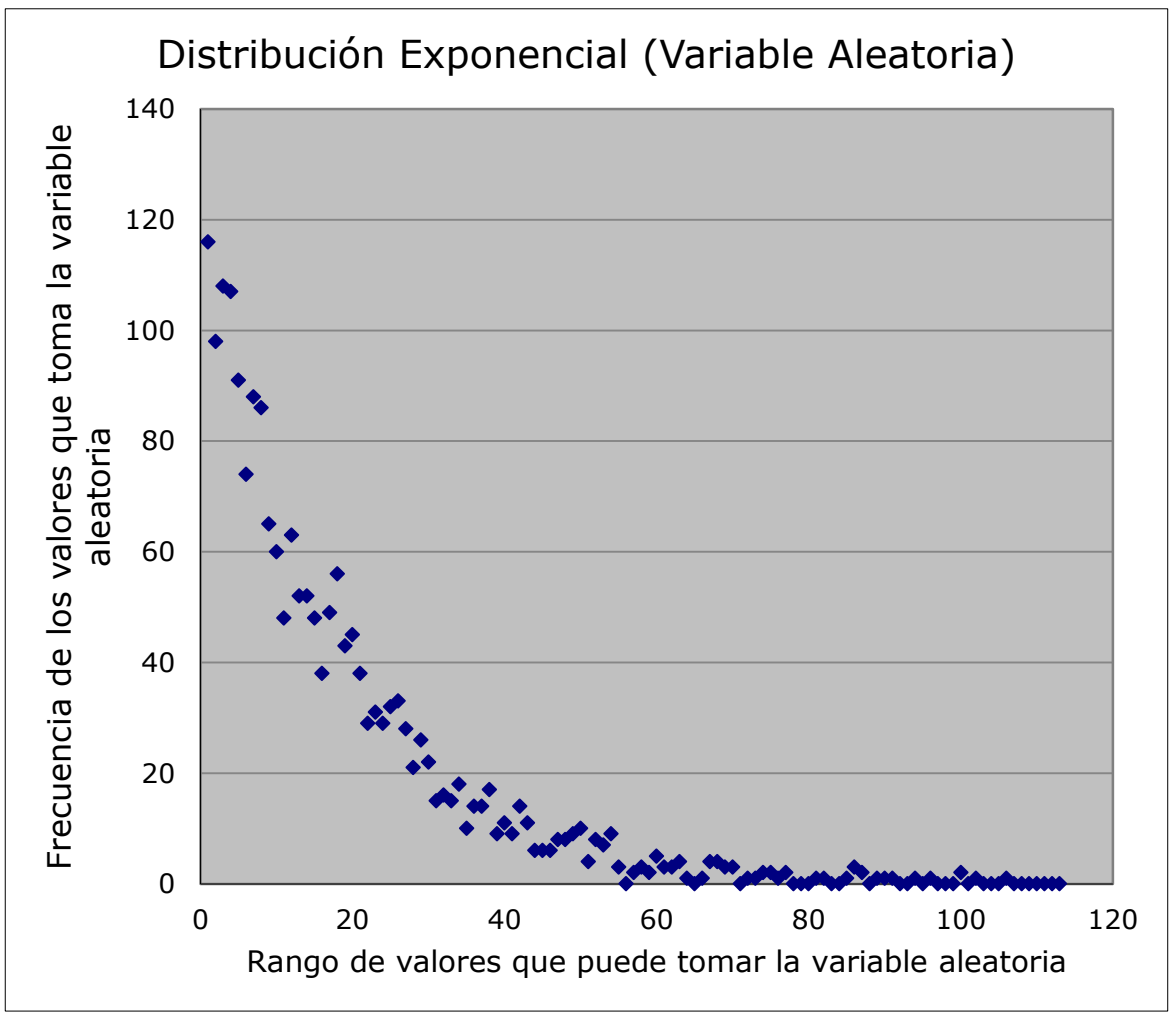

Figura 16: Distribución exponencial

Distribución Uniforme

Se requiere un valor aleatorio entre A y B, y que su ocurrencia sea uniforme entre esos dos valores. La función de distribución acumulada F(x) de la distribución uniforme está dada por:

$$F(x) = \frac{x - A}{B - A}$$

Como F(x) se iguala a un número aleatorio r, entonces se debe despejar X

$$F(x) = r$$

$$\frac{x - A}{B - A} = r$$

$$x = r * (B - A) + A$$

Ejemplo

Distribuir uniformemente entre 30 y 70. Luego A=30 y B=70

r	x
0.2145	38.58
0.1278	35.112
0.9121	66.484
0.5314	51.256

Distribuciones de variables aleatorias discretas

Distribución Bernoulli

La generación de datos con distribución Bernoulli es muy sencilla y rápida:

$$F(x)\begin{cases} éxito\ si\ r < p \\ fracaso\ en\ caso\ contrario \end{cases}$$

Ejemplo

Para p=0,6 entonces

r	F(x)
0.2145	Éxito
0.1278	Éxito
0.9121	Fracaso
0.5314	Éxito

Distribución Poisson

Su distribución de probabilidad está dada por:

$$f(k;\lambda) = \frac{e^{-\lambda} * \lambda^k}{k!}$$

e es el base del logaritmo natural (e = 2.71828...),

k! es el factorial de k,

λ es un número real positivo, representa cuantas veces ocurre el evento durante un intervalo dado.

Ejemplo

En un sistema de filas de espera se tiene que la media de llegada de clientes y la media de atención son:

Media de llegada de clientes = 3 por hora = 1 cada 20 minutos

Media de atención de clientes = 5 por hora = 1 cada 12 minutos

Como es necesario saber el número de eventos por hora (60 minutos) usando distribución Poisson entonces

$$\lambda = \frac{60}{20} = 3$$

Las probabilidades que se den 0, 1, 2, 3, 4, 5 llegadas en esa hora (es una distribución discreta) son:

$$P(0) = \frac{e^{-3} * 3^0}{0!} = 0{,}049787$$

$$P(1) = \frac{e^{-3} * 3^1}{1!} = 0{,}14936$$

$$P(2) = \frac{e^{-3} * 3^2}{2!} = 0{,}22404$$

Resultado	Probabilidad	Probabilidad Acumulada
0	0,04978706837	0,04978706837
1	0,1493612051	0,1991482735
2	0,2240418077	0,4231900812
3	0,2240418077	0,6472318889
4	0,1680313557	0,8152632446
5	0,1008188134	0,916082058
6	0,05040940672	0,9664914647
7	0,02160403145	0,9880954962
8	0,008101511795	0,996197008
9	0,002700503932	0,9988975119
10	0,0008101511795	0,9997076631
11	0,0002209503217	0,9997297581
12	0,0000552375804	0,9997849957
13	0,0000127471339	0,9997977428

Ejemplos de simulaciones para resolver problemas

Existen unos problemas cuya solución determinística es muy costosa porque requiere una gigantesca cantidad de cálculos que aún hechos estos por veloces computadoras tardarían siglos en resolverlos.

El problema del viajero

Dado un conjunto finito de ciudades, donde hay un costo para ir de una ciudad a otra, hallar la secuencia de visita de todas esas ciudades, de tal manera que solo se visite una sola vez cada ciudad, al menor costo posible.

Para comprender mejor este problema, se tiene la siguiente matriz que señala el costo de ir de una ciudad a otra.

Ciudad	A	B	C	D	E	F	G	H	I	J	K	L	M
A	**0**	6	34	65	11	13	20	19	61	41	60	47	31
B	6	**0**	21	8	51	6	30	23	24	34	36	12	63
C	34	21	**0**	41	21	9	27	41	10	22	50	25	61
D	65	8	41	**0**	18	63	63	19	56	6	37	6	50
E	11	51	21	18	**0**	12	46	61	20	17	40	37	61
F	13	6	9	63	12	**0**	53	53	61	53	14	57	46
G	20	30	27	63	46	53	**0**	41	19	31	29	16	53
H	19	23	41	19	61	53	41	**0**	7	61	43	31	56
I	61	24	10	56	20	61	19	7	**0**	20	5	34	47
J	41	34	22	6	17	53	31	61	20	**0**	52	63	45
K	27	16	31	10	63	62	20	27	18	51	**0**	46	16
L	37	51	10	31	6	61	7	62	52	7	46	**0**	21
M	5	40	48	11	59	61	52	18	49	18	16	21	**0**

Tabla 6: Costo de viajar de una ciudad a otra

Lo que quiere decir esa matriz es que ir de la ciudad C a la F tiene un costo de $9, ir de la ciudad H a la J tiene un costo de $20.

¿Qué ruta se debe hacer visitando solo una vez cada ciudad para minimizar el costo? Por ejemplo la siguiente ruta:

E-A-D-C-J-G-B-I-F-H-M-L-K

Para calcular su costo se realiza la siguiente tabla

Ruta desglosada	Valor
E-A	$ 11
A-D	$ 65
D-C	$ 41
C-J	$ 22
J-G	$ 31
G-B	$ 30
B-I	$ 24
I-F	$ 61
F-H	$ 53
H-M	$ 56
M-L	$ 21
L-K	$ 46
Total	$ 461

Tabla 7: Costo de la ruta

Significa que esa ruta en total costaría $461, ¿habrá una ruta menos costosa? Se genera la siguiente ruta:

A-B-F-C-I-H-D-E-J-G-L-M-K

Ruta desglosada	Valor
A-B	$ 6
B-F	$ 6
F-C	$ 9
C-I	$ 10
I-H	$ 7
H-D	$ 19
D-E	$ 18
E-J	$ 17
J-G	$ 31
G-L	$ 16
L-M	$ 21

M-K	$ 16
Total	$ 176

Tabla 8: Costo de la ruta

Con esta nueva ruta propuesta el costo se reduce y ahora es de $176, ¿habrá una ruta menos costosa aún?

El problema es que se deberían evaluar todas las rutas posibles, y el total de rutas es el factorial del número de ciudades.

$$Total\ Rutas = 13 * 12 * 11 * 10 * 9 * 8 * 7 * 6 * 5 * 4 * 3 * 2 * 1 = 6.227.020.800$$

Más de 6 mil millones de rutas posibles. Si un computador costeara cada ruta por milésima de segundo, tardaría aproximadamente 72 días en probar todas las rutas.

Si no fueran 13 ciudades, sino 30 ciudades, así se tuviese un computador que evaluara mil millones de rutas por segundo, se tardaría $8 * 10^{15}$ años en evaluar todas las rutas para dar con la menos costosa.

Ante ese tipo de problemas existe otra forma de resolverlos y es usando el azar como método de resolución. Este es el procedimiento:

Paso 1: Se genera una ruta completamente al azar y se evalúa su costo.

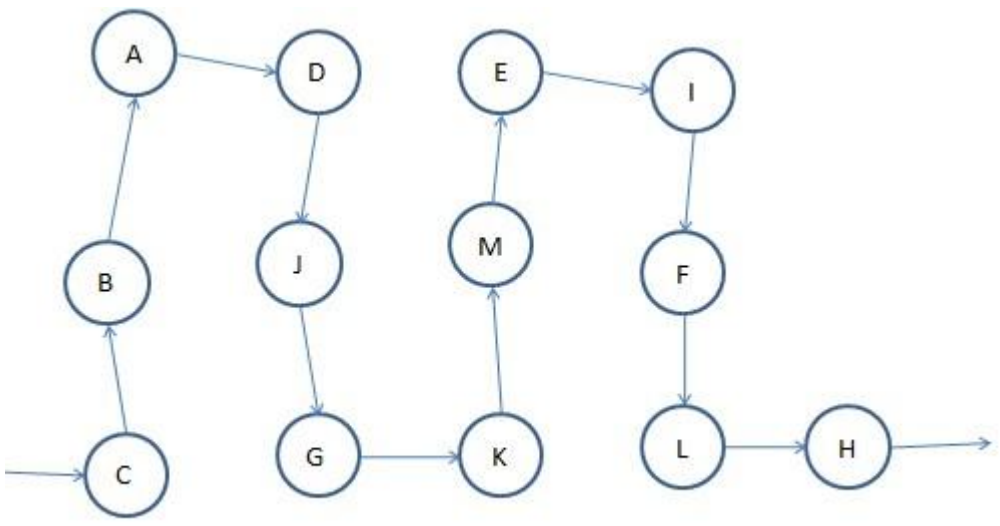

Figura 17: Se genera una ruta al azar

Paso 2: Se hace un cambio aleatorio a esa ruta y se vuelve a ensayar, si es mejor entonces esa nueva ruta se selecciona, si no, se deja la ruta anterior. En el ejemplo se intercambia las ciudades B y J, esta es la nueva ruta.

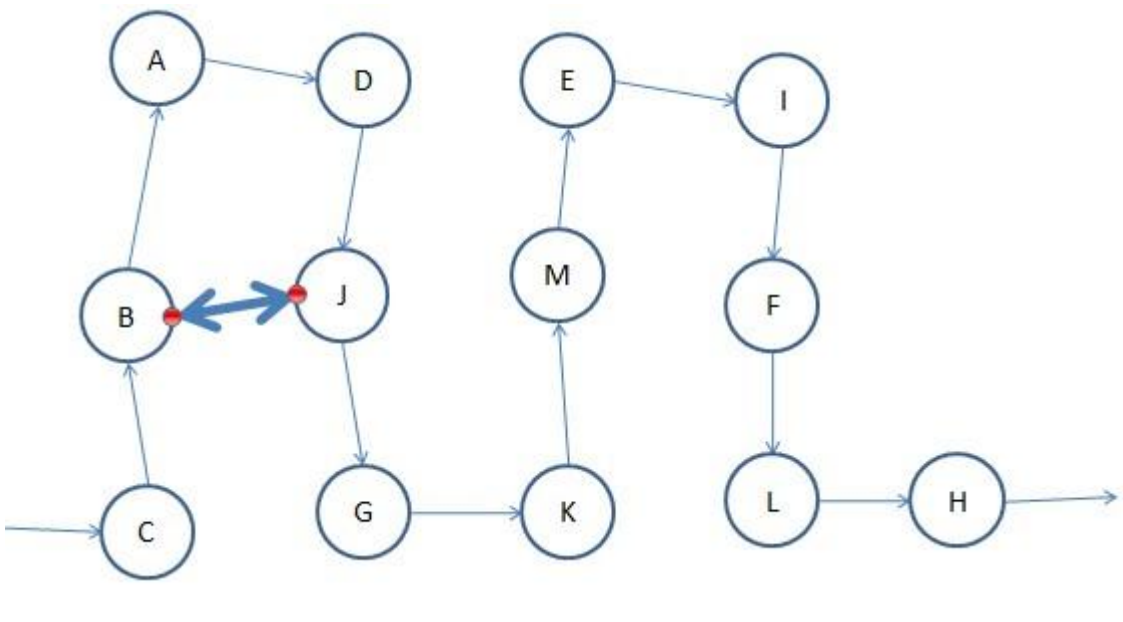

Figura 18: Se selecciona dos ciudades al azar y se hace el intercambio

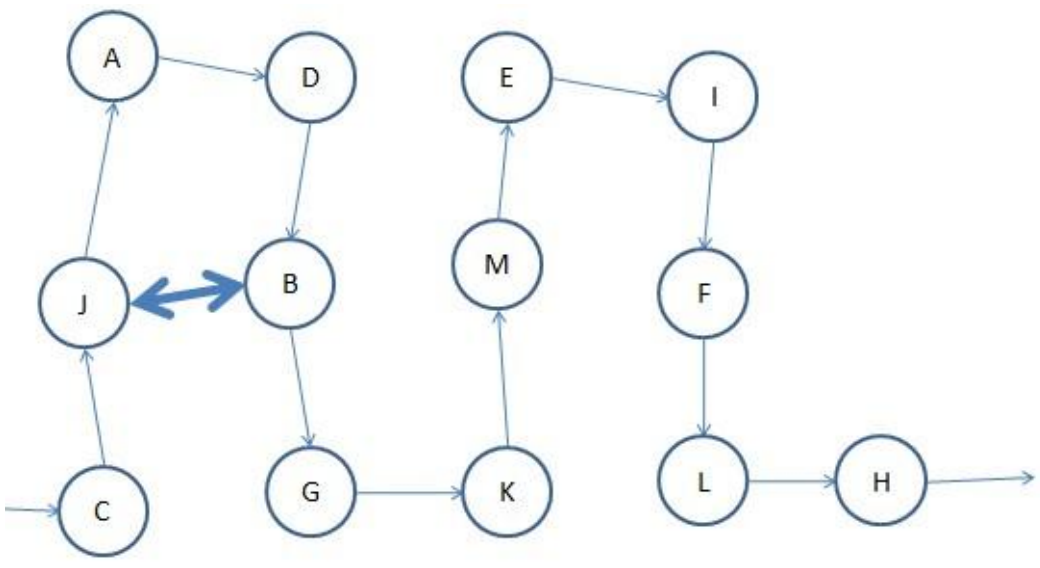

Figura 19: Nueva ruta que será evaluada

Paso 3: Se ejecuta el paso anterior miles de veces. Se obtiene una buena ruta con un costo bajo.

Implementando en Java este algoritmo:

```
import java.util.Random;

public class BuscaRuta
{
  public static void main(String[] args)
  {
   /* Total de ciudades */
   final int CIUDADES = 13;

   /* Llena la matriz de costos */
   int costo[][]=
   { { 0,  6, 34, 65, 11, 13, 20, 19, 61, 41, 60, 47, 31},
     { 6,  0, 21,  8, 51,  6, 30, 23, 24, 34, 36, 12, 63},
     {34, 21,  0, 41, 21,  9, 27, 41, 10, 22, 50, 25, 61},
     {65,  8, 41,  0, 18, 63, 63, 19, 56,  6, 37,  6, 50},
     {11, 51, 21, 18,  0, 12, 46, 61, 20, 17, 40, 37, 61},
     {13,  6,  9, 63, 12,  0, 53, 53, 61, 53, 14, 57, 46},
     {20, 30, 27, 63, 46, 53,  0, 41, 19, 31, 29, 16, 53},
     {19, 23, 41, 19, 61, 53, 41,  0,  7, 61, 43, 31, 56},
     {61, 24, 10, 56, 20, 61, 19,  7,  0, 20,  5, 34, 47},
     {41, 34, 22,  6, 17, 53, 31, 61, 20,  0, 52, 63, 45},
     {27, 16, 31, 10, 63, 62, 20, 27, 18, 51,  0, 46, 16},
     {37, 51, 10, 31,  6, 61,  7, 62, 52,  7, 46,  0, 21},
     { 5, 40, 48, 11, 59, 61, 52, 18, 49, 18, 16, 21,  0}
   };

   /* Define el arreglo unidimensional que tiene la ruta */
   int ruta[] = new int[CIUDADES];

   /* Generador de números pseudo-aleatorios */
   Random azar = new Random();

   /* Genera una ruta al azar */
   int ciudad;
   for (int cont=0; cont<CIUDADES; cont++)
   {
     boolean continuar = true;
     do
     {
       continuar = false;
       ciudad = azar.nextInt(CIUDADES);
       for(int valida=0; valida<cont; valida++)
         if (ruta[valida]==ciudad) continuar = true;
     }while(continuar);
     ruta[cont]=ciudad;
   }

   /* Hace el proceso de intercambio 100.000 veces */
   for (int pruebas=1; pruebas<=100000; pruebas++)
   {
     /* Calcula el costo de la ruta */
     int costoAntes=0;
     for (int genera=0; genera<CIUDADES-1; genera++)
     {
       int origen = ruta[genera];
       int destino = ruta[genera+1];
       costoAntes += costo[origen][destino];
     }

     /* Hace un intercambio al azar entre dos ciudades */
     int sitioInicial = azar.nextInt(CIUDADES);
     int sitioFinal = azar.nextInt(CIUDADES);

     int temporal = ruta[sitioInicial];
     ruta[sitioInicial] = ruta[sitioFinal];
     ruta[sitioFinal] = temporal;

     /* Calcula el nuevo costo con la nueva ruta */
     int costoDespues=0;
     for (int genera=0; genera<CIUDADES-1; genera++)
     {
       int origen = ruta[genera];
       int destino = ruta[genera+1];
       costoDespues += costo[origen][destino];
```

```
      }

      /* Si el intercambio disminuye el costo entonces
         imprime la nueva ruta */
      if (costoDespues < costoAntes)
      {
        for(int cont=0; cont<CIUDADES; cont++)
          System.out.print(ruta[cont] + "-");
        System.out.println(" Costo: " + costoDespues);
      }
      else /* Deja como estaba la ruta anteriormente */
      {
        temporal = ruta[sitioInicial];
        ruta[sitioInicial] = ruta[sitioFinal];
        ruta[sitioFinal] = temporal;
      }
    }
  }
}
```

Se muestra a continuación como sería una ejecución del programa

```
8-7-9-5-3-11-6-0-4-2-10-1-12- Costo: 378
8-7-9-5-3-11-6-0-4-2-10-12-1- Costo: 355
12-7-9-5-3-11-6-0-4-2-10-8-1- Costo: 352
12-7-10-5-3-11-6-0-4-2-9-8-1- Costo: 317
12-7-5-10-3-11-6-0-4-2-9-8-1- Costo: 226
12-7-5-10-3-11-6-0-4-9-2-8-1- Costo: 212
12-7-5-10-3-11-6-0-4-9-8-2-1- Costo: 207
12-7-1-10-3-11-6-0-4-9-8-2-5- Costo: 187
12-7-8-10-3-11-6-0-4-9-1-2-5- Costo: 165
12-7-8-10-3-11-6-9-4-0-1-2-5- Costo: 148
12-7-8-10-3-9-6-11-4-0-1-2-5- Costo: 146
12-7-8-10-3-9-6-11-4-0-1-5-2- Costo: 131
```

Traduciendo la ruta (A=0, B=1, C=2, …M=12), entonces la ruta hallada por el software es:
M-H-I-K-D-J-G-L-E-A-B-F-C con un costo de $131

Hallando el área bajo la curva

Otro problema que puede ser abordado con un proceso en el que el azar está involucrado es el cálculo del área que se encuentra bajo la curva de una ecuación. Se observa la siguiente gráfica generada por la ecuación $Y = X * seno(X)$:

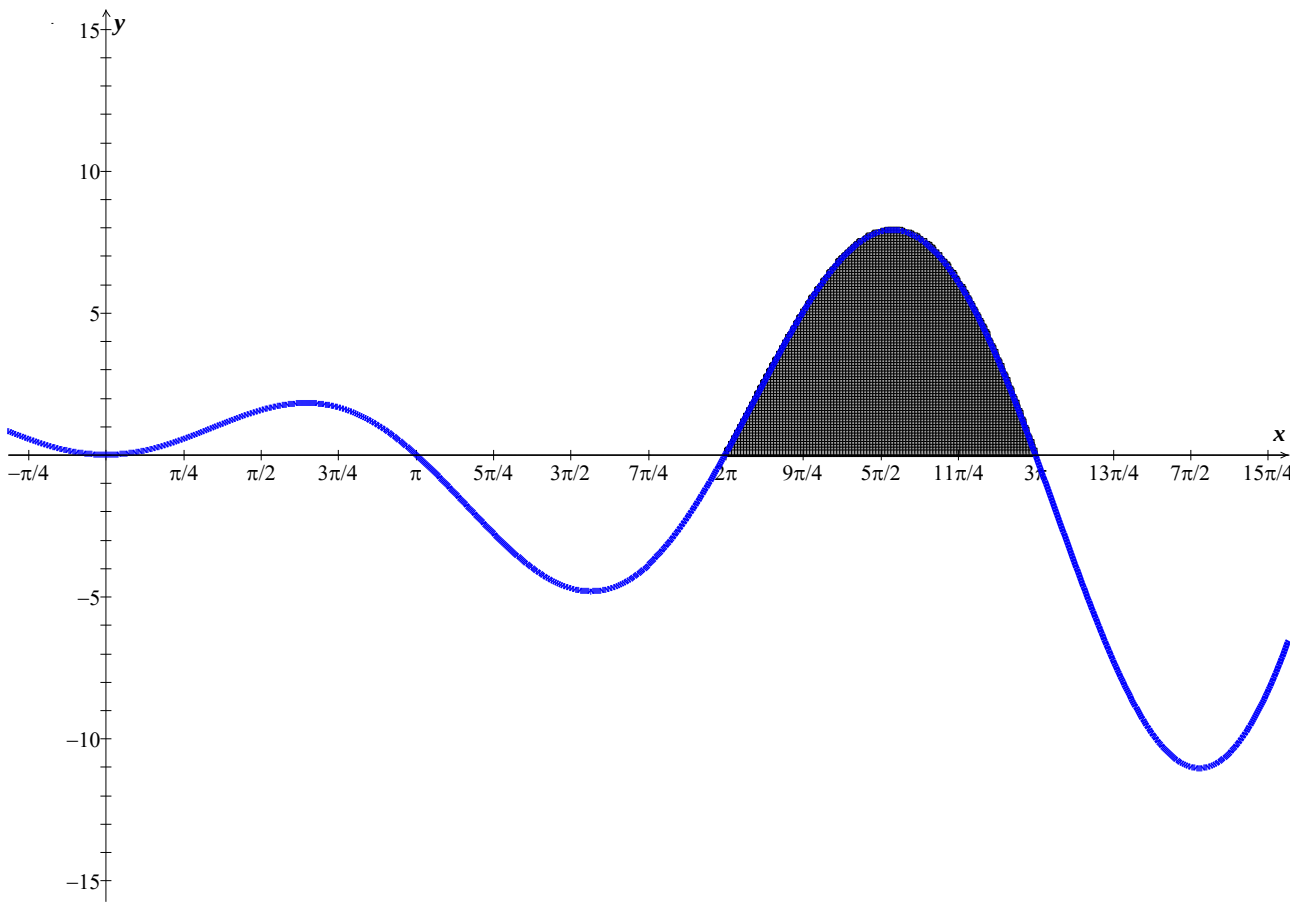

Figura 20: Hallando el área bajo la curva

Se solicita que se encuentre el área que encierra la curva generada por la ecuación: $Y = X * seno(X)$ entre 2π y 3π. En la gráfica aparece esa área sombreada.

Matemáticamente se puede hallar está área resolviendo la siguiente integral:

$$Area = \int_{2\pi}^{3\pi} X * seno(X) dX = 5\pi = 15.70796\ldots$$

El procedimiento de cálculo requiere saber hacer integrales, pero si no se sabe o recuerda el procedimiento o este es muy complejo de realizar, un método para hallar el área bajo la curva es el siguiente:

Paso 1: Dibujar un rectángulo que cubra el área a hallar, la longitud de la base y la altura debe ser conocida.

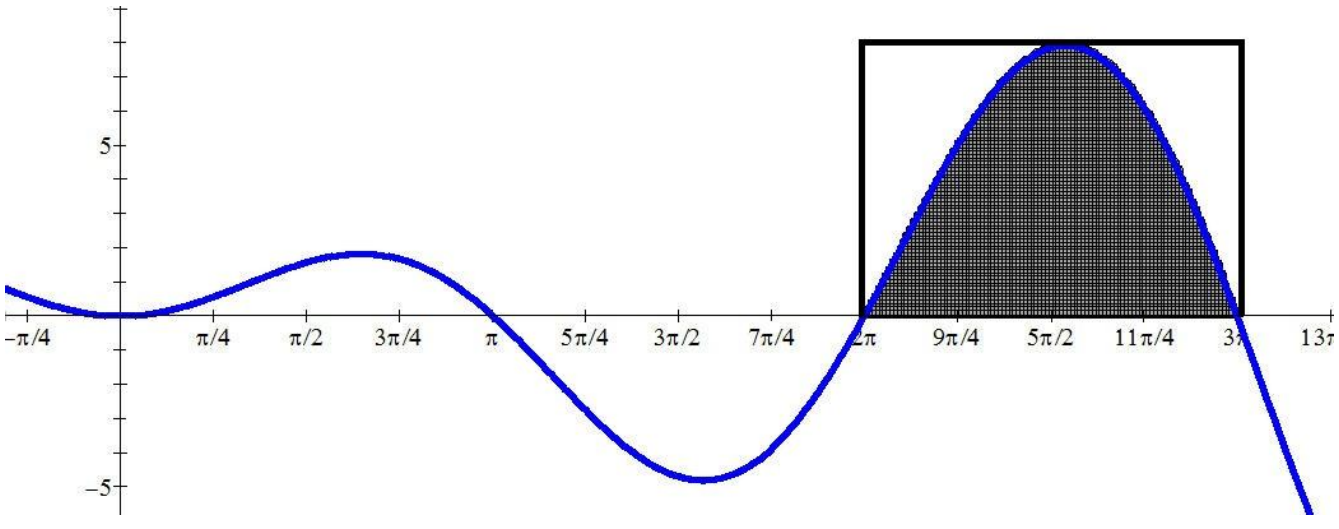

Figura 21: Se dibuja un rectángulo que contenga el área a hallar

La longitud de la base del rectángulo en el ejemplo ya es conocida:

$$Longitud = 3\pi - 2\pi = \pi$$

La altura del rectángulo también debe ser conocida. Una forma de deducir la altura es ver que el máximo de la curva entre 2π y 3π es cuando X=5π/2, allí el valor de Y=5π/2*seno(5π/2)= 5π/2*1=7.853981…

Luego el rectángulo tiene base= π y altura=5 π/2

Paso 2: Se lanza al azar un número finito de puntos dentro del rectángulo.

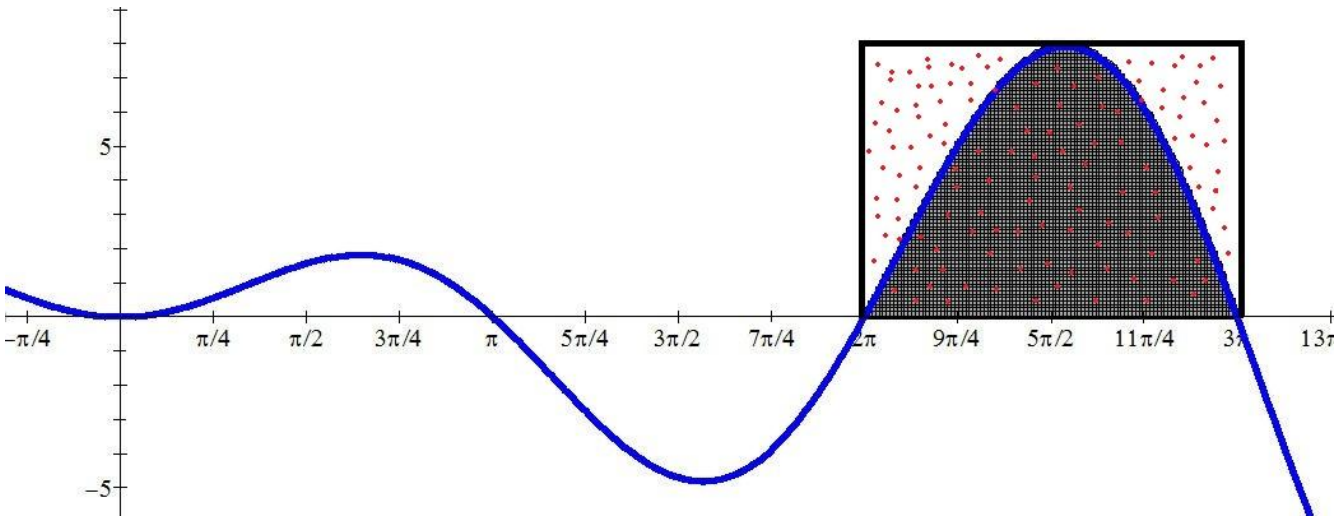

Figura 22: Se lanza un número alto de puntos al azar dentro del rectángulo

Paso 3: Se cuenta el número de puntos que caen dentro del área a hallar.

Paso 4: Se aplica la siguiente fórmula para hallar el área bajo la curva

$$Área = Área\ del\ rectángulo * \frac{Número\ de\ puntos\ que\ caen\ dentro\ del\ área}{Total\ de\ puntos\ lanzados\ al\ azar}$$

El siguiente programa en Java encuentra el área de la ecuación del ejemplo siguiendo el procedimiento.

```
import java.util.Random;

public class AreausandoAzar
{
 public static void main(String[] args)
 {
  /* Generador de números pseudo-aleatorios */
  Random azar = new Random();

  /* Limites en X del área a hallar.
     Sirven también para calcular la base
     del rectángulo. */
  double xmin = 2*Math.PI;
  double xmax = 3*Math.PI;

  /* Altura máxima del rectángulo */
  double ymax = 5*Math.PI/2;

  /* Variable que lleva el total de puntos que
```

```
   caen dentro del área */
int puntosArea = 0;

/* Total de puntos a ser lanzados */
int puntosTotal = 1000000;

/* Lanza puntos al azar dentro del rectángulo */
for (int puntos=1; puntos<=puntosTotal; puntos++)
{
 /* Una posicion X y Y al azar por cada punto */
 double posX = azar.nextDouble()*(xmax-xmin)+xmin;
 double posY = azar.nextDouble()*(ymax-0)+0;

 /* Determina si ese punto cae dentro del área */
 if (posY <= posX*Math.sin(posX)) puntosArea++;
}

/* Calcula el área del rectángulo */
double areaRect = (xmax-xmin)* ymax;

/* Calcula el área bajo la curva */
double area;
area =  areaRect * (double) puntosArea / puntosTotal;

/* Imprime el resultado */
System.out.println("área hallada es: " +  area);

 }
}
```

El resultado ofrecido por el software es:

Área hallada es: 15.701331575594034

Una precisión muy buena al compararla con el resultado real que es 5π = 15.70796…

Resolviendo un Sudoku

El popular pasatiempo llamado Sudoku puede ser resuelto usando técnicas en las que el azar es protagonista.

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

Figura 23: Ejemplo de un Sudoku

El algoritmo para resolverlo, consiste en lanzar números al azar en las casillas vacías siempre y cuando se cumplan las reglas del juego. Sucede que a medida que se llenan las casillas con números, llega un momento en el cual no hay forma de continuar (uno o varios números que en su momento cumplían las reglas no están en la posición correcta), en ese caso, el algoritmo borra al azar algunos números previamente puestos.

En ese proceso de: poner números al azar en casillas vacías cumpliendo reglas del juego + borrar al azar algunos números puestos, con el tiempo, el Sudoku es resuelto.

El algoritmo implementado en Java es el siguiente:

```
import java.util.Random;

public class ResuelveSudoku
{
  public static void main(String[] args)
```

```
{
 /* El Sudoku como es planteado.
  Los ceros(0) son las casillas vacías a completar */
 int sudoku[][]=
 { { 5, 3, 0, 0, 7, 0, 0, 0, 0},
   { 6, 0, 0, 1, 9, 5, 0, 0, 0},
   { 0, 9, 8, 0, 0, 0, 0, 6, 0},
   { 8, 0, 0, 0, 6, 0, 0, 0, 3},
   { 4, 0, 0, 8, 0, 3, 0, 0, 1},
   { 7, 0, 0, 0, 2, 0, 0, 0, 6},
   { 0, 6, 0, 0, 0, 0, 2, 8, 0},
   { 0, 0, 0, 4, 1, 9, 0, 0, 5},
   { 0, 0, 0, 0, 8, 0, 0, 7, 9}
 };


 /* Objeto generador de números pseudo-aleatorios */
 Random azar = new Random();

 /* Mantiene el ciclo hasta que resuelva el Sudoku */
 boolean finalizo=false;

 /* Lleva el número de iteraciones */
 int ciclos=0;

 /* Tablero en el que se trabaja */
 int solucion[][] = new int[9][9];

 /* Cada cuantos ciclos borra números para destrabar */
 final int DESTRUYE = 700;

 /* Ciclo que llenará el sudoku completamente */
 do
 {
  /* Se copia el sudoku sobre el tablero a evaluar */
  for (int fila=0; fila<9; fila++)
   for (int columna=0; columna<9; columna++)
    if (sudoku[fila][columna] != 0)
     solucion[fila][columna] = sudoku[fila][columna];

  /* Busca un número al azar para
     colocar en alguna celda */
  boolean numValido=true;
  do
  {
   /* Una posición X de 0 a 8 */
   int posX = azar.nextInt(9);

   /* Una posición Y de 0 a 8 */
   int posY = azar.nextInt(9);

   /* Un número al azar de 1 a 9 */
   int numero = azar.nextInt(9) + 1;

   /* Chequea si el número no se repite
    ni vertical ni horizontalmente */
   numValido=true;
   for (int cont=0; cont<9; cont++)
    if (solucion[cont][posY]==numero || solucion[posX][cont]==numero)
      numValido=false;

   /* Si el número no se repite entonces
      lo coloca en el tablero */
   if (numValido) solucion[posX][posY]=numero;

 }while(!numValido);


 /* Chequea que NO se viole la regla de que cada uno
  de los 9 cuadros internos no repita número */
 for (int cuadroX=0; cuadroX<=6; cuadroX+=3)
  for (int cuadroY=0; cuadroY<=6; cuadroY+=3)
  {
   int numRepite=0, valor;
   for (valor=1; valor<=9; valor++)
   {
    numRepite=0;
    for (int posX=0; posX<3; posX++)
```

```
      for (int posY=0; posY<3; posY++)
       if (solucion[cuadroX+posX][cuadroY+posY]==valor)
        numRepite++;
     if (numRepite>1) break;
    }

    /* Si detecta repetición, entonces
     borra todos los números repetidos */
    if (numRepite>1)
     for (int posX=0; posX<3; posX++)
      for (int posY=0; posY<3; posY++)
       if (solucion[cuadroX+posX][cuadroY+posY]==valor)
        solucion[cuadroX+posX][cuadroY+posY]=0;
  }

  /* Chequea si se completó el sudoku completamente */
  finalizo=true;
  for (int posX=0; posX<9; posX++)
   for (int posY=0; posY<9; posY++)
    if (solucion[posX][posY]==0)
    finalizo=false;

  /* Cada ciertos ciclos para destrabar,
   borra la tercera parte de lo completado */
  ciclos++;
  if (ciclos%DESTRUYE==0)
  {
   System.out.println("Ciclos: " + ciclos);
   for (int posX=0; posX<9; posX++)
    for (int posY=0; posY<9; posY++)
     if (azar.nextInt()%3==0)
      solucion[posX][posY]=0;
  }
 }while (!finalizo);

 /* Imprime el sudoku terminado */
 System.out.println("Ciclos totales: " + ciclos);
 for (int posX=0; posX<9; posX++)
 {
  for (int posY=0; posY<9; posY++)
   System.out.print(solucion[posX][posY] + " ");
  System.out.println(" ");
 }
 }
}
```

La ejecución del programa es así:

```
Ciclos: 700
Ciclos: 1400
Ciclos: 2100
Ciclos: 2800
Ciclos: 3500
Ciclos: 4200
Ciclos totales: 4244

5 3 4 6 7 8 9 1 2
6 7 2 1 9 5 3 4 8
1 9 8 3 4 2 5 6 7
8 5 9 7 6 1 4 2 3
4 2 6 8 5 3 7 9 1
7 1 3 9 2 4 8 5 6
9 6 1 5 3 7 2 8 4
2 8 7 4 1 9 6 3 5
3 4 5 2 8 6 1 7 9
```

Para ver la potencia de este programa, se prueba con el sudoku más difícil del mundo [7], que es el siguiente:

1					7		9	
	3			2				8
		9	6			5		
		5	3			9		
	1			8				2
6					4			
3							1	
	4							7
		7				3		

Figura 24: El sudoku más difícil del mundo

Ese sudoku llamado por su autor AI Escargot, fue inventado por Arto Inkala, un matemático finlandés experto en matemática aplicada.

Ejecutando el programa, este arroja la respuesta después de poco más de 3 millones de iteraciones:

```
  /* El Sudoku como es planteado.
   Los ceros(0) son las casillas vacías a completar */
  int sudoku[][]=
  { { 1, 0, 0, 0, 0, 7, 0, 9, 0},
    { 0, 3, 0, 0, 2, 0, 0, 0, 8},
    { 0, 0, 9, 6, 0, 0, 5, 0, 0},
    { 0, 0, 5, 3, 0, 0, 9, 0, 0},
    { 0, 1, 0, 0, 8, 0, 0, 0, 2},
    { 6, 0, 0, 0, 0, 4, 0, 0, 0},
    { 3, 0, 0, 0, 0, 0, 0, 1, 0},
    { 0, 4, 0, 0, 0, 0, 0, 0, 7},
    { 0, 0, 7, 0, 0, 0, 3, 0, 0}
  };
```

Ciclos totales: 3015795

```
1 6 2 8 5 7 4 9 3
5 3 4 1 2 9 6 7 8
7 8 9 6 4 3 5 2 1
4 7 5 3 1 2 9 8 6
9 1 3 5 8 6 7 4 2
6 2 8 7 9 4 1 3 5
3 5 6 4 7 8 2 1 9
2 4 1 9 3 5 8 6 7
8 9 7 2 6 1 3 5 4
```

Resolviendo el problema de Monty Hall o de las tres cajas

Este problema pone a prueba la intuición contra la probabilidad. ¿En qué consiste? En un concurso, al jugador se le presentan tres cajas cerradas, una de estas tiene un premio, las otras dos están vacías. El jugador no sabe en qué caja se encuentra el premio así que decide escoger una al azar. El dueño del concurso que si sabe en qué caja está el premio, hace la siguiente maniobra: de las dos cajas restantes, destapa una de ellas y muestra que está vacía. Luego ofrece la oportunidad al concursante de cambiar su elección inicial o seguir con la caja que había escogido al inicio. La decisión es del jugador. ¿Debe cambiar de elección? ¿Debe seguir con la elección inicial? ¿Es irrelevante el cambiar la elección?

Si la respuesta es decir que es irrelevante cambiar o no de caja, es un error. Una rápida consulta en Internet muestra que la mejor decisión es cambiar la elección inicial, es decir, tomar la otra caja que no destapó el dueño. Porque las probabilidades de éxito aumentan el doble que si se mantiene tercamente con la primera caja escogida.

El siguiente programa en Java, simula esta situación varias veces, con un concursante no cambiando su decisión y con otro concursante si cambiando su decisión. Los resultados son concluyentes.

```java
import java.util.Random;

public class Probabilidad
{
 public static void main(String[] args)
 {
  /* Generador de números pseudo-aleatorios */
  Random azar = new Random();

  /* Contador de éxitos si
     el jugador cambia su decisión inicial */
  int cambia = 0;

  /* Contador de éxitos si el jugador
     insiste en mantener su decisión inicial */
  int insiste = 0;

  for (int prueba=1; prueba<=1000000; prueba++)
  {
   /* Selecciona una caja al azar que será la ganadora */
   int cajaGana = azar.nextInt(3);

   /* El jugador selecciona una caja al azar */
   int cajaJuega = azar.nextInt(3);

   /* Si el jugador escogió por casualidad la caja ganadora,
      el dueño escoge al azar alguna de las dos cajas
      restantes */
   int cajaDueno = -1;
   if (cajaGana == cajaJuega)
    do
    {
     cajaDueno = azar.nextInt(3);
    }while(cajaDueno == cajaGana);
   else
    /* El dueño sólo puede escoger
       la caja que no tiene premio */
    do
    {
     cajaDueno = azar.nextInt(3);
    }while(cajaDueno == cajaGana || cajaDueno == cajaJuega);

   /* El jugador NO cambia su elección */
   if (cajaGana == cajaJuega) insiste++;

   /* El jugador SI cambia su elección */
   int nuevaCaja = -1;
   do
   {
    nuevaCaja = azar.nextInt(3);
   }while(nuevaCaja == cajaDueno || nuevaCaja == cajaJuega);
   if (nuevaCaja == cajaGana) cambia++;
  }

  System.out.println("Número de aciertos");
  System.out.println("SIN cambiar la elección inicial: " + insiste);
  System.out.println("CAMBIANDO la elección inicial: " + cambia);
 }

}
```

La ejecución del programa genera la siguiente salida

Número de aciertos

SIN cambiar la elección inicial: 333721

CAMBIANDO la elección inicial: 666279

Simulando se llega a la misma conclusión estadística, es mejor cambiar la elección que mantenerse tercamente con la elección inicial.

Deducir la mejor estrategia para que dos robots se encuentren

Dos robots sin sensores de luz ni sonido están ubicados sobre una superficie cuadriculada que emula a un arreglo bidimensional. ¿Cómo se logra más rápido el encuentro si el desplazamiento de cada robot es aleatorio? ¿Uno de estos quieto, mientras el otro se mueve? ¿Los dos moviéndose?

Dado un arreglo bidimensional vacío de N*M, se ubica en alguna celda, un robot y en otra celda el otro robot.

En cada ciclo, el robot que se mueve puede desplazarse a una celda vacía cualquiera alrededor siempre y cuando no se salga del tablero.

El encuentro es cuando ambos robots coinciden en la misma celda.

Se simula varias veces los dos escenarios: 1. Un robot se mueve, el otro quieto; 2. Los dos robots moviéndose.

R1							
				R2			

Cada robot tiene los siguientes movimientos

Caso 1	Caso 2	Caso 3
Caso 4	**Robot**	Caso 5
Caso 6	Caso 7	Caso 8

El robot puede desplazarse a cualquier casilla vecina por iteración, es el mismo tipo de movimiento del Rey en el Ajedrez.

El primer escenario en el que un robot se mueve y el otro permanece quieto, se usa el siguiente código en Java

```
import java.util.Random;

public class EscenarioA
{
 public static void main(String[] args)
 {
  int limiteX = 50;
  int limiteY = 50;

  /* Ubica al azar robotA y robotB, valida que no
   * queden en la misma celda.
   */
Random azar = new Random();
int robotA_X, robotB_X, robotA_Y, robotB_Y;

do
{
 robotA_X = azar.nextInt(limiteX);
 robotA_Y = azar.nextInt(limiteY);
 robotB_X = azar.nextInt(limiteX);
 robotB_Y = azar.nextInt(limiteY);
}while (robotA_X==robotB_X && robotA_Y==robotB_Y);

System.out.println("Ubicación inicial: robotA(" + robotA_X + "," + robotA_Y + ")");
System.out.println("Ubicación inicial: robotB(" + robotB_X + "," + robotB_Y + ")");

/* robotA se mueve, robotB quieto */
int numMov = 0;
do
{
```

```
numMov++;
boolean movio = false;
do
{
 /* Movimiento al azar del robotA */
 switch(azar.nextInt(8))
 {
  case 0: if (robotA_X>0 && robotA_Y>0)
    { robotA_X--; robotA_Y--; movio=true; } break;
  case 1: if (robotA_Y>0)
    { robotA_Y--; movio=true; } break;
  case 2: if (robotA_X<limiteX && robotA_Y>0)
    { robotA_X++; robotA_Y--; movio=true; } break;
  case 3: if (robotA_X>0)
    { robotA_X--; movio=true; } break;
  case 4: if (robotA_X<limiteX)
    { robotA_X++; movio=true; } break;
  case 5: if (robotA_X>0 && robotA_Y<limiteY )
    { robotA_X--; robotA_Y++; movio=true; } break;
  case 6: if (robotA_Y<limiteY)
    { robotA_Y++; movio=true; } break;
  default: if (robotA_X<limiteX && robotA_Y<limiteX)
     { robotA_X++; robotA_Y++; movio=true; } break;
    }
   }while(movio==false);
  }while (robotA_X!=robotB_X || robotA_Y!=robotB_Y);
  System.out.println("# movimientos para encuentro: " + numMov);
 }
}
```

Un ejemplo de ejecución es el siguiente

Ubicación inicial: robotA(10,27)
Ubicación inicial: robotB(2,13)
movimientos para encuentro: 478

Para el segundo escenario, en el cual los dos robots se mueven, se usa este código en Java

```
import java.util.Random;

public class Escenario2
{
 public static void main(String[] args)
 {
  int limiteX = 50;
  int limiteY = 50;

  /* Ubica al azar robotA y robotB, valida que no
   * queden en la misma celda.
   */
Random azar = new Random();
int robotA_X, robotB_X, robotA_Y, robotB_Y;

do
{
 robotA_X = azar.nextInt(limiteX);
 robotA_Y = azar.nextInt(limiteY);
 robotB_X = azar.nextInt(limiteX);
 robotB_Y = azar.nextInt(limiteY);
}while (robotA_X==robotB_X && robotA_Y==robotB_Y);

System.out.println("Ubicación inicial: robotA(" + robotA_X + "," + robotA_Y + ")");
System.out.println("Ubicación inicial: robotB(" + robotB_X + "," + robotB_Y + ")");

/* robotA se mueve, robotB también */
int numMov = 0;
do
```

```
{
 numMov++;
 boolean movio = false;
 do
 {
  /* Movimiento al azar del robotA */
  switch(azar.nextInt(8))
  {
   case 0: if (robotA_X>0 && robotA_Y>0)
     { robotA_X--; robotA_Y--; movio=true; } break;
   case 1: if (robotA_Y>0)
     { robotA_Y--; movio=true; } break;
   case 2: if (robotA_X<limiteX && robotA_Y>0)
     { robotA_X++; robotA_Y--; movio=true; } break;
   case 3: if (robotA_X>0)
     { robotA_X--; movio=true; } break;
   case 4: if (robotA_X<limiteX)
     { robotA_X++; movio=true; } break;
   case 5: if (robotA_X>0 && robotA_Y<limiteY )
     { robotA_X--; robotA_Y++; movio=true; } break;
   case 6: if (robotA_Y<limiteY)
     { robotA_Y++; movio=true; } break;
   default: if (robotA_X<limiteX && robotA_Y<limiteX)
      { robotA_X++; robotA_Y++; movio=true; } break;
  }
 }while(movio==false);

 movio = false;
 do
 {
  /* Movimiento al azar del robotB */
  switch(azar.nextInt(8))
  {
   case 0: if (robotB_X>0 && robotB_Y>0)
     { robotB_X--; robotB_Y--; movio=true; } break;
   case 1: if (robotB_Y>0)
     { robotB_Y--; movio=true; } break;
   case 2: if (robotB_X<limiteX && robotB_Y>0)
     { robotB_X++; robotB_Y--; movio=true; } break;
   case 3: if (robotB_X>0)
     { robotB_X--; movio=true; } break;
   case 4: if (robotB_X<limiteX)
     { robotB_X++; movio=true; } break;
   case 5: if (robotB_X>0 && robotB_Y<limiteY )
     { robotB_X--; robotB_Y++; movio=true; } break;
   case 6: if (robotB_Y<limiteY)
     { robotB_Y++; movio=true; } break;
   default: if (robotB_X<limiteX && robotB_Y<limiteX)
      { robotB_X++; robotB_Y++; movio=true; } break;
    }
   }while(movio==false);

  }while (robotA_X!=robotB_X || robotA_Y!=robotB_Y);
  System.out.println("# movimientos para encuentro: " + numMov);
 }
}
```

Para poder comparar bien los dos escenarios, se debe en primer lugar iniciar los robots en las mismas posiciones tanto para el escenario 1 como para el escenario 2, y finalmente simular miles de veces para poder tomar una decisión. El código en Java que une los dos escenarios y simula miles de veces es el siguiente.

```
public class Escenario
{
 Random azar = new Random();
 int robotA_X, robotB_X, robotA_Y, robotB_Y;

 /* Limites del tablero */
 int limiteX = 50;
```

```
int limiteY = 50;

public static void main(String[] args)
{
 Escenario simular = new Escenario();

 /* Acumulado de movimientos */
 int totalMov1 = 0, totalMov2 = 0;

 /* Simula miles de veces */
 int totalIteracion = 100000;
 for (int iteracion=1; iteracion<=totalIteracion; iteracion++)
 {
 simular.UbicaRobotsAzar();

 int robotA_Xorig = simular.robotA_X;
 int robotA_Yorig = simular.robotA_Y;
 int robotB_Xorig = simular.robotB_X;
 int robotB_Yorig = simular.robotB_Y;

 /* Escenario 1: Robot A se mueve, Robot B quieto */
 totalMov1 += simular.Escenario1();

 /* Restaura la posición original de los robots */
 simular.robotA_X = robotA_Xorig;
 simular.robotA_Y = robotA_Yorig;
 simular.robotB_X = robotB_Xorig;
 simular.robotB_Y = robotB_Yorig;

 /*Escenario 2: Robot A se mueve, Robot B también */
 totalMov2 += simular.Escenario2();
 }
 System.out.println("Promedio movimientos Escenario 1: " + (float) totalMov1/totalIteracion);
 System.out.println("Promedio movimientos Escenario 2: " + (float) totalMov2/totalIteracion);
}

/* Robot A se mueve, robot B quieto */
public int Escenario1()
{
 int numMov = 0;
 do
 {
  numMov++;
  MoverRobotA();
 }while (robotA_X!=robotB_X || robotA_Y!=robotB_Y);
 return numMov;
}

/* Los dos robots se mueven */
public int Escenario2()
{
 int numMov = 0;
 do
 {
  numMov++;
  MoverRobotA();
  MoverRobotB();
 }while (robotA_X!=robotB_X || robotA_Y!=robotB_Y);
 return numMov;
}

public void MoverRobotA()
{
  boolean movio = false;
  do
  {
   /* Movimiento al azar del robotA */
   switch(azar.nextInt(8))
   {
    case 0: if (robotA_X>0 && robotA_Y>0)
      { robotA_X--; robotA_Y--; movio=true; } break;
    case 1: if (robotA_Y>0)
      { robotA_Y--; movio=true; } break;
    case 2: if (robotA_X<limiteX && robotA_Y>0)
      { robotA_X++; robotA_Y--; movio=true; } break;
    case 3: if (robotA_X>0)
      { robotA_X--; movio=true; } break;
    case 4: if (robotA_X<limiteX)
```

```
      { robotA_X++; movio=true; } break;
    case 5: if (robotA_X>0 && robotA_Y<limiteY )
      { robotA_X--; robotA_Y++; movio=true; } break;
    case 6: if (robotA_Y<limiteY)
      { robotA_Y++; movio=true; } break;
    default: if (robotA_X<limiteX && robotA_Y<limiteX)
       { robotA_X++; robotA_Y++; movio=true; } break;
   }
  }while(movio==false);
}

public void MoverRobotB()
{
  boolean movio = false;
  do
  {
   /* Movimiento al azar del robotB */
   switch(azar.nextInt(8))
   {
    case 0: if (robotB_X>0 && robotB_Y>0)
      { robotB_X--; robotB_Y--; movio=true; } break;
    case 1: if (robotB_Y>0)
      { robotB_Y--; movio=true; } break;
    case 2: if (robotB_X<limiteX && robotB_Y>0)
      { robotB_X++; robotB_Y--; movio=true; } break;
    case 3: if (robotB_X>0)
      { robotB_X--; movio=true; } break;
    case 4: if (robotB_X<limiteX)
      { robotB_X++; movio=true; } break;
    case 5: if (robotB_X>0 && robotB_Y<limiteY )
      { robotB_X--; robotB_Y++; movio=true; } break;
    case 6: if (robotB_Y<limiteY)
      { robotB_Y++; movio=true; } break;
    default: if (robotB_X<limiteX && robotB_Y<limiteX)
       { robotB_X++; robotB_Y++; movio=true; } break;
   }
  }while(movio==false);
 }

 public void UbicaRobotsAzar()
 {
  /* Ubica al azar robotA y robotB, valida que no
   * queden en la misma celda. */
  do
  {
   robotA_X = azar.nextInt(limiteX);
   robotA_Y = azar.nextInt(limiteY);
   robotB_X = azar.nextInt(limiteX);
   robotB_Y = azar.nextInt(limiteY);
  }while (robotA_X==robotB_X && robotA_Y==robotB_Y);
 }
}
```

Con 100.000 pruebas

Promedio movimientos Escenario 1: 6907.5073
Promedio movimientos Escenario 2: 4784.733

Luego se prueba que es preferible que los dos robots se muevan, si el tablero es cuadrado.

¿Qué sucede cuando el escenario es rectangular? ¿Qué sucede si los dos robots siempre inician en posiciones alejadas? ¿Qué sucede si los dos robots siempre inician en posiciones cercanas? En todos estos casos se debe modificar el software para responder cada una de esas preguntas y volver a ejecutar las simulaciones.

Simulando un proceso empresarial: Filas de Espera

Las filas de espera (también conocidas como líneas de espera o colas) son un proceso muy frecuente en las organizaciones por ejemplo:

- Fila para retirar dinero de un cajero electrónico
- Fila para pagar las facturas de servicios públicos
- Fila para comprar los boletos en el cine
- Fila para disfrutar una atracción turística

A la gente no le gusta esperar: después de un largo tiempo, abandonan la fila. Sin embargo, un servicio muy rápido (atender más rápidamente) tendría un costo muy elevado. Luego es necesario encontrar un balance adecuado.

El objetivo de la simulación es determinar una capacidad de servicio apropiada.

Sistema de filas de espera

Un sistema de filas puede dividirse en dos componentes principales:

- La cola, y,
- El servicio

Los clientes o llegadas vienen en forma individual para recibir el servicio. Estos clientes o llegadas pueden ser:

- Personas
- Automóviles (esperando en el semáforo por ejemplo)
- Máquinas que requieren reparación
- Documentos en espera de ser procesados

Si nadie está en la fila, cuando la persona llega va a ser atendida inmediatamente, de lo contrario, se une a la fila. La longitud de la fila no incluye a la persona que en ese momento está siendo atendida. La formación de la fila obedece a unas reglas o disciplina. Generalmente, ésta disciplina es primero en llegar, primero en ser atendido (FIFO: First Input, First Output), pero, pueden haber otras reglas o filas con prioridades, por ejemplo: en una sala de emergencias médicas, una persona en estado crítico debe ser atendida con alta prioridad así llegue después de una persona que tenga una herida de poca consideración.

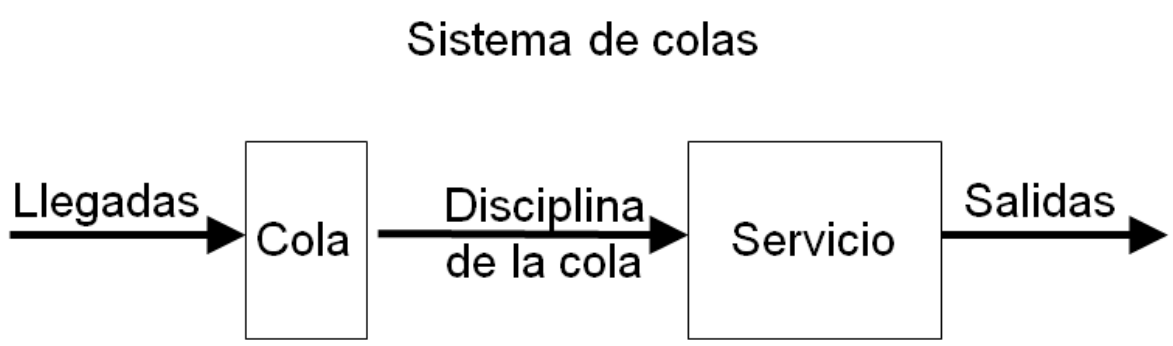

Figura 25: Estructura típica de una fila de espera

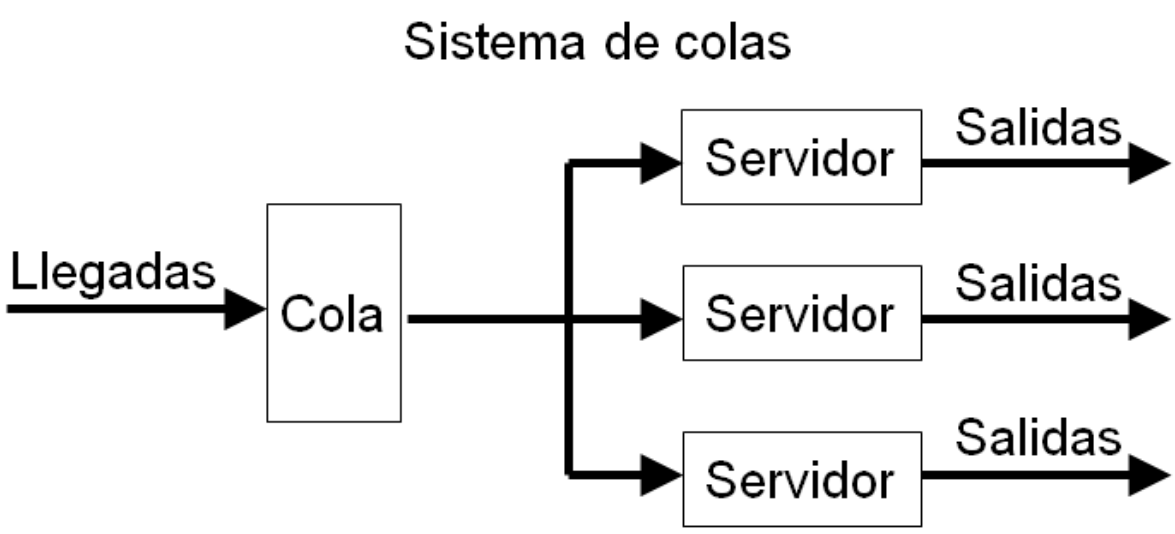

Figura 26: Una línea, múltiples servidores

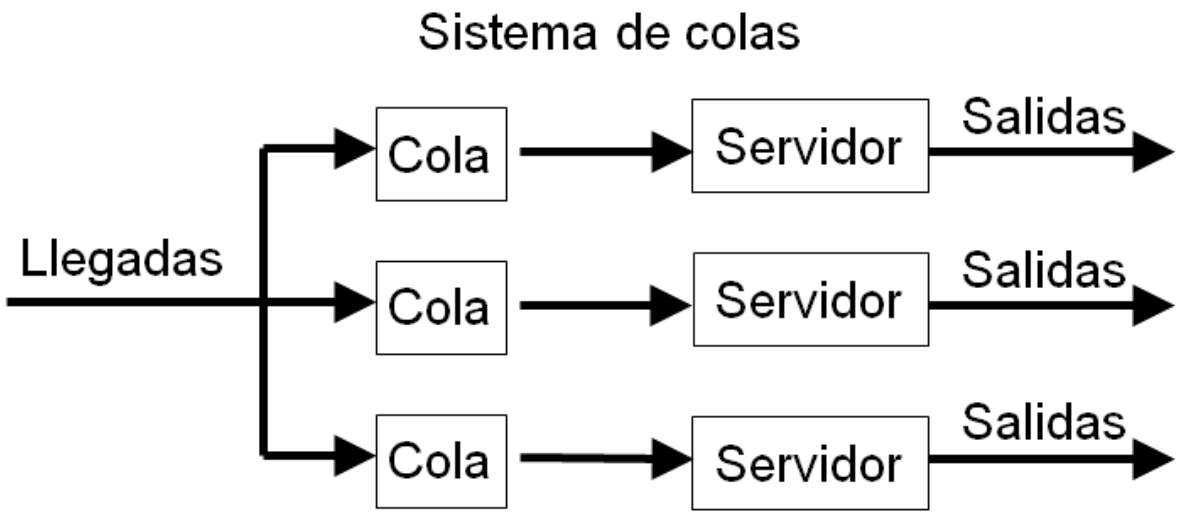

Figura 27: Varias líneas, múltiples servidores

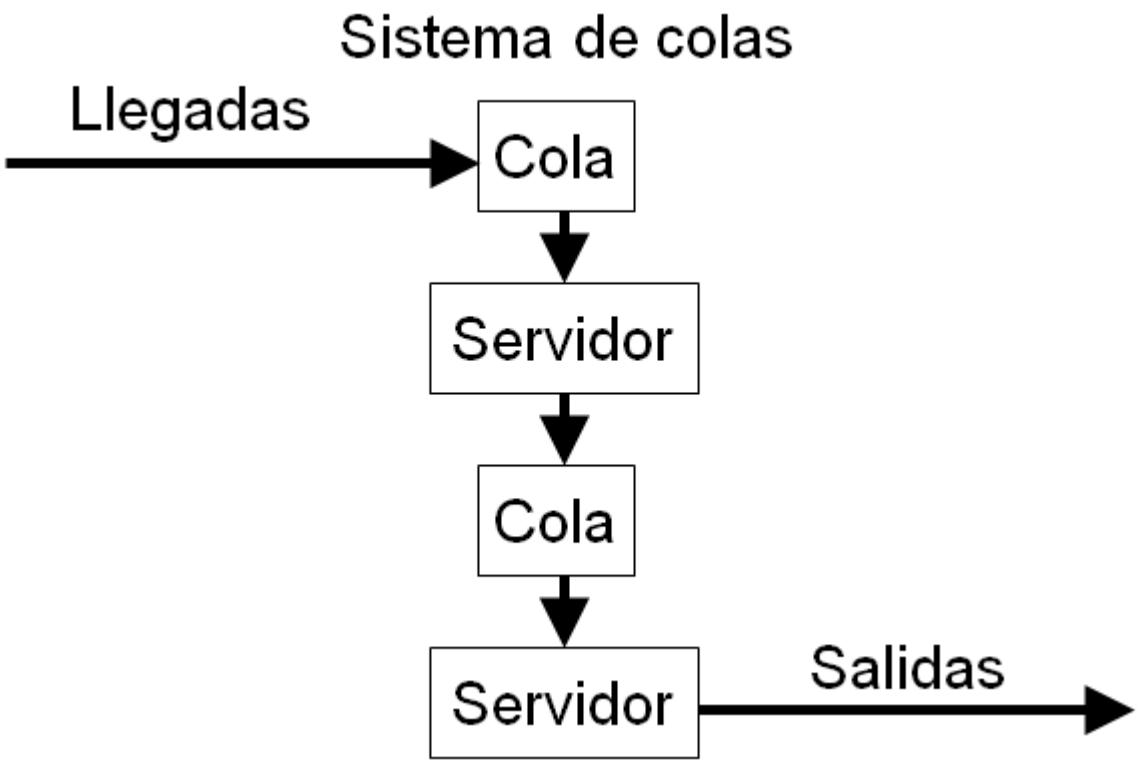

Figura 28: Una línea, servidores secuenciales

Costos de un sistema de filas de espera

1. Costo de espera: Es el costo para la persona tener que esperar. Un sistema con un bajo costo de espera es una fuente importante de competitividad.

2. Costo de servicio: Es el costo de operación del servicio brindado. Es más fácil de estimar (costo del empleado que atiende + costo de hardware/software usado + costo depreciación de los muebles + costo del local…).

Costo total = Costo de espera + Costo del servicio

El objetivo de un sistema de filas es encontrar el sistema del costo total mínimo.

Las llegadas en un sistema de filas de espera

El número esperado de llegadas por unidad de tiempo se llama tasa media de llegadas (λ), por ejemplo, llegan 50 documentos/minuto.

El tiempo esperado entre llegadas es 1/λ. Por ejemplo, si la tasa media de llegadas es λ=20 clientes por hora, significa que el tiempo esperado entre llegadas es 1/λ = 1/20 = 0,05 horas = 3 minutos.

Debido a que esto es una variable aleatoria, es necesario estimar la distribución de probabilidad. Generalmente se supone una distribución exponencial. La última llegada no influye en la probabilidad de llegada de la siguiente.

La fila

El número de clientes en la fila es el número de clientes que esperan a ser atendidos. El número de clientes en el sistema es el número de clientes que hacen fila más el número de clientes que reciben el servicio en ese momento

La capacidad de la fila es el número máximo de clientes que pueden estar en la cola. Generalmente se supone que la fila es infinita aunque también la fila puede ser finita.

El servicio

El servicio puede ser brindado por uno o múltiples servidores. El tiempo de servicio varía de cliente a cliente (por ejemplo, tarda mas tiempo atender a una persona con una compleja transacción financiera que otra persona que solo paga un recibo en efectivo). El tiempo esperado de servicio depende de la tasa media del servicio (μ). El tiempo esperado de servicio equivale a 1/μ. Por ejemplo, si la tasa media de servicio es de 25 clientes por hora, entonces el tiempo esperado de servicio es 1/μ = 1/25 = 0.04 horas = 2.4 minutos.

La simulación

En principio el sistema está en un estado inicial (fila vacía). A medida que se ejecuta la simulación, la fila pasa una condición de estado estable o nivel normal de operación. Cuando se alcanza ese estado, se pueden emitir conclusiones.

Lo buscado en la simulación es la tasa óptima de servicio.

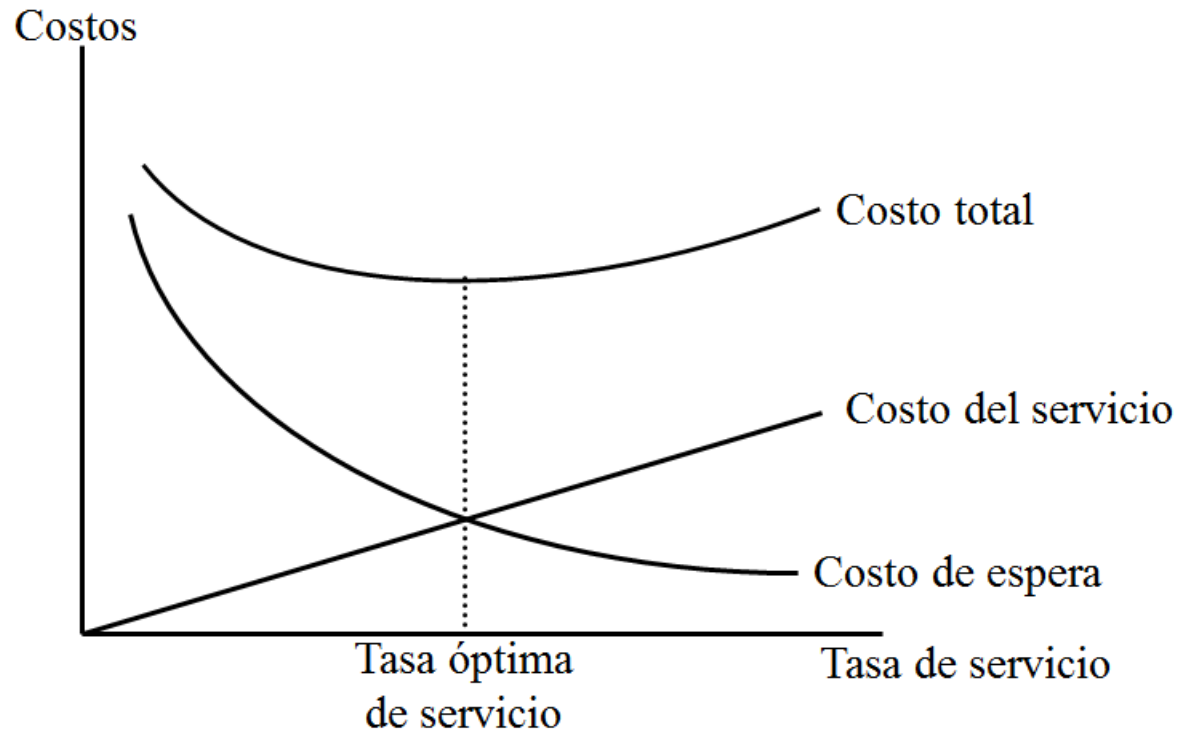

Figura 29: Tasa óptima de servicio

Ejemplo 1 de simulación de una fila de espera

En un sistema de filas de espera se tiene que la tasa de llegada de clientes (λ) y la tasa de atención (μ) son

$$\lambda = 3\ por\ hora$$

$$\mu = 5\ por\ hora$$

La simulación de filas de espera se desarrolla por unidad de tiempo seleccionada, cada 0.1 horas (6 minutos), es decir, dividir la hora en 10 partes.

$$P = 1 - e^{-\lambda * t}$$

Como los tiempos de llegadas y salidas tienen una distribución exponencial, la probabilidad P_A de que un intervalo de 0.1 (1/10) horas incluya una llegada es

$$P_A = 1 - e^{-3*1/10} = 0{,}259$$

Y la probabilidad de que incluya una salida (finalización del servicio), dado que se estaba sirviendo un cliente al inicio del servicio es

$$P_D = 1 - e^{-5*1/10} = 0{,}393$$

Se genera un número aleatorio r1 de varios dígitos y se deduce lo siguiente:

Si $r_1 < 0{,}259$ entonces ocurre una llegada

Si $r_1 \geq 0{,}259$ entonces no ocurre una llegada

De igual manera, se genera otro número aleatorio r2 y se deduce lo siguiente:

Si $r_2 < 0{,}393$ entonces ocurre una salida

Si $r_2 \geq 0{,}393$ entonces no ocurre una salida

En la tabla se simula la fila avanzando de 1/10 por hora es decir 6 minutos:

Tiempo	r_1	¿Llegada en el intervalo?	r_2	¿Salida en el intervalo?
0	0		0	
6	0,096	Si	-	
12	0,569	No	0,665	No
18	0,764	No	0,842	No
24	0,492	No	0,224	Si
30	0,950	No	-	
36	0,610	No	-	
42	0,145	Si	-	
48	0,484	No	0,552	No
54	0,350	No	0,590	No
60	0,430	No	0,041	Si

Número de Personas que llegaron a la cola en una hora: 2

Número de Personas atendidas en una hora: 2

Tiempo promedio de las personas en la cola en una hora: 0

Tiempo promedio de las personas siendo atendidas: ((24-6)+(60-42))/2=18 minutos

Esa simulación es por avance por unidad de tiempo fija que se caracteriza por lo siguiente:

1. Se avanza el tiempo por una cantidad fija pequeña.
2. Se actualiza el sistema determinando los eventos que ocurrieron durante este lapso y el estado del sistema que resulta. También se registra la información deseada sobre el comportamiento del sistema.

Ejemplo 2 de simulación de una fila de espera

En un sistema de filas de espera se tiene que la tasa de llegada de clientes (λ) y la tasa de atención (μ) son

$$\lambda = 3\ por\ hora$$

$$\mu = 5\ por\ hora$$

La simulación de filas de espera se desarrolla por unidad de tiempo seleccionada, cada minuto, es decir, dividir la hora en 60 partes.

$$P = 1 - e^{-\lambda * t}$$

Como los tiempos de llegadas y de salidas tienen una distribución exponencial, la probabilidad P_A de que un intervalo de 1/60 horas incluya una llegada es

$$P_A = 1 - e^{-3*1/60} = 0{,}04877058$$

Y la probabilidad de que incluya una salida (finalización del servicio), dado que se estaba sirviendo un cliente al inicio del servicio es

$$P_D = 1 - e^{-5*1/60} = 0{,}07995559$$

Se genera un número aleatorio r1 de varios dígitos y se deduce lo siguiente:

Si $r_1 < 0{,}04877058$ entonces ocurre una llegada

Si $r_1 \geq 0{,}04877058$ entonces no ocurre una llegada

De igual manera, se genera otro número aleatorio r2 y se deduce lo siguiente:

Si $r_2 < 0{,}07995559$ entonces ocurre una salida

Si $r_2 \geq 0{,}07995559$ entonces no ocurre una salida

En la tabla se simula la fila avanzando de 1 minuto:

Tiempo (minuto)	r_1	¿Llega usuario?	Turno	r_2	¿Atendió usuario?	¿Quién?
0	0,77650442			0,60117171	NO	
1	0,94275093			0,05224472		
2	0,66641548			0,38094446		
3	0,76487987			0,21211305		
4	0,13243907			0,41858164		
5	0,17974426			0,13027231		
6	0,63052314			0,87731285		
7	0,66448849			0,73302382		
8	0,30974182			0,44864403		
9	0,75790031			0,60791612		
10	0,02571072	SI	1	0,92136336		
11	0,58543407			0,78165786		
12	0,41782067			0,93462978		
13	0,62793321			0,62701941		
14	0,17784589			0,28448186		
15	0,94571115			0,60368954		
16	0,68005684			0,76625536		
17	0,16722402			0,85211953		
18	0,16272679			0,59815458		
19	0,69040802			0,006085	SI	1
20	0,14082537			0,70623304		
21	0,95813322			0,17045818		
22	0,03703422	SI	2	0,61885717		
23	0,60438102			0,10086073		
24	0,88603326			0,12716227		
25	0,7886413			0,06943013	SI	2
26	0,88286028			0,73059954		
27	0,82766029			0,05977628		
28	0,94172301			0,36491122		
29	0,14507122			0,37841808		
30	0,40839799			0,44239527		
31	0,5427537			0,47731154		
32	0,36735393			0,45366328		
33	0,07392429			0,44987745		
34	0,31898798			0,54688381		
35	0,24129619			0,37095574		
36	0,90444053			0,27273652		
37	0,53173599			0,45067639		
38	0,68145796			0,47956027		
39	0,14360894			0,63794235		
40	0,34446455			0,34905287		
41	0,74873241			0,38858364		
42	0,41690344			0,90416263		
43	0,92737488			0,86647223		
44	0,53374722			0,92475769		
45	0,66173763			0,15062455		
46	0,9057334			0,38886044		
47	0,88985574			0,71692401		
48	0,78101784			0,31736587		
49	0,95652728			0,28631476		
50	0,15883767			0,31467531		
51	0,79639011			0,58351036		
52	0,16125232			0,91119773		
53	0,88240816			0,07451972		
54	0,97167163			0,72185893		
55	0,78877336			0,1010477		
56	0,32002766			0,6163067		
57	0,43980554			0,5669924		

58	0,98734995			0,08371423		
59	0,25659035			0,88138916		
60	0,93067502			0,2499344		

Ejemplo 3 de simulación de una fila de espera

En una organización, las personas pueden realizar hasta cuatro trámites distintos. Cada trámite tiene una distribución uniforme de tiempo para ser atendido:

Trámite A: Entre A1 y A2 minutos en atenderse (distribución uniforme)

Trámite B: Entre B1 y B2 minutos en atenderse (distribución uniforme)

Trámite C: Entre C1 y C2 minutos en atenderse (distribución uniforme)

Trámite D: Entre D1 y D2 minutos en atenderse (distribución uniforme)

Las personas cuando llegan a la organización hacen uno, dos, tres o los cuatro trámites (por ejemplo, Pedro necesita hacer el trámite A y D, Liliana necesita hacer solo el trámite C, Joaquín debe hacer los cuatro trámites y así con otros usuarios). Es uniforme la probabilidad de cuantos trámites a hacer (por ejemplo, si son 1000 personas, 250 sólo hacen un trámite, 250 hacen dos trámites, 250 hacen tres trámites, 250 hacen cuatro trámites).

Hay una discusión con respecto a cómo se deben atender esas personas, un grupo dice que debe haber una sola cola de personas y esa cola atendida por los cuatro cajeros. Otro grupo alega que es mejor que hayan cuatro colas, una cola solo atiende trámites A, otra trámites B, otra trámites C y una última dedicada a trámites D. Y hay un escéptico que dice que ambas soluciones son lo mismo.

¿Quién tiene la razón? Porque si fuesen las cuatro colas, en el caso de Pedro, él debe hacer la cola para el trámite A y luego hacer la cola para el trámite D, en el caso de Joaquín tiene que hacer las cuatro colas, pero para Liliana solo es hacer la cola del trámite C.

Tómese el caso de Pedro: una vez lo atiendan del trámite A, instantáneamente hace la cola del trámite D. Así con todos los usuarios. Luego cada cola se llena ya sea por usuarios que recién entran al sistema o usuarios que ya terminaron otros trámites.

Si un usuario tiene que hacer varios trámites, esta persona inteligentemente buscará hacer la cola más corta.

Desarrollar la simulación variando la tasa de llegada exponencial de las personas y concluir cuál es la mejor estrategia o si ambas tienen el mismo resultado.

Desarrollo del modelo

En este sistema se tiene unos objetos claramente definidos: las personas. Esas personas son las que deben hacer unos trámites y son las que hacen las filas. Luego se define así en Java la clase persona:

Persona.java

```
import java.util.Random;

/* Gestiona la persona que hace la fila de espera */
public class Persona
{
 int codigo; /* Código del cliente */

 int tramA; /* Duración del trámite A */
 int tramB; /* Duración del trámite B */
 int tramC; /* Duración del trámite C */
 int tramD; /* Duración del trámite D */

 int tramAini; /* Duración original del trámite A */
 int tramBini; /* Duración original del trámite B */
 int tramCini; /* Duración original del trámite C */
 int tramDini; /* Duración original del trámite D */

 /* 1=Haciendo fila, 2=Atendiéndose, 3=Sale del sistema */
 int estado;

 /* En que minuto llegó la persona a la cola */
 int minLlega;

 /* En que minuto sale del sistema */
 int minSale;
```

```
/* En que minuto accede a un servidor */
int minServidor;

public int getCodigo() { return codigo; }
public int getTramiteA() { return tramA; }
public int getTramiteB() { return tramB; }
public int getTramiteC() { return tramC; }
public int getTramiteD() { return tramD; }
public int getEstado() { return estado; }
public int getMinutoLlega() { return minLlega; }
public int getMinutoSale() { return minSale; }
public void setEstado(int estado) { this.estado = estado; }

public void DisminuyeTramite(int minuto, int tramite)
{
 //Almacena el minuto en que es atendida por el servidor
 if (minServidor == 0) minServidor = minuto;

 switch (tramite)
 {
 case 0: tramA--; break;
 case 1: tramB--; break;
 case 2: tramC--; break;
 case 3: tramD--; break;
 }

 /* Si terminó de atenderse todos los trámites,
  la persona sale del sistema */
 if (tramA == 0 && tramB == 0 && tramC == 0 && tramD == 0)
 {
  estado = 3; //Sale del sistema
  minSale = minuto;
 }
}

/* Comportamiento de la persona por cada minuto que
  es atendida en el escenario 1 */
public void AvanzaAtencion(int minuto)
{
 /* Almacena el minuto en que es atendida
    por el servidor */
 if (minServidor == 0) minServidor = minuto;

 /* Disminuye en un minuto algún trámite
    que debe hacer */
 if (tramA > 0) { tramA--; return; }
 if (tramB > 0) { tramB--; return; }
 if (tramC > 0) { tramC--; return; }
 if (tramD > 0) { tramD--; return; }

 /* Si terminó de atenderse todos los trámites,
  la persona sale del sistema */
 estado = 3;
 minSale = minuto;
}

/* Muestra informe */
public String getTiempos()
{
 String Texto;
 Texto = "TramiteA= " + tramAini;
 Texto += "\tTramiteB= " + tramBini;
 Texto += "\tTramiteC= " + tramCini;
 Texto += "\tTramiteD= " + tramDini;
 Texto += "\tLlega = " + minLlega;
 Texto += "\tAccede = " + minServidor;
```

```
 Texto += "\tSale = " + minSale;
 return Texto;
}

/* Llega una persona a la fila */
public Persona(int Codigo, Random azar, int MinLlega,
    int tramAmin, int tramAmax, int tramBmin,
    int tramBmax, int tramCmin, int tramCmax,
    int tramDmin, int tramDmax)
{
 /* Código del cliente */
 this.codigo = Codigo;

 /* Almacena en que minuto llega a la fila */
 this.minLlega = MinLlega;
 this.minSale = 0;
 this.minServidor = 0;

 /* Estado de la persona por defecto es inactiva */
 estado = 0;

 /* Inicializa los tiempos de los trámites a cero */
tramA=0;
tramB=0;
tramC=0;
tramD=0;

/* De 1 a 4 trámites */
int TotalTramite = (int) (azar.nextDouble()*(4-1)+1);

/* Dependiendo del número de trámites,
 selecciona que trámites tendrá que
 hacer la persona y cuanto tiempo
 por trámite */
while (TotalTramite > 0)
{
double numAzar = azar.nextDouble();
if (numAzar < 0.25 && tramA == 0)
{
 tramA = (int) (azar.nextDouble()*(tramAmax - tramAmin) + tramAmin);
 TotalTramite--;
}

if (numAzar >= 0.25 && numAzar<0.5 && tramB == 0)
{
 tramB = (int) (azar.nextDouble() * (tramBmax - tramBmin) + tramBmin);
 TotalTramite--;
}

if (numAzar >= 0.5 && numAzar < 0.75 && tramC == 0)
{
 tramC = (int) (azar.nextDouble() * (tramCmax - tramCmin) + tramCmin);
 TotalTramite--;
}

if (numAzar >= 0.75 && tramD == 0)
{
 tramD = (int) (azar.nextDouble() * (tramDmax - tramDmin) + tramDmin);
 TotalTramite--;
}
}

/* Guarda originalmente la duración de los trámites */
tramAini = tramA;
tramBini = tramB;
tramCini = tramC;
```

```
  tramDini = tramD;
 }
}
```

El siguiente programa sirve para simular el caso de una sola fila atendida por los cuatro(4) servicios

SimularFila.java

```
import java.util.ArrayList;
import java.util.Random;

public class SimularFila
{
 public static void main(String Parametros[])
 {
   /* Sólo un generador de números aleatorios */
  Random azar = new Random();

  /* Condiciones de la simulación */
  final double TASALLEGADAHORA = 5;

  /* Duración mínima y máxima de los trámites en minutos */
  final int TRAMITEAMIN = 2;
  final int TRAMITEAMAX = 5;
  final int TRAMITEBMIN = 3;
  final int TRAMITEBMAX = 6;
  final int TRAMITECMIN = 1;
  final int TRAMITECMAX = 4;
  final int TRAMITEDMIN = 4;
  final int TRAMITEDMAX = 7;

  /* Total minutos a simular */
  final int TOTALMINUTOS = 600;

  /* Llegada de clientes por hora */
  double tasaLlega = TASALLEGADAHORA;
  double limLlega = 1 - Math.exp(-tasaLlega * 1 / 60);

  /* La cola es representada en una lista enlazada */
  ArrayList<Persona> fila = new ArrayList<Persona>();

  /* Número de servidores libres */
  int servLibre = 4;

  /* Código del cliente */
  int codigo = 0;

  /* Simulación minuto a minuto */
  for (int minuto = 1; minuto <= TOTALMINUTOS; minuto++)
  {
   /* Comprueba si llega una persona en ese minuto */
   if (azar.nextDouble() < limLlega)
   {
    Persona persona = new Persona(codigo, azar, minuto, TRAMITEAMIN, TRAMITEAMAX,
TRAMITEBMIN, TRAMITEBMAX, TRAMITECMIN, TRAMITECMAX, TRAMITEDMIN, TRAMITEDMAX);
    persona.setEstado(1); /* De una vez va a hacer cola */
    fila.add(persona);
    codigo++;
   }

   /* Se atiende la fila siempre y cuando
      existan personas en la fila */
   for(int persona = 0; persona < fila.size(); persona++)
   {
    /* Si hay servidores libres, chequea si hay usuarios
       en espera. Dado el caso el usuario pasa a ser
       atendido y un servidor se usa para atención. */
```

```
    if (servLibre > 0 && fila.get(persona).getEstado() == 1)
    {
     fila.get(persona).setEstado(2);
     servLibre--;
    }

    /* Si ya la estaban atendiendo chequea si terminó o no */
    if (fila.get(persona).getEstado() == 2)
    {
     fila.get(persona).AvanzaAtencion(minuto);

     /* Si terminó entonces aumenta el número de servidores libres */
     if (fila.get(persona).getEstado() == 3)
      servLibre++;
    }
   }
}

/* Muestra el resultado por pantalla */
int contador = 1;
for(int persona = 0; persona < fila.size(); persona++)
{
   System.out.println("Cliente [" + contador + "] " + fila.get(persona).getTiempos());
   contador++;
  }

  /* Muestra los tiempos finales */
  int Acumula = 0;
  contador = 1;
  int Atendidos = 0;
  for(int persona = 0; persona < fila.size(); persona++)
  {
   if (fila.get(persona).getEstado() == 3)
   {
    int TiempoTramitando = fila.get(persona).getMinutoSale() -
fila.get(persona).getMinutoLlega();
    Acumula += TiempoTramitando;
    System.out.println("Cliente [" + contador + "] Llegó: " +
fila.get(persona).getMinutoLlega() + " Terminó: " + fila.get(persona).getMinutoSale()
+ " Diferencia: " + TiempoTramitando);
    Atendidos++;
   }
   contador++;
  }
  System.out.println("Tiempo Promedio = " + (float)Acumula / (float)Atendidos);
 }
}
```

Un ejemplo del resultado lanzado por este programa es el siguiente

```
Cliente [1] Llegó: 9 Terminó: 16 Diferencia: 7
Cliente [2] Llegó: 16 Terminó: 26 Diferencia: 10
Cliente [3] Llegó: 19 Terminó: 28 Diferencia: 9
Cliente [4] Llegó: 43 Terminó: 52 Diferencia: 9
Cliente [5] Llegó: 44 Terminó: 53 Diferencia: 9
Cliente [6] Llegó: 46 Terminó: 52 Diferencia: 6
Cliente [7] Llegó: 50 Terminó: 60 Diferencia: 10
Cliente [8] Llegó: 63 Terminó: 67 Diferencia: 4
Cliente [9] Llegó: 69 Terminó: 79 Diferencia: 10
Cliente [10] Llegó: 70 Terminó: 74 Diferencia: 4
Cliente [11] Llegó: 71 Terminó: 76 Diferencia: 5
Cliente [12] Llegó: 86 Terminó: 99 Diferencia: 13
Cliente [13] Llegó: 87 Terminó: 96 Diferencia: 9

Tiempo Promedio = 6.4166665
```

En el ejemplo, se observa que el cliente 1 llego en el minuto 9, le atendieron sus trámites y salió del sistema en el minuto 16, es decir, estuvo haciendo fila y esperando a gestionar cada trámite por 7 minutos.

En el ejemplo de la simulación el tiempo promedio de los clientes dentro del sistema fue de 6.4166 minutos.

Cambiando el programa para que simule el escenario 2, en el cual son cuatro filas especializadas se usa el siguiente programa:

```
import java.util.ArrayList;
import java.util.Random;

public class SimularMultipleFila
{
 public static void main(String Parametros[])
 {
  /* Sólo un generador de números aleatorios */
  Random azar = new Random();

  /* Condiciones de la simulación */
  final double TASALLEGADAHORA = 5;

  /* Duración mínima y máxima de los trámites en minutos */
  final int TRAMITEAMIN = 2;
  final int TRAMITEAMAX = 5;
  final int TRAMITEBMIN = 3;
  final int TRAMITEBMAX = 6;
  final int TRAMITECMIN = 1;
  final int TRAMITECMAX = 4;
  final int TRAMITEDMIN = 4;
  final int TRAMITEDMAX = 7;

  /* Total minutos a simular */
  final int TOTALMINUTOS = 600;

  /* Llegada de clientes por hora */
  double tasaLlega = TASALLEGADAHORA;
  double limLlega = 1 - Math.exp(-tasaLlega * 1 / 60);

  /* Código del cliente */
  int codigo = 0;

  /* Filas vacías */
  ArrayList<Persona> filaIni = new ArrayList<Persona>();
  ArrayList<Persona> filaA = new ArrayList<Persona>();
  ArrayList<Persona> filaB = new ArrayList<Persona>();
  ArrayList<Persona> filaC = new ArrayList<Persona>();
  ArrayList<Persona> filaD = new ArrayList<Persona>();

  int[] masCorta = new int[4];
  for (int Cont = 0; Cont < masCorta.length; Cont++) masCorta[Cont] = 0;

  /* Simulación minuto a minuto */
  for (int minuto = 1; minuto <= TOTALMINUTOS; minuto++)
  {
   /* Chequea persona por persona para asignarla a una cola */
   for(int persona = 0; persona < filaIni.size(); persona++)
   {
    /* Asigna la persona a hacer la cola de menor tamaño */
    if (filaIni.get(persona).getEstado() == 0)
    {
     filaIni.get(persona).setEstado(1); /* Hace la fila */

     masCorta[0] = filaA.size();
     masCorta[1] = filaB.size();
     masCorta[2] = filaC.size();
     masCorta[3] = filaD.size();
```

```
     int ColaHacer = RetornaFilaMenor(masCorta, filaIni.get(persona).getTramiteA(),
filaIni.get(persona).getTramiteB(), filaIni.get(persona).getTramiteC(),
filaIni.get(persona).getTramiteD());
     String filaHacer = "";
     switch (ColaHacer)
     {
      case 0: filaA.add(filaIni.get(persona)); filaHacer = "A"; break;
      case 1: filaB.add(filaIni.get(persona)); filaHacer = "B"; break;
      case 2: filaC.add(filaIni.get(persona)); filaHacer = "C"; break;
      case 3: filaD.add(filaIni.get(persona)); filaHacer = "D"; break;
     }
     System.out.println("Minuto: " + minuto + " Cliente [" +
filaIni.get(persona).getCodigo() + "] hace fila: " + filaHacer);
    }
   }

   /* Mira cada cola para ir disminuyendo
      el trámite de la persona que está atendiendo */
   for(int persona = 0; persona < filaA.size(); persona++)
   {
    filaA.get(persona).DisminuyeTramite(minuto, 0);
    System.out.println("Minuto: " + minuto + " Cliente [" +
filaA.get(persona).getCodigo() + "] disminuye trámite A");
    if (filaA.get(persona).getTramiteA() == 0)
    {
     if (filaA.get(persona).getEstado()!=3 ) filaA.get(persona).setEstado(0);
     System.out.println("Minuto: " + minuto + " Cliente [" +
filaA.get(persona).getCodigo() + "] FINALIZA trámite A");
     filaA.remove(persona);
    }
    break;
   }

   for(int persona = 0; persona < filaB.size(); persona++)
   {
    filaB.get(persona).DisminuyeTramite(minuto, 1);
    System.out.println("Minuto: " + minuto + " Cliente [" +
filaB.get(persona).getCodigo() + "] disminuye trámite B");
    if (filaB.get(persona).getTramiteB() == 0)
    {
     if (filaB.get(persona).getEstado()!=3 ) filaB.get(persona).setEstado(0);
     System.out.println("Minuto: " + minuto + " Cliente [" +
filaB.get(persona).getCodigo() + "] FINALIZA trámite B");
     filaB.remove(persona);
    }
    break;
   }

   for(int persona = 0; persona < filaC.size(); persona++)
   {
    filaC.get(persona).DisminuyeTramite(minuto, 2);
    System.out.println("Minuto: " + minuto + " Cliente [" +
filaC.get(persona).getCodigo() + "] disminuye trámite C");
    if (filaC.get(persona).getTramiteC() == 0)
    {
     if (filaC.get(persona).getEstado()!=3 ) filaC.get(persona).setEstado(0);
     System.out.println("Minuto: " + minuto + " Cliente [" +
filaC.get(persona).getCodigo() + "] FINALIZA trámite C");
     filaC.remove(persona);
    }
    break;
   }

   for(int persona = 0; persona < filaD.size(); persona++)
   {
    filaD.get(persona).DisminuyeTramite(minuto, 3);
```

```
    System.out.println("Minuto: " + minuto + " Cliente [" +
filaD.get(persona).getCodigo() + "] disminuye trámite D");
    if (filaD.get(persona).getTramiteD() == 0)
    {
     if (filaD.get(persona).getEstado()!=3 ) filaD.get(persona).setEstado(0);
     System.out.println("Minuto: " + minuto + " Cliente [" +
filaD.get(persona).getCodigo() + "] FINALIZA trámite D");
     filaD.remove(persona);
    }
    break;
   }

   /* Comprueba si llega una persona en ese minuto */
   if (azar.nextDouble() < limLlega)
   {
    Persona persona = new Persona(codigo, azar, minuto, TRAMITEAMIN, TRAMITEAMAX,
TRAMITEBMIN, TRAMITEBMAX, TRAMITECMIN, TRAMITECMAX, TRAMITEDMIN, TRAMITEDMAX);
    filaIni.add(persona);
    codigo++;

    System.out.println(" ");
    System.out.println("Minuto: " + minuto + " Cliente [" + persona.getCodigo() + "]
Llega. Trámite A = " + persona.getTramiteA() + " Trámite B = " + persona.getTramiteB()
+ " Trámite C = " + persona.getTramiteC() + " Trámite D = " + persona.getTramiteD());

   }

  }

  /* Muestra los tiempos finales */
  System.out.println("======== INDICADORES FINALES ==========");
  int acumula = 0;
  int atendidos = 0;
  for(int persona = 0; persona < filaIni.size(); persona++)
  {
   if (filaIni.get(persona).getEstado() == 3)
   {
    int TiempoTramitando = filaIni.get(persona).getMinutoSale() -
filaIni.get(persona).getMinutoLlega();
    acumula += TiempoTramitando;
    atendidos++;
    System.out.println("Cliente [" + persona + "] Llegó: " +
filaIni.get(persona).getMinutoLlega() + " Terminó: " +
filaIni.get(persona).getMinutoSale() + " Diferencia: " + TiempoTramitando);
   }
  }
  System.out.println("Tiempo Promedio = " + (float)acumula / (float)atendidos);
 }


 /* Dependiendo que tramite debe hacer el usuario, retorna cual cola corta debe hacer
*/
 private static int RetornaFilaMenor(int[] masCorta, int TramiteA, int TramiteB, int
TramiteC, int TramiteD)
 {
  int[] masCortaTmp = new int[4];
  for (int cont = 0; cont < masCorta.length; cont++) masCortaTmp[cont] =
masCorta[cont];
  if (TramiteA == 0) masCortaTmp[0] = 99999999;
  if (TramiteB == 0) masCortaTmp[1] = 99999999;
  if (TramiteC == 0) masCortaTmp[2] = 99999999;
  if (TramiteD == 0) masCortaTmp[3] = 99999999;

  if (masCortaTmp[0] <= masCortaTmp[1] && masCortaTmp[0] <= masCortaTmp[2] &&
masCortaTmp[0] <= masCortaTmp[3])
   return 0;
```

```
  else if (masCortaTmp[1] <= masCortaTmp[0] && masCortaTmp[1] <= masCortaTmp[2] &&
masCortaTmp[1] <= masCortaTmp[3])
    return 1;
  else if (masCortaTmp[2] <= masCortaTmp[0] && masCortaTmp[2] <= masCortaTmp[1] &&
masCortaTmp[2] <= masCortaTmp[3])
    return 2;
  else
    return 3;
  }
}
```

Un ejemplo de su ejecución es el siguiente

```
Minuto: 38 Cliente [0] Llega. Trámite A = 0 Trámite B = 5 Trámite C = 2 Trámite D = 0
Minuto: 39 Cliente [0] hace fila: B
Minuto: 39 Cliente [0] disminuye trámite B
Minuto: 40 Cliente [0] disminuye trámite B
Minuto: 41 Cliente [0] disminuye trámite B
Minuto: 42 Cliente [0] disminuye trámite B
Minuto: 43 Cliente [0] disminuye trámite B
Minuto: 43 Cliente [0] FINALIZA trámite B
Minuto: 44 Cliente [0] hace fila: C
Minuto: 44 Cliente [0] disminuye trámite C
Minuto: 45 Cliente [0] disminuye trámite C
Minuto: 45 Cliente [0] FINALIZA trámite C

Minuto: 46 Cliente [1] Llega. Trámite A = 2 Trámite B = 5 Trámite C = 2 Trámite D = 0
Minuto: 47 Cliente [1] hace fila: A
Minuto: 47 Cliente [1] disminuye trámite A
Minuto: 48 Cliente [1] disminuye trámite A
Minuto: 48 Cliente [1] FINALIZA trámite A
Minuto: 49 Cliente [1] hace fila: B
Minuto: 49 Cliente [1] disminuye trámite B
Minuto: 50 Cliente [1] disminuye trámite B

Minuto: 50 Cliente [2] Llega. Trámite A = 0 Trámite B = 0 Trámite C = 1 Trámite D = 5
Minuto: 51 Cliente [2] hace fila: C
Minuto: 51 Cliente [1] disminuye trámite B
Minuto: 51 Cliente [2] disminuye trámite C
Minuto: 51 Cliente [2] FINALIZA trámite C
Minuto: 52 Cliente [2] hace fila: D
Minuto: 52 Cliente [1] disminuye trámite B
Minuto: 52 Cliente [2] disminuye trámite D
Minuto: 53 Cliente [1] disminuye trámite B
Minuto: 53 Cliente [1] FINALIZA trámite B
Minuto: 53 Cliente [2] disminuye trámite D
Minuto: 54 Cliente [1] hace fila: C
Minuto: 54 Cliente [1] disminuye trámite C
Minuto: 54 Cliente [2] disminuye trámite D
Minuto: 55 Cliente [1] disminuye trámite C
Minuto: 55 Cliente [1] FINALIZA trámite C
Minuto: 55 Cliente [2] disminuye trámite D
Minuto: 56 Cliente [2] disminuye trámite D
Minuto: 56 Cliente [2] FINALIZA trámite D
….

========== INDICADORES FINALES ============

Cliente [0] Llegó: 38 Terminó: 45 Diferencia: 7
Cliente [1] Llegó: 46 Terminó: 55 Diferencia: 9
Cliente [2] Llegó: 50 Terminó: 56 Diferencia: 6
Cliente [3] Llegó: 60 Terminó: 65 Diferencia: 5
Cliente [4] Llegó: 63 Terminó: 75 Diferencia: 12
Cliente [5] Llegó: 77 Terminó: 91 Diferencia: 14

…

Tiempo Promedio = 7.1714287
```

Cómo se puede observar, el tiempo promedio de espera de las personas aumentó al usar filas especializadas. Por supuesto, no se puede sacar conclusiones con una sola prueba. Es necesario hacer varias veces. En la siguiente tabla, se muestran las pruebas hechas.

Prueba	Escenario 1: Una sola fila y 4 servicios	Escenario 2: 4 filas, 4 servicios,

1	7,088889	5,808511
2	7,714286	6,92
3	7,2244897	6,521739
4	7,8913045	6,375
5	6,962963	6,847826
6	5,9166665	8,290909
7	7,0	6,4130435
8	6,8125	8,04878
9	7,470588	8,404255
10	6,86	6,3333335
11	6,4	6,9166665
12	6,757576	8,513514
13	7,625	6,7708335
14	7,2765956	7,357143
15	6,8913045	7,236842
Promedio	**7,0594775**	**7,1172264**

Las pruebas en si muestran prácticamente un empate entre ambos escenarios.

Simulando un proceso empresarial: Inventarios

Las empresas hacen uso de inventarios para guardar material como mercancías ya sea para vender al público o como insumos para generar nuevos productos. El manejo de inventarios presenta una serie de dilemas para el empresario porque hay dos variables aleatorias: la demanda del producto que se encuentra almacenado en bodega y el tiempo que tarda el proveedor del producto en traerlo a la bodega.

Por el lado de los costos:

No es lo mismo almacenar 10 unidades de un producto a tener almacenadas 10.000 unidades del mismo. Las condiciones cambian, por tamaño del sitio en donde se debe almacenar, vigilancia, administración, etc.

No es lo mismo hacer un pedido al proveedor cada mes a hacer un pedido cada 2 días, el proveedor cobra cada vez por enviar el producto.

Y existe un costo llamado "pérdida de goodwill" o "pérdida de prestigio de la empresa" o "pérdida de buena imagen a medida que la bodega no tenga la cantidad suficiente para satisfacer la demanda".

Hay dos preguntas que deben resolverse:

¿Cuántas unidades de producto debe pedírsele al proveedor cada vez que se hace el pedido?

¿Cuál sería la frecuencia en días de pedido de producto al proveedor?

Cuánto	Cuando
Adquirir una cantidad (Q) fija	Adquirir por un período (T) fijo
Adquirir una cantidad (Q) de tal manera que el inventario se llegue a un tope máximo.	Adquirir cuando se llegue a un tope mínimo del inventario

Políticas de Inventario

Sistema de Punto de Reorden (Sistema RQ)

Cuando el inventario esté igual o por debajo del punto de reorden R se genera una orden de Q unidades fijas. La simulación sirve para definir el mejor R y Q. Se debe anotar que así se haga el pedido justo cuando el inventario caiga por debajo del punto R, hay un tiempo de espera no determinado hasta que el proveedor traiga la mercancía, mientras tanto el empresario debe defenderse con lo que le queda en el inventario para atender la demanda variable.

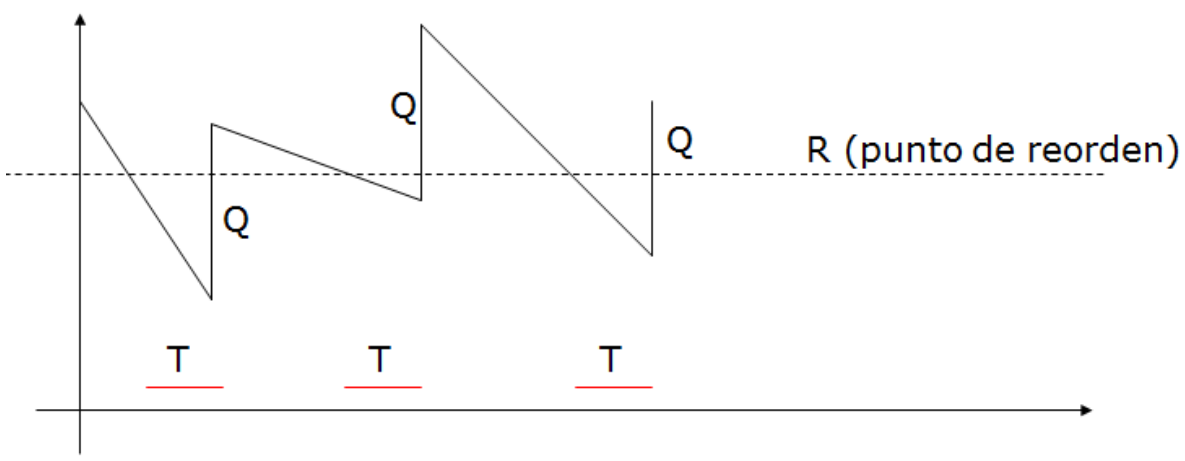

Figura 30: Sistema RQ de inventario

Sistema de revisión periódica o Sistema T

Cada T unidades de tiempo, revisa el inventario existente y genera una orden por una cantidad tal que eleve el inventario al nivel máximo S. La simulación sirve para definir T y S.

En el gráfico, el modelo retira un factor variable: el tiempo de espera de entrega del proveedor.

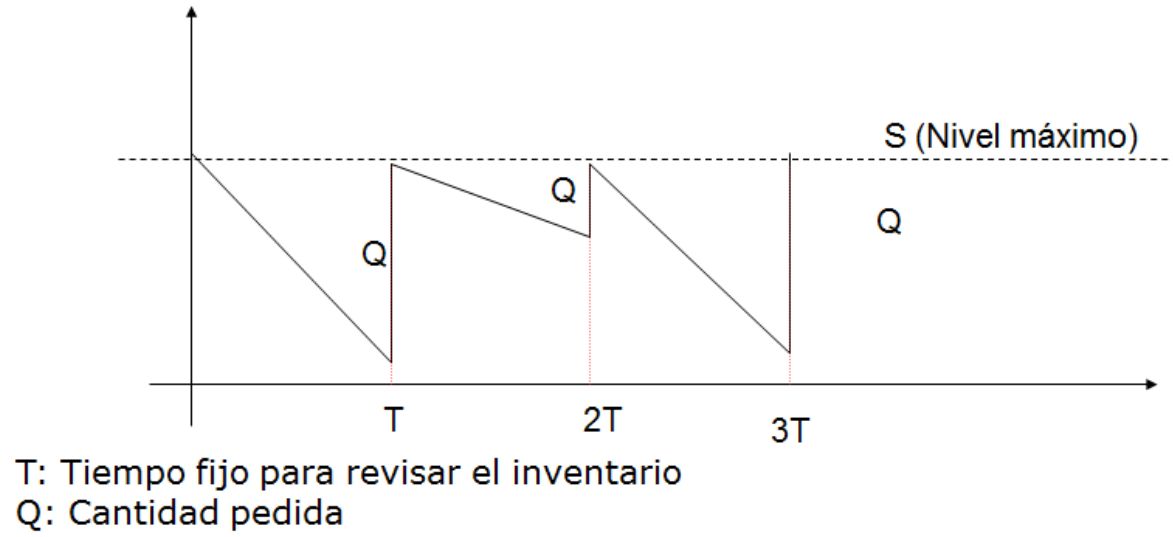

Figura 31: Sistema de revisión periódica

Sistema de suministro opcional

Cada T unidades de tiempo, revisa el inventario existente, si el inventario está en punto de reorden o por debajo de ese punto, se genera un pedido Q tal que el inventario sea elevado al nivel máximo. Si el inventario no ha llegado al punto de reorden no se genera ninguna orden. Se debe encontrar entonces, T, R y S.

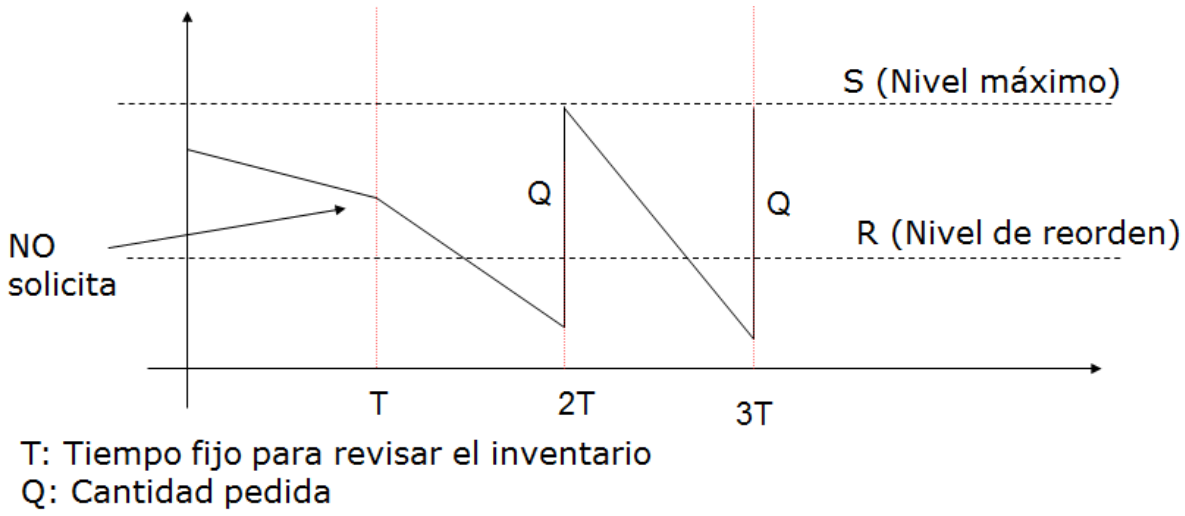

Figura 32: Sistema de suministro opcional

Ejemplo 1 de simulación de inventario

La demanda diaria de un producto está expresada en la siguiente tabla:

Demanda	**Probabilidad**
25	10%
26	20%
27	30%
28	30%
29	10%

El tiempo en días desde el pedido al proveedor y que éste haga la entrega, está expresada en la siguiente tabla:

Tiempo en días	**Probabilidad**
1	20%
2	30%
3	35%
4	10%
5	5%

Costo de ordenar: $200

Costo de inventario: $50/unidad por día

Costo de faltante (pérdida de imagen o goodwill): $120/unidad no vendida

En una política de sistema de Punto de Reorden (Sistema RQ) ¿Cuál es el óptimo R y Q?

Se empieza con un valor R=100 y Q=110 y un inventario inicial de 200 unidades.

Día	#Azar	Demanda de ese día	Inventario	Costo Inventario	Costo de Ordenar	#Azar	Llega Pedido	Cuenta atrás pedido	Costo GoodWill
0			200					-1	
1	0,902084688	29	171	$ 8.550,00	$ 0,00	-1,000	0	-1	
2	0,825244433	28	143	$ 7.150,00	$ 0,00	-1,000	0	-1	
3	0,963613785	29	114	$ 5.700,00	$ 0,00	-1,000	0	-1	
4	0,999112314	29	85	$ 4.250,00	$ 200,00	0,393	2	-1	
5	0,401594559	27	58	$ 2.900,00	$ 0,00	-1,000	0	2	
6	0,485172418	27	31	$ 1.550,00	$ 0,00	-1,000	0	1	
7	0,354153807	27	114	$ 5.700,00	$ 0,00	-1,000	0	0	
8	0,328321364	27	87	$ 4.350,00	$ 200,00	0,898	4	-1	
9	0,684177714	28	59	$ 2.950,00	$ 0,00	-1,000	0	4	
10	0,771624036	28	31	$ 1.550,00	$ 0,00	-1,000	0	3	
11	0,418056737	27	4	$ 200,00	$ 0,00	-1,000	0	2	
12	0,116846084	26	-22	$ 0,00	$ 0,00	-1,000	0	1	$ 2.640,0
13	0,449786448	27	83	$ 4.150,00	$ 200,00	0,349	2	0	
14	0,486076794	27	56	$ 2.800,00	$ 0,00	-1,000	0	2	
15	0,079714421	25	31	$ 1.550,00	$ 0,00	-1,000	0	1	
16	0,681874139	28	113	$ 5.650,00	$ 0,00	-1,000	0	0	
17	0,415244159	27	86	$ 4.300,00	$ 200,00	0,970	5	-1	
18	0,562321226	27	59	$ 2.950,00	$ 0,00	-1,000	0	5	
19	0,474514222	27	32	$ 1.600,00	$ 0,00	-1,000	0	4	
20	0,77082896	28	4	$ 200,00	$ 0,00	-1,000	0	3	
21	0,953051781	29	-25	$ 0,00	$ 0,00	-1,000	0	2	$ 3.000,0
22	0,143461696	26	-26	$ 0,00	$ 0,00	-1,000	0	1	$ 3.120,0
23	0,420843489	27	83	$ 4.150,00	$ 200,00	0,927	4	0	
24	0,953750653	29	54	$ 2.700,00	$ 0,00	-1,000	0	4	
25	0,381198731	27	27	$ 1.350,00	$ 0,00	-1,000	0	3	
26	0,301085762	27	0	$ 0,00	$ 0,00	-1,000	0	2	
27	0,627277762	28	-28	$ 0,00	$ 0,00	-1,000	0	1	$ 3.360,0
28	0,675102576	28	82	$ 4.100,00	$ 200,00	0,941	4	0	
29	0,995381946	29	53	$ 2.650,00	$ 0,00	-1,000	0	4	
30	0,470388351	27	26	$ 1.300,00	$ 0,00	-1,000	0	3	

En esa simulación de 30 días, se obtienen los siguientes valores:

Total costo de inventario: $84.300

Total costo de pedido: $1.200

Total costo de pérdida de GoodWill: $12.120

Total costo: $97.620

En la simulación anterior hay un problema importante: se simuló por muy poco tiempo (sólo 30 días) por lo que los datos están afectados por el valor del inventario inicial. Si se hubiese puesto un valor de inventario inicial muy bajo, el proceso de pedido y los costos de pérdida de imagen serían muy tempranos, en cambio, si se hubiese puesto un valor de inventario inicial muy alto, entonces no habría durante bastantes días la necesidad de hacer pedido y mucho menos costos por pérdida de imagen. El problema es que no se sabe de antemano cuanto debe ser el inventario inicial. Para resolver ese problema se debe simular por una cantidad inmensa de días (por lo menos 600 días) y tomar los últimos 200 días para generar métricas de costos. Al simular por tanto tiempo, hace que el valor inicial pierda toda relevancia.

Se repite entonces la simulación:

Día	Azar	Demanda ese día	Inventario	Valor Inventario	Pedido	Azar	Llega Pedido	Cuenta atrás llega ese pedido	Costo de pérdida de goodwill
801	0,142849517	26	116	$ 5.800,00	$ 0,00	-1,000	0	0	
802	0,60938385	28	88	$ 4.400,00	$ 200,00	0,677	3	-1	
803	0,269473243	26	62	$ 3.100,00	$ 0,00	-1,000	0	3	
804	0,824672033	28	34	$ 1.700,00	$ 0,00	-1,000	0	2	
805	0,836126144	28	6	$ 300,00	$ 0,00	-1,000	0	1	
806	0,913678941	29	87	$ 4.350,00	$ 200,00	0,713	3	0	
807	0,708196762	28	59	$ 2.950,00	$ 0,00	-1,000	0	3	
808	0,626411454	28	31	$ 1.550,00	$ 0,00	-1,000	0	2	
809	0,611556178	28	3	$ 150,00	$ 0,00	-1,000	0	1	
810	0,083665546	25	88	$ 4.400,00	$ 200,00	0,616	3	0	
811	0,341924391	27	61	$ 3.050,00	$ 0,00	-1,000	0	3	
812	0,35273422	27	34	$ 1.700,00	$ 0,00	-1,000	0	2	
813	0,749821492	28	6	$ 300,00	$ 0,00	-1,000	0	1	
814	0,30609012	27	89	$ 4.450,00	$ 200,00	0,725	3	0	
815	0,300426007	27	62	$ 3.100,00	$ 0,00	-1,000	0	3	
816	0,201892346	26	36	$ 1.800,00	$ 0,00	-1,000	0	2	
817	0,862909019	29	7	$ 350,00	$ 0,00	-1,000	0	1	
818	0,208594003	26	91	$ 4.550,00	$ 200,00	0,676	3	0	
819	0,719800392	28	63	$ 3.150,00	$ 0,00	-1,000	0	3	
820	0,226672355	26	37	$ 1.850,00	$ 0,00	-1,000	0	2	
821	0,652768352	28	9	$ 450,00	$ 0,00	-1,000	0	1	
822	0,176552748	26	93	$ 4.650,00	$ 200,00	0,244	2	0	
823	0,131079062	26	67	$ 3.350,00	$ 0,00	-1,000	0	2	
824	0,364042679	27	40	$ 2.000,00	$ 0,00	-1,000	0	1	
825	0,776334195	28	122	$ 6.100,00	$ 0,00	-1,000	0	0	
826	0,369063125	27	95	$ 4.750,00	$ 200,00	0,363	2	-1	
827	0,508278706	27	68	$ 3.400,00	$ 0,00	-1,000	0	2	
828	0,58374575	27	41	$ 2.050,00	$ 0,00	-1,000	0	1	
829	0,026490682	25	126	$ 6.300,00	$ 0,00	-1,000	0	0	
830	0,647098472	28	98	$ 4.900,00	$ 200,00	0,106	1	-1	
831	0,473840339	27	71	$ 3.550,00	$ 0,00	-1,000	0	1	
832	0,626888629	28	153	$ 7.650,00	$ 0,00	-1,000	0	0	
833	0,602832973	28	125	$ 6.250,00	$ 0,00	-1,000	0	-1	
834	0,273088151	26	99	$ 4.950,00	$ 200,00	0,038	1	-1	
835	0,988923101	29	70	$ 3.500,00	$ 0,00	-1,000	0	1	
836	0,343802184	27	153	$ 7.650,00	$ 0,00	-1,000	0	0	
837	0,818506569	28	125	$ 6.250,00	$ 0,00	-1,000	0	-1	
838	0,413119728	27	98	$ 4.900,00	$ 200,00	0,744	3	-1	
839	0,244828735	26	72	$ 3.600,00	$ 0,00	-1,000	0	3	
840	0,477436827	27	45	$ 2.250,00	$ 0,00	-1,000	0	2	
841	0,268140765	26	19	$ 950,00	$ 0,00	-1,000	0	1	
842	0,975010711	29	100	$ 5.000,00	$ 200,00	0,075	1	0	
843	0,901863847	29	71	$ 3.550,00	$ 0,00	-1,000	0	1	
844	0,395966343	27	154	$ 7.700,00	$ 0,00	-1,000	0	0	
845	0,627906618	28	126	$ 6.300,00	$ 0,00	-1,000	0	-1	
846	0,577247675	27	99	$ 4.950,00	$ 200,00	0,717	3	-1	
847	0,876048987	29	70	$ 3.500,00	$ 0,00	-1,000	0	3	
848	0,161712494	26	44	$ 2.200,00	$ 0,00	-1,000	0	2	
849	0,226266125	26	18	$ 900,00	$ 0,00	-1,000	0	1	
850	0,752828822	28	100	$ 5.000,00	$ 200,00	0,071	1	0	

851	0,37533666	27	73	$ 3.650,00	$ 0,00	-1,000	0	1	
852	0,441361739	27	156	$ 7.800,00	$ 0,00	-1,000	0	0	
853	0,793316081	28	128	$ 6.400,00	$ 0,00	-1,000	0	-1	
854	0,942470879	29	99	$ 4.950,00	$ 200,00	0,565	3	-1	
855	0,115650911	26	73	$ 3.650,00	$ 0,00	-1,000	0	3	
856	0,15113523	26	47	$ 2.350,00	$ 0,00	-1,000	0	2	
857	0,085263037	25	22	$ 1.100,00	$ 0,00	-1,000	0	1	
858	0,502436318	27	105	$ 5.250,00	$ 0,00	-1,000	0	0	
859	0,807354312	28	77	$ 3.850,00	$ 200,00	0,192	1	-1	
860	0,620673974	28	49	$ 2.450,00	$ 0,00	-1,000	0	1	
861	0,667070159	28	131	$ 6.550,00	$ 0,00	-1,000	0	0	
862	0,114272596	26	105	$ 5.250,00	$ 0,00	-1,000	0	-1	
863	0,321505293	27	78	$ 3.900,00	$ 200,00	0,216	2	-1	
864	0,386711213	27	51	$ 2.550,00	$ 0,00	-1,000	0	2	
865	0,798046992	28	23	$ 1.150,00	$ 0,00	-1,000	0	1	
866	0,206103989	26	107	$ 5.350,00	$ 0,00	-1,000	0	0	
867	0,786360155	28	79	$ 3.950,00	$ 200,00	0,324	2	-1	
868	0,17076735	26	53	$ 2.650,00	$ 0,00	-1,000	0	2	
869	0,230755395	26	27	$ 1.350,00	$ 0,00	-1,000	0	1	
870	0,970326542	29	108	$ 5.400,00	$ 0,00	-1,000	0	0	
871	0,788180498	28	80	$ 4.000,00	$ 200,00	0,704	3	-1	
872	0,603770688	28	52	$ 2.600,00	$ 0,00	-1,000	0	3	
873	0,871511936	29	23	$ 1.150,00	$ 0,00	-1,000	0	2	
874	0,91370531	29	-6	$ 0,00	$ 0,00	-1,000	0	1	$ 720,0
875	0,117414118	26	84	$ 4.200,00	$ 200,00	0,882	4	0	
876	0,978079772	29	55	$ 2.750,00	$ 0,00	-1,000	0	4	
877	0,08468076	25	30	$ 1.500,00	$ 0,00	-1,000	0	3	
878	0,765092746	28	2	$ 100,00	$ 0,00	-1,000	0	2	
879	0,52754725	27	-25	$ 0,00	$ 0,00	-1,000	0	1	$ 3.000,0
880	0,736495228	28	82	$ 4.100,00	$ 200,00	0,155	1	0	
881	0,66941463	28	54	$ 2.700,00	$ 0,00	-1,000	0	1	
882	0,472929959	27	137	$ 6.850,00	$ 0,00	-1,000	0	0	
883	0,813881881	28	109	$ 5.450,00	$ 0,00	-1,000	0	-1	
884	0,623995121	28	81	$ 4.050,00	$ 200,00	0,657	3	-1	
885	0,064277711	25	56	$ 2.800,00	$ 0,00	-1,000	0	3	
886	0,46947391	27	29	$ 1.450,00	$ 0,00	-1,000	0	2	
887	0,874358948	29	0	$ 0,00	$ 0,00	-1,000	0	1	
888	0,975498937	29	81	$ 4.050,00	$ 200,00	0,295	2	0	
889	0,790496653	28	53	$ 2.650,00	$ 0,00	-1,000	0	2	
890	0,374147078	27	26	$ 1.300,00	$ 0,00	-1,000	0	1	
891	0,971117863	29	107	$ 5.350,00	$ 0,00	-1,000	0	0	
892	0,565946801	27	80	$ 4.000,00	$ 200,00	0,748	3	-1	
893	0,571706989	27	53	$ 2.650,00	$ 0,00	-1,000	0	3	
894	0,758005861	28	25	$ 1.250,00	$ 0,00	-1,000	0	2	
895	0,019819909	25	0	$ 0,00	$ 0,00	-1,000	0	1	
896	0,082787452	25	85	$ 4.250,00	$ 200,00	0,264	2	0	
897	0,131398748	26	59	$ 2.950,00	$ 0,00	-1,000	0	2	
898	0,429092784	27	32	$ 1.600,00	$ 0,00	-1,000	0	1	
899	0,204302175	26	116	$ 5.800,00	$ 0,00	-1,000	0	0	
900	0,148972008	26	90	$ 4.500,00	$ 200,00	0,198	1	-1	
901	0,569723977	27	63	$ 3.150,00	$ 0,00	-1,000	0	1	
902	0,719563982	28	145	$ 7.250,00	$ 0,00	-1,000	0	0	
903	0,014687709	25	120	$ 6.000,00	$ 0,00	-1,000	0	-1	
904	0,550170032	27	93	$ 4.650,00	$ 200,00	0,584	3	-1	

905	0,832510738	28	65	$ 3.250,00	$ 0,00	-1,000	0	3	
906	0,917877064	29	36	$ 1.800,00	$ 0,00	-1,000	0	2	
907	0,461571465	27	9	$ 450,00	$ 0,00	-1,000	0	1	
908	0,464338658	27	92	$ 4.600,00	$ 200,00	0,685	3	0	
909	0,445748559	27	65	$ 3.250,00	$ 0,00	-1,000	0	3	
910	0,890554656	29	36	$ 1.800,00	$ 0,00	-1,000	0	2	
911	0,419082498	27	9	$ 450,00	$ 0,00	-1,000	0	1	
912	0,491298665	27	92	$ 4.600,00	$ 200,00	0,047	1	0	
913	0,314796162	27	65	$ 3.250,00	$ 0,00	-1,000	0	1	
914	0,547131798	27	148	$ 7.400,00	$ 0,00	-1,000	0	0	
915	0,783852137	28	120	$ 6.000,00	$ 0,00	-1,000	0	-1	
916	0,352194333	27	93	$ 4.650,00	$ 200,00	0,938	4	-1	
917	0,89475457	29	64	$ 3.200,00	$ 0,00	-1,000	0	4	
918	0,658739176	28	36	$ 1.800,00	$ 0,00	-1,000	0	3	
919	0,479831552	27	9	$ 450,00	$ 0,00	-1,000	0	2	
920	0,76375922	28	-19	$ 0,00	$ 0,00	-1,000	0	1	$ 2.280,0
921	0,197008924	26	84	$ 4.200,00	$ 200,00	0,507	3	0	
922	0,175502614	26	58	$ 2.900,00	$ 0,00	-1,000	0	3	
923	0,05015966	25	33	$ 1.650,00	$ 0,00	-1,000	0	2	
924	0,841249661	28	5	$ 250,00	$ 0,00	-1,000	0	1	
925	0,461012802	27	88	$ 4.400,00	$ 200,00	0,270	2	0	
926	0,778431396	28	60	$ 3.000,00	$ 0,00	-1,000	0	2	
927	0,962444711	29	31	$ 1.550,00	$ 0,00	-1,000	0	1	
928	0,809859723	28	113	$ 5.650,00	$ 0,00	-1,000	0	0	
929	0,546202852	27	86	$ 4.300,00	$ 200,00	0,235	2	-1	
930	0,22216969	26	60	$ 3.000,00	$ 0,00	-1,000	0	2	
931	0,635772147	28	32	$ 1.600,00	$ 0,00	-1,000	0	1	
932	0,540679693	27	115	$ 5.750,00	$ 0,00	-1,000	0	0	
933	0,852432819	29	86	$ 4.300,00	$ 200,00	0,934	4	-1	
934	0,583463061	27	59	$ 2.950,00	$ 0,00	-1,000	0	4	
935	0,296404936	26	33	$ 1.650,00	$ 0,00	-1,000	0	3	
936	0,475169486	27	6	$ 300,00	$ 0,00	-1,000	0	2	
937	0,829372543	28	-22	$ 0,00	$ 0,00	-1,000	0	1	$ 2.640,0
938	0,52752513	27	83	$ 4.150,00	$ 200,00	0,373	2	0	
939	0,63995141	28	55	$ 2.750,00	$ 0,00	-1,000	0	2	
940	0,963037453	29	26	$ 1.300,00	$ 0,00	-1,000	0	1	
941	0,260423749	26	110	$ 5.500,00	$ 0,00	-1,000	0	0	
942	0,166388119	26	84	$ 4.200,00	$ 200,00	0,376	2	-1	
943	0,76634915	28	56	$ 2.800,00	$ 0,00	-1,000	0	2	
944	0,489997228	27	29	$ 1.450,00	$ 0,00	-1,000	0	1	
945	0,563309028	27	112	$ 5.600,00	$ 0,00	-1,000	0	0	
946	0,792143395	28	84	$ 4.200,00	$ 200,00	0,146	1	-1	
947	0,47843578	27	57	$ 2.850,00	$ 0,00	-1,000	0	1	
948	0,382136736	27	140	$ 7.000,00	$ 0,00	-1,000	0	0	
949	0,891813254	29	111	$ 5.550,00	$ 0,00	-1,000	0	-1	
950	0,66562343	28	83	$ 4.150,00	$ 200,00	0,365	2	-1	
951	0,137103896	26	57	$ 2.850,00	$ 0,00	-1,000	0	2	
952	0,774958031	28	29	$ 1.450,00	$ 0,00	-1,000	0	1	
953	0,398247571	27	112	$ 5.600,00	$ 0,00	-1,000	0	0	
954	0,227092148	26	86	$ 4.300,00	$ 200,00	0,067	1	-1	
955	0,227919932	26	60	$ 3.000,00	$ 0,00	-1,000	0	1	
956	0,659163821	28	142	$ 7.100,00	$ 0,00	-1,000	0	0	
957	0,971185005	29	113	$ 5.650,00	$ 0,00	-1,000	0	-1	
958	0,570223879	27	86	$ 4.300,00	$ 200,00	0,938	4	-1	

959	0,157134105	26	60	$ 3.000,00	$ 0,00	-1,000	0	4	
960	0,371404306	27	33	$ 1.650,00	$ 0,00	-1,000	0	3	
961	0,485896897	27	6	$ 300,00	$ 0,00	-1,000	0	2	
962	0,561266293	27	-21	$ 0,00	$ 0,00	-1,000	0	1	$ 2.520,0
963	0,187012905	26	84	$ 4.200,00	$ 200,00	0,047	1	0	
964	0,804076203	28	56	$ 2.800,00	$ 0,00	-1,000	0	1	
965	0,303126993	27	139	$ 6.950,00	$ 0,00	-1,000	0	0	
966	0,513280691	27	112	$ 5.600,00	$ 0,00	-1,000	0	-1	
967	0,254385794	26	86	$ 4.300,00	$ 200,00	0,075	1	-1	
968	0,353536282	27	59	$ 2.950,00	$ 0,00	-1,000	0	1	
969	0,255030677	26	143	$ 7.150,00	$ 0,00	-1,000	0	0	
970	0,122483942	26	117	$ 5.850,00	$ 0,00	-1,000	0	-1	
971	0,316191725	27	90	$ 4.500,00	$ 200,00	0,970	5	-1	
972	0,49887889	27	63	$ 3.150,00	$ 0,00	-1,000	0	5	
973	0,687139927	28	35	$ 1.750,00	$ 0,00	-1,000	0	4	
974	0,089974441	25	10	$ 500,00	$ 0,00	-1,000	0	3	
975	0,881102693	29	-19	$ 0,00	$ 0,00	-1,000	0	2	$ 2.280,0
976	0,072835571	25	-25	$ 0,00	$ 0,00	-1,000	0	1	$ 3.000,0
977	0,073663044	25	85	$ 4.250,00	$ 200,00	0,609	3	0	
978	0,225501837	26	59	$ 2.950,00	$ 0,00	-1,000	0	3	
979	0,974981914	29	30	$ 1.500,00	$ 0,00	-1,000	0	2	
980	0,494707844	27	3	$ 150,00	$ 0,00	-1,000	0	1	
981	0,719592255	28	85	$ 4.250,00	$ 200,00	0,055	1	0	
982	0,693715437	28	57	$ 2.850,00	$ 0,00	-1,000	0	1	
983	0,321999689	27	140	$ 7.000,00	$ 0,00	-1,000	0	0	
984	0,089726425	25	115	$ 5.750,00	$ 0,00	-1,000	0	-1	
985	0,182254073	26	89	$ 4.450,00	$ 200,00	0,154	1	-1	
986	0,745568899	28	61	$ 3.050,00	$ 0,00	-1,000	0	1	
987	0,10330006	26	145	$ 7.250,00	$ 0,00	-1,000	0	0	
988	0,479095028	27	118	$ 5.900,00	$ 0,00	-1,000	0	-1	
989	0,893500542	29	89	$ 4.450,00	$ 200,00	0,268	2	-1	
990	0,206381577	26	63	$ 3.150,00	$ 0,00	-1,000	0	2	
991	0,935817947	29	34	$ 1.700,00	$ 0,00	-1,000	0	1	
992	0,362371293	27	117	$ 5.850,00	$ 0,00	-1,000	0	0	
993	0,932924451	29	88	$ 4.400,00	$ 200,00	0,182	1	-1	
994	0,987496942	29	59	$ 2.950,00	$ 0,00	-1,000	0	1	
995	0,83000261	28	141	$ 7.050,00	$ 0,00	-1,000	0	0	
996	0,387488181	27	114	$ 5.700,00	$ 0,00	-1,000	0	-1	
997	0,639862429	28	86	$ 4.300,00	$ 200,00	0,542	3	-1	
998	0,714571607	28	58	$ 2.900,00	$ 0,00	-1,000	0	3	
999	0,978864806	29	29	$ 1.450,00	$ 0,00	-1,000	0	2	
1000	0,146630102	26	3	$ 150,00	$ 0,00	-1,000	0	1	

En esa simulación de 1000 días, tomando los últimos 200 días, se obtienen los siguientes valores:

Total costo de inventario: $694.400

Total costo de pedido: $9.600

Total costo de pérdida de Goodwill: $16.440

Total costo: $720.440

Cambiando el R y Q por otros valores como R=180 y Q=200 se obtiene lo siguiente

Día	Azar	Demanda ese día	Inventario	Valor Inventario	Pedido	Azar	Llega Pedido	Cuenta atrás llega ese pedido	Costo de pérdida de goodwill
801	0,670271459	28	185	$ 9.250,00	$ 0,00	-1,000	0	-1	
802	0,678096206	28	157	$ 7.850,00	$ 200,00	0,679	3	-1	
803	0,347873168	27	130	$ 6.500,00	$ 0,00	-1,000	0	3	
804	0,516641164	27	103	$ 5.150,00	$ 0,00	-1,000	0	2	
805	0,641518836	28	75	$ 3.750,00	$ 0,00	-1,000	0	1	
806	0,802105795	28	247	$ 12.350,00	$ 0,00	-1,000	0	0	
807	0,921427049	29	218	$ 10.900,00	$ 0,00	-1,000	0	-1	
808	0,243481261	26	192	$ 9.600,00	$ 0,00	-1,000	0	-1	
809	0,426084622	27	165	$ 8.250,00	$ 200,00	0,470	2	-1	
810	0,622849729	28	137	$ 6.850,00	$ 0,00	-1,000	0	2	
811	0,254225696	26	111	$ 5.550,00	$ 0,00	-1,000	0	1	
812	0,359411464	27	284	$ 14.200,00	$ 0,00	-1,000	0	0	
813	0,029213764	25	259	$ 12.950,00	$ 0,00	-1,000	0	-1	
814	0,911547961	29	230	$ 11.500,00	$ 0,00	-1,000	0	-1	
815	0,347254894	27	203	$ 10.150,00	$ 0,00	-1,000	0	-1	
816	0,028581843	25	178	$ 8.900,00	$ 200,00	0,709	3	-1	
817	0,218174291	26	152	$ 7.600,00	$ 0,00	-1,000	0	3	
818	0,962237647	29	123	$ 6.150,00	$ 0,00	-1,000	0	2	
819	0,83427599	28	95	$ 4.750,00	$ 0,00	-1,000	0	1	
820	0,834562633	28	267	$ 13.350,00	$ 0,00	-1,000	0	0	
821	0,827429598	28	239	$ 11.950,00	$ 0,00	-1,000	0	-1	
822	0,102971136	26	213	$ 10.650,00	$ 0,00	-1,000	0	-1	
823	0,155008084	26	187	$ 9.350,00	$ 0,00	-1,000	0	-1	
824	0,750118141	28	159	$ 7.950,00	$ 200,00	0,065	1	-1	
825	0,276149746	26	133	$ 6.650,00	$ 0,00	-1,000	0	1	
826	0,03545437	25	308	$ 15.400,00	$ 0,00	-1,000	0	0	
827	0,209510946	26	282	$ 14.100,00	$ 0,00	-1,000	0	-1	
828	0,14577234	26	256	$ 12.800,00	$ 0,00	-1,000	0	-1	
829	0,251002519	26	230	$ 11.500,00	$ 0,00	-1,000	0	-1	
830	0,706524245	28	202	$ 10.100,00	$ 0,00	-1,000	0	-1	
831	0,741378686	28	174	$ 8.700,00	$ 200,00	0,640	3	-1	
832	0,246647106	26	148	$ 7.400,00	$ 0,00	-1,000	0	3	
833	0,616094951	28	120	$ 6.000,00	$ 0,00	-1,000	0	2	
834	0,980691487	29	91	$ 4.550,00	$ 0,00	-1,000	0	1	
835	0,055607295	25	266	$ 13.300,00	$ 0,00	-1,000	0	0	
836	0,30872165	27	239	$ 11.950,00	$ 0,00	-1,000	0	-1	
837	0,567909147	27	212	$ 10.600,00	$ 0,00	-1,000	0	-1	
838	0,605007348	28	184	$ 9.200,00	$ 0,00	-1,000	0	-1	
839	0,987726541	29	155	$ 7.750,00	$ 200,00	0,416	2	-1	
840	0,513757668	27	128	$ 6.400,00	$ 0,00	-1,000	0	2	
841	0,419995879	27	101	$ 5.050,00	$ 0,00	-1,000	0	1	
842	0,419678311	27	274	$ 13.700,00	$ 0,00	-1,000	0	0	
843	0,592061828	27	247	$ 12.350,00	$ 0,00	-1,000	0	-1	
844	0,573203073	27	220	$ 11.000,00	$ 0,00	-1,000	0	-1	
845	0,269554415	26	194	$ 9.700,00	$ 0,00	-1,000	0	-1	
846	0,552075982	27	167	$ 8.350,00	$ 200,00	0,289	2	-1	
847	0,175444064	26	141	$ 7.050,00	$ 0,00	-1,000	0	2	
848	0,791271377	28	113	$ 5.650,00	$ 0,00	-1,000	0	1	

849	0,718758809	28	285	$ 14.250,00	$ 0,00	-1,000	0	0	
850	0,362402069	27	258	$ 12.900,00	$ 0,00	-1,000	0	-1	
851	0,203535201	26	232	$ 11.600,00	$ 0,00	-1,000	0	-1	
852	0,104269424	26	206	$ 10.300,00	$ 0,00	-1,000	0	-1	
853	0,67518786	28	178	$ 8.900,00	$ 200,00	0,479	2	-1	
854	0,367678385	27	151	$ 7.550,00	$ 0,00	-1,000	0	2	
855	0,074762157	25	126	$ 6.300,00	$ 0,00	-1,000	0	1	
856	0,540422884	27	299	$ 14.950,00	$ 0,00	-1,000	0	0	
857	0,156314206	26	273	$ 13.650,00	$ 0,00	-1,000	0	-1	
858	0,449748745	27	246	$ 12.300,00	$ 0,00	-1,000	0	-1	
859	0,170358331	26	220	$ 11.000,00	$ 0,00	-1,000	0	-1	
860	0,052177658	25	195	$ 9.750,00	$ 0,00	-1,000	0	-1	
861	0,068990344	25	170	$ 8.500,00	$ 200,00	0,065	1	-1	
862	0,945078009	29	141	$ 7.050,00	$ 0,00	-1,000	0	1	
863	0,936911454	29	312	$ 15.600,00	$ 0,00	-1,000	0	0	
864	0,06286461	25	287	$ 14.350,00	$ 0,00	-1,000	0	-1	
865	0,016308715	25	262	$ 13.100,00	$ 0,00	-1,000	0	-1	
866	0,193947007	26	236	$ 11.800,00	$ 0,00	-1,000	0	-1	
867	0,932561947	29	207	$ 10.350,00	$ 0,00	-1,000	0	-1	
868	0,480076439	27	180	$ 9.000,00	$ 200,00	0,038	1	-1	
869	0,198989515	26	154	$ 7.700,00	$ 0,00	-1,000	0	1	
870	0,157740078	26	328	$ 16.400,00	$ 0,00	-1,000	0	0	
871	0,740390438	28	300	$ 15.000,00	$ 0,00	-1,000	0	-1	
872	0,755793719	28	272	$ 13.600,00	$ 0,00	-1,000	0	-1	
873	0,122136866	26	246	$ 12.300,00	$ 0,00	-1,000	0	-1	
874	0,813719984	28	218	$ 10.900,00	$ 0,00	-1,000	0	-1	
875	0,543195754	27	191	$ 9.550,00	$ 0,00	-1,000	0	-1	
876	0,361820935	27	164	$ 8.200,00	$ 200,00	0,444	2	-1	
877	0,952971222	29	135	$ 6.750,00	$ 0,00	-1,000	0	2	
878	0,007575131	25	110	$ 5.500,00	$ 0,00	-1,000	0	1	
879	0,359840362	27	283	$ 14.150,00	$ 0,00	-1,000	0	0	
880	0,344676958	27	256	$ 12.800,00	$ 0,00	-1,000	0	-1	
881	0,791575554	28	228	$ 11.400,00	$ 0,00	-1,000	0	-1	
882	0,550269898	27	201	$ 10.050,00	$ 0,00	-1,000	0	-1	
883	0,8931738	29	172	$ 8.600,00	$ 200,00	0,673	3	-1	
884	0,10388089	26	146	$ 7.300,00	$ 0,00	-1,000	0	3	
885	0,507196912	27	119	$ 5.950,00	$ 0,00	-1,000	0	2	
886	0,618716854	28	91	$ 4.550,00	$ 0,00	-1,000	0	1	
887	0,485091363	27	264	$ 13.200,00	$ 0,00	-1,000	0	0	
888	0,950212046	29	235	$ 11.750,00	$ 0,00	-1,000	0	-1	
889	0,938804077	29	206	$ 10.300,00	$ 0,00	-1,000	0	-1	
890	0,14340788	26	180	$ 9.000,00	$ 200,00	0,851	4	-1	
891	0,654242708	28	152	$ 7.600,00	$ 0,00	-1,000	0	4	
892	0,335403218	27	125	$ 6.250,00	$ 0,00	-1,000	0	3	
893	0,5739478	27	98	$ 4.900,00	$ 0,00	-1,000	0	2	
894	0,541514742	27	71	$ 3.550,00	$ 0,00	-1,000	0	1	
895	0,771578733	28	243	$ 12.150,00	$ 0,00	-1,000	0	0	
896	0,414554701	27	216	$ 10.800,00	$ 0,00	-1,000	0	-1	
897	0,27342179	26	190	$ 9.500,00	$ 0,00	-1,000	0	-1	
898	0,383004972	27	163	$ 8.150,00	$ 200,00	0,367	2	-1	
899	0,4591243	27	136	$ 6.800,00	$ 0,00	-1,000	0	2	
900	0,622797279	28	108	$ 5.400,00	$ 0,00	-1,000	0	1	
901	0,438850852	27	281	$ 14.050,00	$ 0,00	-1,000	0	0	
902	0,994570265	29	252	$ 12.600,00	$ 0,00	-1,000	0	-1	

903	0,850841125	29	223	$ 11.150,00	$ 0,00	-1,000	0	-1	
904	0,532620702	27	196	$ 9.800,00	$ 0,00	-1,000	0	-1	
905	0,088359265	25	171	$ 8.550,00	$ 200,00	0,897	4	-1	
906	0,707594904	28	143	$ 7.150,00	$ 0,00	-1,000	0	4	
907	0,336181737	27	116	$ 5.800,00	$ 0,00	-1,000	0	3	
908	0,067219147	25	91	$ 4.550,00	$ 0,00	-1,000	0	2	
909	0,1937371	26	65	$ 3.250,00	$ 0,00	-1,000	0	1	
910	0,468516832	27	238	$ 11.900,00	$ 0,00	-1,000	0	0	
911	0,102275098	26	212	$ 10.600,00	$ 0,00	-1,000	0	-1	
912	0,276872962	26	186	$ 9.300,00	$ 0,00	-1,000	0	-1	
913	0,604495834	28	158	$ 7.900,00	$ 200,00	0,370	2	-1	
914	0,216296637	26	132	$ 6.600,00	$ 0,00	-1,000	0	2	
915	0,83037614	28	104	$ 5.200,00	$ 0,00	-1,000	0	1	
916	0,779029356	28	276	$ 13.800,00	$ 0,00	-1,000	0	0	
917	0,020088267	25	251	$ 12.550,00	$ 0,00	-1,000	0	-1	
918	0,755451984	28	223	$ 11.150,00	$ 0,00	-1,000	0	-1	
919	0,430956625	27	196	$ 9.800,00	$ 0,00	-1,000	0	-1	
920	0,717889039	28	168	$ 8.400,00	$ 200,00	0,173	1	-1	
921	0,747604104	28	140	$ 7.000,00	$ 0,00	-1,000	0	1	
922	0,636354218	28	312	$ 15.600,00	$ 0,00	-1,000	0	0	
923	0,078452052	25	287	$ 14.350,00	$ 0,00	-1,000	0	-1	
924	0,600836502	28	259	$ 12.950,00	$ 0,00	-1,000	0	-1	
925	0,792348652	28	231	$ 11.550,00	$ 0,00	-1,000	0	-1	
926	0,348716929	27	204	$ 10.200,00	$ 0,00	-1,000	0	-1	
927	0,298486778	26	178	$ 8.900,00	$ 200,00	0,751	3	-1	
928	0,765469088	28	150	$ 7.500,00	$ 0,00	-1,000	0	3	
929	0,395518835	27	123	$ 6.150,00	$ 0,00	-1,000	0	2	
930	0,74188265	28	95	$ 4.750,00	$ 0,00	-1,000	0	1	
931	0,703448466	28	267	$ 13.350,00	$ 0,00	-1,000	0	0	
932	0,842170588	28	239	$ 11.950,00	$ 0,00	-1,000	0	-1	
933	0,718561579	28	211	$ 10.550,00	$ 0,00	-1,000	0	-1	
934	0,8795626	29	182	$ 9.100,00	$ 0,00	-1,000	0	-1	
935	0,960731242	29	153	$ 7.650,00	$ 200,00	0,608	3	-1	
936	0,690806912	28	125	$ 6.250,00	$ 0,00	-1,000	0	3	
937	0,157666935	26	99	$ 4.950,00	$ 0,00	-1,000	0	2	
938	0,112435651	26	73	$ 3.650,00	$ 0,00	-1,000	0	1	
939	0,606690346	28	245	$ 12.250,00	$ 0,00	-1,000	0	0	
940	0,283582834	26	219	$ 10.950,00	$ 0,00	-1,000	0	-1	
941	0,723261828	28	191	$ 9.550,00	$ 0,00	-1,000	0	-1	
942	0,924319403	29	162	$ 8.100,00	$ 200,00	0,993	5	-1	
943	0,272309301	26	136	$ 6.800,00	$ 0,00	-1,000	0	5	
944	0,18308056	26	110	$ 5.500,00	$ 0,00	-1,000	0	4	
945	0,928497888	29	81	$ 4.050,00	$ 0,00	-1,000	0	3	
946	0,539786917	27	54	$ 2.700,00	$ 0,00	-1,000	0	2	
947	0,442052007	27	27	$ 1.350,00	$ 0,00	-1,000	0	1	
948	0,369407664	27	200	$ 10.000,00	$ 0,00	-1,000	0	0	
949	0,385917413	27	173	$ 8.650,00	$ 200,00	0,791	3	-1	
950	0,04922286	25	148	$ 7.400,00	$ 0,00	-1,000	0	3	
951	0,445291524	27	121	$ 6.050,00	$ 0,00	-1,000	0	2	
952	0,478882879	27	94	$ 4.700,00	$ 0,00	-1,000	0	1	
953	0,125837083	26	268	$ 13.400,00	$ 0,00	-1,000	0	0	
954	0,977758104	29	239	$ 11.950,00	$ 0,00	-1,000	0	-1	
955	0,263541087	26	213	$ 10.650,00	$ 0,00	-1,000	0	-1	
956	0,541922153	27	186	$ 9.300,00	$ 0,00	-1,000	0	-1	

957	0,461126143	27	159	$ 7.950,00	$ 200,00	0,550	3	-1	
958	0,50227489	27	132	$ 6.600,00	$ 0,00	-1,000	0	3	
959	0,415950521	27	105	$ 5.250,00	$ 0,00	-1,000	0	2	
960	0,488653698	27	78	$ 3.900,00	$ 0,00	-1,000	0	1	
961	0,046850466	25	253	$ 12.650,00	$ 0,00	-1,000	0	0	
962	0,3203018	27	226	$ 11.300,00	$ 0,00	-1,000	0	-1	
963	0,108547652	26	200	$ 10.000,00	$ 0,00	-1,000	0	-1	
964	0,572993347	27	173	$ 8.650,00	$ 200,00	0,163	1	-1	
965	0,121925183	26	147	$ 7.350,00	$ 0,00	-1,000	0	1	
966	0,058093116	25	322	$ 16.100,00	$ 0,00	-1,000	0	0	
967	0,035886528	25	297	$ 14.850,00	$ 0,00	-1,000	0	-1	
968	0,811456336	28	269	$ 13.450,00	$ 0,00	-1,000	0	-1	
969	0,060575455	25	244	$ 12.200,00	$ 0,00	-1,000	0	-1	
970	0,691200221	28	216	$ 10.800,00	$ 0,00	-1,000	0	-1	
971	0,644355977	28	188	$ 9.400,00	$ 0,00	-1,000	0	-1	
972	0,813594614	28	160	$ 8.000,00	$ 200,00	0,317	2	-1	
973	0,427009073	27	133	$ 6.650,00	$ 0,00	-1,000	0	2	
974	0,90905807	29	104	$ 5.200,00	$ 0,00	-1,000	0	1	
975	0,363623767	27	277	$ 13.850,00	$ 0,00	-1,000	0	0	
976	0,63597157	28	249	$ 12.450,00	$ 0,00	-1,000	0	-1	
977	0,604669468	28	221	$ 11.050,00	$ 0,00	-1,000	0	-1	
978	0,671362256	28	193	$ 9.650,00	$ 0,00	-1,000	0	-1	
979	0,690434278	28	165	$ 8.250,00	$ 200,00	0,343	2	-1	
980	0,433977121	27	138	$ 6.900,00	$ 0,00	-1,000	0	2	
981	0,993365737	29	109	$ 5.450,00	$ 0,00	-1,000	0	1	
982	0,205231345	26	283	$ 14.150,00	$ 0,00	-1,000	0	0	
983	0,230986307	26	257	$ 12.850,00	$ 0,00	-1,000	0	-1	
984	0,321017469	27	230	$ 11.500,00	$ 0,00	-1,000	0	-1	
985	0,558104882	27	203	$ 10.150,00	$ 0,00	-1,000	0	-1	
986	0,431291628	27	176	$ 8.800,00	$ 200,00	0,115	1	-1	
987	0,957452921	29	147	$ 7.350,00	$ 0,00	-1,000	0	1	
988	0,838561516	28	319	$ 15.950,00	$ 0,00	-1,000	0	0	
989	0,755946181	28	291	$ 14.550,00	$ 0,00	-1,000	0	-1	
990	0,978937424	29	262	$ 13.100,00	$ 0,00	-1,000	0	-1	
991	0,710262737	28	234	$ 11.700,00	$ 0,00	-1,000	0	-1	
992	0,856473843	29	205	$ 10.250,00	$ 0,00	-1,000	0	-1	
993	0,300924895	27	178	$ 8.900,00	$ 200,00	0,642	3	-1	
994	0,172760254	26	152	$ 7.600,00	$ 0,00	-1,000	0	3	
995	0,566213656	27	125	$ 6.250,00	$ 0,00	-1,000	0	2	
996	0,83988748	28	97	$ 4.850,00	$ 0,00	-1,000	0	1	
997	0,69018984	28	269	$ 13.450,00	$ 0,00	-1,000	0	0	
998	0,916855846	29	240	$ 12.000,00	$ 0,00	-1,000	0	-1	
999	0,466582863	27	213	$ 10.650,00	$ 0,00	-1,000	0	-1	
1000	0,380337769	27	186	$ 9.300,00	$ 0,00	-1,000	0	-1	

En esa simulación de 1000 días, tomando los últimos 200 días, se obtiene los siguientes valores:

Total costo de inventario: $1.903.050

Total costo de pedido: $5.400

Total costo de pérdida de Goodwill: $0

Total costo: $1.908.450

Como se puede observar, en la segunda simulación, no hay pérdida de goodwill pero aumentó considerablemente el costo de inventario.

Se busca entonces un valor Q y R que finalmente generen el menor costo.

Consideraciones sobre la simulación de procesos de inventario

1. Un generador pseudo-aleatorio que haya pasado las pruebas de independencia y uniformidad tiene una ventaja sobre un generador completamente aleatorio: Es capaz de repetir la secuencia de números aleatorios cada vez que se desee.

2. Repetir una secuencia de números aleatorios ofrece la ventaja que cualquier cambio que se le haga al modelo planteado en la simulación (en inventario con el sistema RQ en el que se varía R y Q) mostrará rápida y claramente las diferencias de los resultados entre uno y otro valor. Lograr el mismo efecto con un generador totalmente aleatorio, requiere almacenar los números aleatorios en algún tipo de estructura (sea arreglos unidimensionales o listas simplemente enlazadas) con el consumo de memoria que esto conlleva.

3. El inventario inicial es un valor necesario, pero afectó la simulación en los primeros días. Si arranca con un inventario inicial muy alto, los costos de almacenamiento se elevarán aunque se evita el proceso de pedidos y espera. Un inventario inicial muy bajo o cero, involucra pedidos desde el inicio de la simulación e involucra costos de goodwill.

4. Para evitar la influencia del valor inicial (inventario inicial), la simulación debe ejecutarse por muchos días (1000 o 2000) y tomar decisiones con los últimos grupos de valores. Se desecha los primeros días. Toda simulación debe esperar hasta que se estabilice.

5. Utilizar un generador de números pseudo-aleatorios distinto para cada variable aleatoria usada, de esa manera lo expuesto en el punto 2 puede llevarse a cabo.

6. Los inventarios pueden llegar a manejar muchas variables aleatorias mas, por ejemplo, el cliente puede dar una espera a que la mercancía llegue, o que los costos de inventario cambien dependiendo del número de unidades que estén almacenadas (es más lento y costoso retirar una unidad de un elemento de un contenedor completamente lleno), los costos del pedido pueden variar (consecuencia de oferta-demanda, inflación, cambio de divisas), etc. Los modelos tienden a simplificarse para poder tomar decisiones.

7. Al desarrollar un software que simule inventarios, se debe comunicar claramente qué modelo de inventario se va a seguir. La entrada de los datos debe ser fácil.

8. Un software con el modelo RQ (Sistema de Punto de Reorden) debe permitir cambiar al usuario continuamente los valores Q y R pero no el resto para poder ver cómo se comporta ante cada cambio de Q y R. Luego debe mostrar un informe simple consolidado mostrando los costos finales para cada Q y R escogido por el usuario.

9. Si el usuario quiere cambiar todos los datos, entonces se reinicia toda la simulación y se borra la memoria de las anteriores simulaciones.

10. Se debe recordar que las variables aleatorias no necesariamente tienen una distribución empírica también pueden tener otras distribuciones.

11. Mientras un pedido del proveedor esté activo (todavía falta tiempo para la entrega), no se pueden hacer más pedidos.

Ejemplo 2 de simulación de inventario

En un sistema de inventario T pero añadiendo la variable aleatoria de tiempo de espera del proveedor para entregar los productos, se tienen los siguientes datos:

Inventario Inicial	700
Costo Ordenar	$ 200
Costo Inventario	$ 10
Costo Faltante	$ 150
Costo Revisar Fijo	$ 300

Las variables aleatorias son las siguientes:

a. Demanda diaria de producto, con una distribución normal de media = 60 y desviación = 7

b. Tiempo en días que tarda el proveedor en entregar el pedido desde que se hace la solicitud, con una distribución uniforme entre 4 y 8 días.

c. Daños a productos dentro del inventario con una distribución exponencial con alpha = 0,7

Se simula por 1000 días y se toman valores con los últimos 200 días.

Empieza con un S=500 (tope máximo del inventario) y un T=2 (cada dos días se hace revisión del inventario).

Día	Demanda	Daños	Llega	Inventario	Costo Inventario	Costo Revisar Fijo	Costo Ordenar	Tiempo en llegar	Costo Goodwill
801	63	0	0	157	$ 1.570	$ 0	$ 0	-1	$ 0
802	62	1	0	94	$ 940	$ 300	$ 200	5	$ 0
803	60	1	0	33	$ 330	$ 0	$ 0	4	$ 0
804	58	2	0	0	$ 0	$ 0	$ 0	3	$ 3.750
805	64	4	0	0	$ 0	$ 0	$ 0	2	$ 9.600
806	44	1	0	0	$ 0	$ 0	$ 0	1	$ 6.600
807	57	1	406	348	$ 3.480	$ 0	$ 0	0	$ 8.550
808	66	1	0	281	$ 2.810	$ 300	$ 200	7	$ 0
809	61	0	0	220	$ 2.200	$ 0	$ 0	6	$ 0
810	65	1	0	154	$ 1.540	$ 0	$ 0	5	$ 0
811	59	0	0	95	$ 950	$ 0	$ 0	4	$ 0
812	61	0	0	34	$ 340	$ 0	$ 0	3	$ 0
813	49	1	0	0	$ 0	$ 0	$ 0	2	$ 2.250
814	54	2	0	0	$ 0	$ 0	$ 0	1	$ 8.100
815	69	1	219	149	$ 1.490	$ 0	$ 0	0	$ 10.350
816	61	1	0	87	$ 870	$ 300	$ 200	5	$ 0
817	58	0	0	29	$ 290	$ 0	$ 0	4	$ 0
818	67	1	0	0	$ 0	$ 0	$ 0	3	$ 5.700
819	57	1	0	0	$ 0	$ 0	$ 0	2	$ 8.550
820	67	1	0	0	$ 0	$ 0	$ 0	1	$ 10.050
821	69	2	413	342	$ 3.420	$ 0	$ 0	0	$ 10.350
822	60	1	0	281	$ 2.810	$ 300	$ 200	5	$ 0
823	68	0	0	213	$ 2.130	$ 0	$ 0	4	$ 0
824	51	1	0	161	$ 1.610	$ 0	$ 0	3	$ 0
825	52	0	0	109	$ 1.090	$ 0	$ 0	2	$ 0
826	64	1	0	44	$ 440	$ 0	$ 0	1	$ 0
827	69	1	219	193	$ 1.930	$ 0	$ 0	0	$ 3.750
828	64	2	0	127	$ 1.270	$ 300	$ 200	6	$ 0
829	54	0	0	73	$ 730	$ 0	$ 0	5	$ 0
830	61	1	0	11	$ 110	$ 0	$ 0	4	$ 0
831	60	2	0	0	$ 0	$ 0	$ 0	3	$ 7.350
832	47	1	0	0	$ 0	$ 0	$ 0	2	$ 7.050
833	69	1	0	0	$ 0	$ 0	$ 0	1	$ 10.350
834	62	1	373	310	$ 3.100	$ 0	$ 0	0	$ 9.300
835	54	0	0	256	$ 2.560	$ 0	$ 0	-1	$ 0
836	56	3	0	197	$ 1.970	$ 300	$ 200	8	$ 0
837	63	1	0	133	$ 1.330	$ 0	$ 0	7	$ 0
838	53	4	0	76	$ 760	$ 0	$ 0	6	$ 0
839	53	4	0	19	$ 190	$ 0	$ 0	5	$ 0
840	70	1	0	0	$ 0	$ 0	$ 0	4	$ 7.650
841	53	0	0	0	$ 0	$ 0	$ 0	3	$ 7.950
842	51	0	0	0	$ 0	$ 0	$ 0	2	$ 7.650
843	54	0	0	0	$ 0	$ 0	$ 0	1	$ 8.100
844	60	2	303	241	$ 2.410	$ 0	$ 0	0	$ 9.000
845	69	1	0	171	$ 1.710	$ 0	$ 0	-1	$ 0
846	69	3	0	99	$ 990	$ 300	$ 200	5	$ 0
847	56	0	0	43	$ 430	$ 0	$ 0	4	$ 0
848	56	0	0	0	$ 0	$ 0	$ 0	3	$ 1.950
849	54	1	0	0	$ 0	$ 0	$ 0	2	$ 8.100
850	62	1	0	0	$ 0	$ 0	$ 0	1	$ 9.300
851	60	0	401	341	$ 3.410	$ 0	$ 0	0	$ 9.000

852	60	0	0	281	$ 2.810	$ 300	$ 200	7	$ 0
853	60	0	0	221	$ 2.210	$ 0	$ 0	6	$ 0
854	63	0	0	158	$ 1.580	$ 0	$ 0	5	$ 0
855	59	6	0	93	$ 930	$ 0	$ 0	4	$ 0
856	55	2	0	36	$ 360	$ 0	$ 0	3	$ 0
857	74	2	0	0	$ 0	$ 0	$ 0	2	$ 5.700
858	62	2	0	0	$ 0	$ 0	$ 0	1	$ 9.300
859	62	3	219	154	$ 1.540	$ 0	$ 0	0	$ 9.300
860	56	0	0	98	$ 980	$ 300	$ 200	6	$ 0
861	59	4	0	35	$ 350	$ 0	$ 0	5	$ 0
862	60	3	0	0	$ 0	$ 0	$ 0	4	$ 3.750
863	53	1	0	0	$ 0	$ 0	$ 0	3	$ 7.950
864	54	5	0	0	$ 0	$ 0	$ 0	2	$ 8.100
865	49	1	0	0	$ 0	$ 0	$ 0	1	$ 7.350
866	59	0	402	343	$ 3.430	$ 0	$ 0	0	$ 8.850
867	52	5	0	286	$ 2.860	$ 0	$ 0	-1	$ 0
868	71	0	0	215	$ 2.150	$ 300	$ 200	5	$ 0
869	59	0	0	156	$ 1.560	$ 0	$ 0	4	$ 0
870	52	1	0	103	$ 1.030	$ 0	$ 0	3	$ 0
871	64	2	0	37	$ 370	$ 0	$ 0	2	$ 0
872	55	2	0	0	$ 0	$ 0	$ 0	1	$ 2.700
873	58	1	285	226	$ 2.260	$ 0	$ 0	0	$ 8.700
874	59	0	0	167	$ 1.670	$ 300	$ 200	6	$ 0
875	64	1	0	102	$ 1.020	$ 0	$ 0	5	$ 0
876	75	1	0	26	$ 260	$ 0	$ 0	4	$ 0
877	44	1	0	0	$ 0	$ 0	$ 0	3	$ 2.700
878	54	4	0	0	$ 0	$ 0	$ 0	2	$ 8.100
879	74	0	0	0	$ 0	$ 0	$ 0	1	$ 11.100
880	56	3	333	274	$ 2.740	$ 0	$ 0	0	$ 8.400
881	50	0	0	224	$ 2.240	$ 0	$ 0	-1	$ 0
882	53	0	0	171	$ 1.710	$ 300	$ 200	7	$ 0
883	60	2	0	109	$ 1.090	$ 0	$ 0	6	$ 0
884	48	2	0	59	$ 590	$ 0	$ 0	5	$ 0
885	67	5	0	0	$ 0	$ 0	$ 0	4	$ 1.200
886	64	4	0	0	$ 0	$ 0	$ 0	3	$ 9.600
887	60	2	0	0	$ 0	$ 0	$ 0	2	$ 9.000
888	51	0	0	0	$ 0	$ 0	$ 0	1	$ 7.650
889	57	3	329	269	$ 2.690	$ 0	$ 0	0	$ 8.550
890	58	7	0	204	$ 2.040	$ 300	$ 200	5	$ 0
891	61	2	0	141	$ 1.410	$ 0	$ 0	4	$ 0
892	76	1	0	64	$ 640	$ 0	$ 0	3	$ 0
893	70	1	0	0	$ 0	$ 0	$ 0	2	$ 900
894	68	0	0	0	$ 0	$ 0	$ 0	1	$ 10.200
895	54	1	296	241	$ 2.410	$ 0	$ 0	0	$ 8.100
896	67	1	0	173	$ 1.730	$ 300	$ 200	6	$ 0
897	70	2	0	101	$ 1.010	$ 0	$ 0	5	$ 0
898	48	2	0	51	$ 510	$ 0	$ 0	4	$ 0
899	57	0	0	0	$ 0	$ 0	$ 0	3	$ 900
900	65	0	0	0	$ 0	$ 0	$ 0	2	$ 9.750
901	55	2	0	0	$ 0	$ 0	$ 0	1	$ 8.250
902	50	1	327	276	$ 2.760	$ 0	$ 0	0	$ 7.500
903	53	0	0	223	$ 2.230	$ 0	$ 0	-1	$ 0
904	67	0	0	156	$ 1.560	$ 300	$ 200	4	$ 0
905	50	2	0	104	$ 1.040	$ 0	$ 0	3	$ 0

906	55	2	0	47	$ 470	$ 0	$ 0	2	$ 0
907	64	2	0	0	$ 0	$ 0	$ 0	1	$ 2.550
908	56	0	344	288	$ 2.880	$ 0	$ 0	0	$ 8.400
909	55	1	0	232	$ 2.320	$ 0	$ 0	-1	$ 0
910	63	1	0	168	$ 1.680	$ 300	$ 200	7	$ 0
911	50	1	0	117	$ 1.170	$ 0	$ 0	6	$ 0
912	57	4	0	56	$ 560	$ 0	$ 0	5	$ 0
913	61	0	0	0	$ 0	$ 0	$ 0	4	$ 750
914	61	2	0	0	$ 0	$ 0	$ 0	3	$ 9.150
915	63	4	0	0	$ 0	$ 0	$ 0	2	$ 9.450
916	64	2	0	0	$ 0	$ 0	$ 0	1	$ 9.600
917	55	4	332	273	$ 2.730	$ 0	$ 0	0	$ 8.250
918	60	8	0	205	$ 2.050	$ 300	$ 200	7	$ 0
919	58	1	0	146	$ 1.460	$ 0	$ 0	6	$ 0
920	60	2	0	84	$ 840	$ 0	$ 0	5	$ 0
921	60	1	0	23	$ 230	$ 0	$ 0	4	$ 0
922	67	1	0	0	$ 0	$ 0	$ 0	3	$ 6.600
923	70	1	0	0	$ 0	$ 0	$ 0	2	$ 10.500
924	56	1	0	0	$ 0	$ 0	$ 0	1	$ 8.400
925	63	2	295	230	$ 2.300	$ 0	$ 0	0	$ 9.450
926	55	3	0	172	$ 1.720	$ 300	$ 200	5	$ 0
927	64	2	0	106	$ 1.060	$ 0	$ 0	4	$ 0
928	46	1	0	59	$ 590	$ 0	$ 0	3	$ 0
929	60	1	0	0	$ 0	$ 0	$ 0	2	$ 150
930	61	2	0	0	$ 0	$ 0	$ 0	1	$ 9.150
931	66	1	328	261	$ 2.610	$ 0	$ 0	0	$ 9.900
932	69	0	0	192	$ 1.920	$ 300	$ 200	6	$ 0
933	52	1	0	139	$ 1.390	$ 0	$ 0	5	$ 0
934	54	0	0	85	$ 850	$ 0	$ 0	4	$ 0
935	58	0	0	27	$ 270	$ 0	$ 0	3	$ 0
936	61	2	0	0	$ 0	$ 0	$ 0	2	$ 5.100
937	58	2	0	0	$ 0	$ 0	$ 0	1	$ 8.700
938	50	3	308	255	$ 2.550	$ 0	$ 0	0	$ 7.500
939	68	0	0	187	$ 1.870	$ 0	$ 0	-1	$ 0
940	57	3	0	127	$ 1.270	$ 300	$ 200	5	$ 0
941	55	0	0	72	$ 720	$ 0	$ 0	4	$ 0
942	62	0	0	10	$ 100	$ 0	$ 0	3	$ 0
943	61	4	0	0	$ 0	$ 0	$ 0	2	$ 7.650
944	63	1	0	0	$ 0	$ 0	$ 0	1	$ 9.450
945	65	5	373	303	$ 3.030	$ 0	$ 0	0	$ 9.750
946	58	1	0	244	$ 2.440	$ 300	$ 200	4	$ 0
947	56	1	0	187	$ 1.870	$ 0	$ 0	3	$ 0
948	60	1	0	126	$ 1.260	$ 0	$ 0	2	$ 0
949	59	0	0	67	$ 670	$ 0	$ 0	1	$ 0
950	54	2	256	267	$ 2.670	$ 0	$ 0	0	$ 0
951	71	1	0	195	$ 1.950	$ 0	$ 0	-1	$ 0
952	51	1	0	143	$ 1.430	$ 300	$ 200	6	$ 0
953	69	3	0	71	$ 710	$ 0	$ 0	5	$ 0
954	54	1	0	16	$ 160	$ 0	$ 0	4	$ 0
955	61	1	0	0	$ 0	$ 0	$ 0	3	$ 6.750
956	49	0	0	0	$ 0	$ 0	$ 0	2	$ 7.350
957	69	1	0	0	$ 0	$ 0	$ 0	1	$ 10.350
958	69	2	357	286	$ 2.860	$ 0	$ 0	0	$ 10.350
959	67	0	0	219	$ 2.190	$ 0	$ 0	-1	$ 0

960	57	1	0	161	$ 1.610	$ 300	$ 200	6	$ 0
961	50	1	0	110	$ 1.100	$ 0	$ 0	5	$ 0
962	52	2	0	56	$ 560	$ 0	$ 0	4	$ 0
963	52	2	0	2	$ 20	$ 0	$ 0	3	$ 0
964	62	2	0	0	$ 0	$ 0	$ 0	2	$ 9.000
965	63	2	0	0	$ 0	$ 0	$ 0	1	$ 9.450
966	51	2	339	286	$ 2.860	$ 0	$ 0	0	$ 7.650
967	71	4	0	211	$ 2.110	$ 0	$ 0	-1	$ 0
968	71	4	0	136	$ 1.360	$ 300	$ 200	7	$ 0
969	61	1	0	74	$ 740	$ 0	$ 0	6	$ 0
970	68	2	0	4	$ 40	$ 0	$ 0	5	$ 0
971	59	1	0	0	$ 0	$ 0	$ 0	4	$ 8.250
972	57	3	0	0	$ 0	$ 0	$ 0	3	$ 8.550
973	58	6	0	0	$ 0	$ 0	$ 0	2	$ 8.700
974	57	1	0	0	$ 0	$ 0	$ 0	1	$ 8.550
975	59	3	364	302	$ 3.020	$ 0	$ 0	0	$ 8.850
976	50	3	0	249	$ 2.490	$ 300	$ 200	4	$ 0
977	61	0	0	188	$ 1.880	$ 0	$ 0	3	$ 0
978	64	1	0	123	$ 1.230	$ 0	$ 0	2	$ 0
979	58	0	0	65	$ 650	$ 0	$ 0	1	$ 0
980	64	1	251	251	$ 2.510	$ 0	$ 0	0	$ 0
981	53	0	0	198	$ 1.980	$ 0	$ 0	-1	$ 0
982	61	0	0	137	$ 1.370	$ 300	$ 200	6	$ 0
983	59	0	0	78	$ 780	$ 0	$ 0	5	$ 0
984	60	0	0	18	$ 180	$ 0	$ 0	4	$ 0
985	59	0	0	0	$ 0	$ 0	$ 0	3	$ 6.150
986	56	2	0	0	$ 0	$ 0	$ 0	2	$ 8.400
987	66	2	0	0	$ 0	$ 0	$ 0	1	$ 9.900
988	67	1	363	295	$ 2.950	$ 0	$ 0	0	$ 10.050
989	63	1	0	231	$ 2.310	$ 0	$ 0	-1	$ 0
990	70	3	0	158	$ 1.580	$ 300	$ 200	7	$ 0
991	78	0	0	80	$ 800	$ 0	$ 0	6	$ 0
992	65	1	0	14	$ 140	$ 0	$ 0	5	$ 0
993	64	3	0	0	$ 0	$ 0	$ 0	4	$ 7.500
994	63	0	0	0	$ 0	$ 0	$ 0	3	$ 9.450
995	68	1	0	0	$ 0	$ 0	$ 0	2	$ 10.200
996	60	1	0	0	$ 0	$ 0	$ 0	1	$ 9.000
997	63	3	342	276	$ 2.760	$ 0	$ 0	0	$ 9.450
998	52	0	0	224	$ 2.240	$ 300	$ 200	5	$ 0
999	61	1	0	162	$ 1.620	$ 0	$ 0	4	$ 0
1000	63	2	0	97	$ 970	$ 0	$ 0	3	$ 0

Costo Inventario	$ 205.690
Costo Revisar	$ 8.400
Costo Ordenar	$ 5.600
Costo GoodWill	$ 708.300
Costo Total	**$ 927.990**

Si se cambiase a un valor de T = 5 (cada cinco días revisa) y con un S = 900 se observa lo siguiente:

Día	Demanda	Daños	Llega	Inventario	Costo Inventario	Costo Revisar Fijo	Costo Ordenar	Tiempo en llegar	Costo Goodwill
801	59	3	0	377	$ 3.770	$ 0	$ 0	4	$ 0
802	60	0	0	317	$ 3.170	$ 0	$ 0	3	$ 0
803	56	0	0	261	$ 2.610	$ 0	$ 0	2	$ 0
804	60	6	0	195	$ 1.950	$ 0	$ 0	1	$ 0
805	69	2	461	585	$ 5.850	$ 0	$ 0	0	$ 0
806	52	1	0	532	$ 5.320	$ 0	$ 0	-1	$ 0
807	50	1	0	481	$ 4.810	$ 0	$ 0	-2	$ 0
808	63	0	0	418	$ 4.180	$ 0	$ 0	-3	$ 0
809	54	2	0	362	$ 3.620	$ 0	$ 0	-4	$ 0
810	56	0	0	306	$ 3.060	$ 300	$ 200	5	$ 0
811	56	2	0	248	$ 2.480	$ 0	$ 0	4	$ 0
812	62	12	0	174	$ 1.740	$ 0	$ 0	3	$ 0
813	63	2	0	109	$ 1.090	$ 0	$ 0	2	$ 0
814	68	1	0	40	$ 400	$ 0	$ 0	1	$ 0
815	54	2	594	578	$ 5.780	$ 0	$ 0	0	$ 2.100
816	57	8	0	513	$ 5.130	$ 0	$ 0	-1	$ 0
817	59	0	0	454	$ 4.540	$ 0	$ 0	-2	$ 0
818	56	2	0	396	$ 3.960	$ 0	$ 0	-3	$ 0
819	60	2	0	334	$ 3.340	$ 0	$ 0	-4	$ 0
820	58	3	0	273	$ 2.730	$ 300	$ 200	5	$ 0
821	71	1	0	201	$ 2.010	$ 0	$ 0	4	$ 0
822	55	0	0	146	$ 1.460	$ 0	$ 0	3	$ 0
823	65	0	0	81	$ 810	$ 0	$ 0	2	$ 0
824	54	1	0	26	$ 260	$ 0	$ 0	1	$ 0
825	57	1	627	595	$ 5.950	$ 0	$ 0	0	$ 4.650
826	59	2	0	534	$ 5.340	$ 0	$ 0	-1	$ 0
827	64	1	0	469	$ 4.690	$ 0	$ 0	-2	$ 0
828	55	1	0	413	$ 4.130	$ 0	$ 0	-3	$ 0
829	59	1	0	353	$ 3.530	$ 0	$ 0	-4	$ 0
830	62	4	0	287	$ 2.870	$ 300	$ 200	6	$ 0
831	46	1	0	240	$ 2.400	$ 0	$ 0	5	$ 0
832	65	2	0	173	$ 1.730	$ 0	$ 0	4	$ 0
833	60	1	0	112	$ 1.120	$ 0	$ 0	3	$ 0
834	57	1	0	54	$ 540	$ 0	$ 0	2	$ 0
835	42	0	0	12	$ 120	$ 0	$ 0	1	$ 0
836	68	0	613	557	$ 5.570	$ 0	$ 0	0	$ 8.400
837	67	0	0	490	$ 4.900	$ 0	$ 0	-1	$ 0
838	62	3	0	425	$ 4.250	$ 0	$ 0	-2	$ 0
839	52	0	0	373	$ 3.730	$ 0	$ 0	-3	$ 0
840	61	1	0	311	$ 3.110	$ 300	$ 200	4	$ 0
841	69	1	0	241	$ 2.410	$ 0	$ 0	3	$ 0
842	55	0	0	186	$ 1.860	$ 0	$ 0	2	$ 0
843	76	2	0	108	$ 1.080	$ 0	$ 0	1	$ 0
844	62	1	589	634	$ 6.340	$ 0	$ 0	0	$ 0
845	54	1	0	579	$ 5.790	$ 300	$ 200	5	$ 0
846	61	3	0	515	$ 5.150	$ 0	$ 0	4	$ 0
847	58	3	0	454	$ 4.540	$ 0	$ 0	3	$ 0
848	63	0	0	391	$ 3.910	$ 0	$ 0	2	$ 0
849	64	4	0	323	$ 3.230	$ 0	$ 0	1	$ 0
850	60	2	321	582	$ 5.820	$ 0	$ 0	0	$ 0
851	67	1	0	514	$ 5.140	$ 0	$ 0	-1	$ 0
852	66	0	0	448	$ 4.480	$ 0	$ 0	-2	$ 0
853	59	0	0	389	$ 3.890	$ 0	$ 0	-3	$ 0
854	60	3	0	326	$ 3.260	$ 0	$ 0	-4	$ 0
855	72	0	0	254	$ 2.540	$ 300	$ 200	7	$ 0
856	60	1	0	193	$ 1.930	$ 0	$ 0	6	$ 0
857	53	1	0	139	$ 1.390	$ 0	$ 0	5	$ 0
858	59	1	0	79	$ 790	$ 0	$ 0	4	$ 0
859	62	0	0	17	$ 170	$ 0	$ 0	3	$ 0
860	47	1	0	0	$ 0	$ 0	$ 0	2	$ 4.500
861	60	0	0	0	$ 0	$ 0	$ 0	1	$ 9.000
862	56	0	646	590	$ 5.900	$ 0	$ 0	0	$ 8.400
863	65	0	0	525	$ 5.250	$ 0	$ 0	-1	$ 0
864	63	0	0	462	$ 4.620	$ 0	$ 0	-2	$ 0
865	56	1	0	405	$ 4.050	$ 300	$ 200	7	$ 0
866	63	0	0	342	$ 3.420	$ 0	$ 0	6	$ 0
867	56	0	0	286	$ 2.860	$ 0	$ 0	5	$ 0
868	71	0	0	215	$ 2.150	$ 0	$ 0	4	$ 0
869	62	0	0	153	$ 1.530	$ 0	$ 0	3	$ 0
870	63	1	0	89	$ 890	$ 0	$ 0	2	$ 0
871	59	1	0	29	$ 290	$ 0	$ 0	1	$ 0
872	58	4	495	462	$ 4.620	$ 0	$ 0	0	$ 4.350
873	58	0	0	404	$ 4.040	$ 0	$ 0	-1	$ 0
874	59	4	0	341	$ 3.410	$ 0	$ 0	-2	$ 0
875	50	3	0	288	$ 2.880	$ 300	$ 200	5	$ 0

876	51	0	0	237	$ 2.370	$ 0	$ 0	4	$ 0
877	62	1	0	174	$ 1.740	$ 0	$ 0	3	$ 0
878	64	0	0	110	$ 1.100	$ 0	$ 0	2	$ 0
879	51	1	0	58	$ 580	$ 0	$ 0	1	$ 0
880	57	0	612	613	$ 6.130	$ 0	$ 0	0	$ 0
881	63	2	0	548	$ 5.480	$ 0	$ 0	-1	$ 0
882	67	1	0	480	$ 4.800	$ 0	$ 0	-2	$ 0
883	59	0	0	421	$ 4.210	$ 0	$ 0	-3	$ 0
884	61	1	0	359	$ 3.590	$ 0	$ 0	-4	$ 0
885	51	8	0	300	$ 3.000	$ 300	$ 200	7	$ 0
886	51	4	0	245	$ 2.450	$ 0	$ 0	6	$ 0
887	83	0	0	162	$ 1.620	$ 0	$ 0	5	$ 0
888	61	0	0	101	$ 1.010	$ 0	$ 0	4	$ 0
889	49	1	0	51	$ 510	$ 0	$ 0	3	$ 0
890	55	1	0	0	$ 0	$ 0	$ 0	2	$ 600
891	64	1	0	0	$ 0	$ 0	$ 0	1	$ 9.600
892	64	3	600	533	$ 5.330	$ 0	$ 0	0	$ 9.600
893	62	2	0	469	$ 4.690	$ 0	$ 0	-1	$ 0
894	68	4	0	397	$ 3.970	$ 0	$ 0	-2	$ 0
895	48	0	0	349	$ 3.490	$ 300	$ 200	7	$ 0
896	59	3	0	287	$ 2.870	$ 0	$ 0	6	$ 0
897	65	0	0	222	$ 2.220	$ 0	$ 0	5	$ 0
898	55	1	0	166	$ 1.660	$ 0	$ 0	4	$ 0
899	55	4	0	107	$ 1.070	$ 0	$ 0	3	$ 0
900	61	2	0	44	$ 440	$ 0	$ 0	2	$ 0
901	54	1	0	0	$ 0	$ 0	$ 0	1	$ 1.500
902	52	8	551	491	$ 4.910	$ 0	$ 0	0	$ 7.800
903	58	1	0	432	$ 4.320	$ 0	$ 0	-1	$ 0
904	57	0	0	375	$ 3.750	$ 0	$ 0	-2	$ 0
905	50	1	0	324	$ 3.240	$ 300	$ 200	8	$ 0
906	53	0	0	271	$ 2.710	$ 0	$ 0	7	$ 0
907	61	0	0	210	$ 2.100	$ 0	$ 0	6	$ 0
908	62	1	0	147	$ 1.470	$ 0	$ 0	5	$ 0
909	70	3	0	74	$ 740	$ 0	$ 0	4	$ 0
910	62	4	0	8	$ 80	$ 0	$ 0	3	$ 0
911	52	3	0	0	$ 0	$ 0	$ 0	2	$ 6.600
912	59	2	0	0	$ 0	$ 0	$ 0	1	$ 8.850
913	59	2	576	515	$ 5.150	$ 0	$ 0	0	$ 8.850
914	56	1	0	458	$ 4.580	$ 0	$ 0	-1	$ 0
915	53	1	0	404	$ 4.040	$ 300	$ 200	5	$ 0
916	45	1	0	358	$ 3.580	$ 0	$ 0	4	$ 0
917	59	0	0	299	$ 2.990	$ 0	$ 0	3	$ 0
918	54	1	0	244	$ 2.440	$ 0	$ 0	2	$ 0
919	62	0	0	182	$ 1.820	$ 0	$ 0	1	$ 0
920	60	1	496	617	$ 6.170	$ 0	$ 0	0	$ 0
921	61	0	0	556	$ 5.560	$ 0	$ 0	-1	$ 0
922	68	1	0	487	$ 4.870	$ 0	$ 0	-2	$ 0
923	66	1	0	420	$ 4.200	$ 0	$ 0	-3	$ 0
924	52	0	0	368	$ 3.680	$ 0	$ 0	-4	$ 0
925	64	2	0	302	$ 3.020	$ 300	$ 200	7	$ 0
926	56	0	0	246	$ 2.460	$ 0	$ 0	6	$ 0
927	49	3	0	194	$ 1.940	$ 0	$ 0	5	$ 0
928	73	1	0	120	$ 1.200	$ 0	$ 0	4	$ 0
929	70	2	0	48	$ 480	$ 0	$ 0	3	$ 0
930	59	2	0	0	$ 0	$ 0	$ 0	2	$ 1.650
931	62	1	0	0	$ 0	$ 0	$ 0	1	$ 9.300
932	55	2	598	541	$ 5.410	$ 0	$ 0	0	$ 8.250
933	67	1	0	473	$ 4.730	$ 0	$ 0	-1	$ 0
934	59	4	0	410	$ 4.100	$ 0	$ 0	-2	$ 0
935	61	2	0	347	$ 3.470	$ 300	$ 200	4	$ 0
936	50	3	0	294	$ 2.940	$ 0	$ 0	3	$ 0
937	62	0	0	232	$ 2.320	$ 0	$ 0	2	$ 0
938	60	3	0	169	$ 1.690	$ 0	$ 0	1	$ 0
939	68	0	553	654	$ 6.540	$ 0	$ 0	0	$ 0
940	52	0	0	602	$ 6.020	$ 300	$ 200	7	$ 0
941	53	0	0	549	$ 5.490	$ 0	$ 0	6	$ 0
942	67	1	0	481	$ 4.810	$ 0	$ 0	5	$ 0
943	59	1	0	421	$ 4.210	$ 0	$ 0	4	$ 0
944	54	1	0	366	$ 3.660	$ 0	$ 0	3	$ 0
945	53	1	0	312	$ 3.120	$ 0	$ 0	2	$ 0
946	61	1	0	250	$ 2.500	$ 0	$ 0	1	$ 0
947	47	1	298	500	$ 5.000	$ 0	$ 0	0	$ 0
948	49	1	0	450	$ 4.500	$ 0	$ 0	-1	$ 0
949	51	1	0	398	$ 3.980	$ 0	$ 0	-2	$ 0
950	64	2	0	332	$ 3.320	$ 300	$ 200	5	$ 0
951	55	1	0	276	$ 2.760	$ 0	$ 0	4	$ 0
952	64	2	0	210	$ 2.100	$ 0	$ 0	3	$ 0
953	67	1	0	142	$ 1.420	$ 0	$ 0	2	$ 0
954	50	0	0	92	$ 920	$ 0	$ 0	1	$ 0
955	60	2	568	598	$ 5.980	$ 0	$ 0	0	$ 0

956	51	3	0	544	$ 5.440	$ 0	$ 0	-1	$ 0
957	65	0	0	479	$ 4.790	$ 0	$ 0	-2	$ 0
958	62	4	0	413	$ 4.130	$ 0	$ 0	-3	$ 0
959	46	1	0	366	$ 3.660	$ 0	$ 0	-4	$ 0
960	54	0	0	312	$ 3.120	$ 300	$ 200	5	$ 0
961	56	1	0	255	$ 2.550	$ 0	$ 0	4	$ 0
962	63	1	0	191	$ 1.910	$ 0	$ 0	3	$ 0
963	54	1	0	136	$ 1.360	$ 0	$ 0	2	$ 0
964	60	4	0	72	$ 720	$ 0	$ 0	1	$ 0
965	57	1	588	602	$ 6.020	$ 0	$ 0	0	$ 0
966	49	1	0	552	$ 5.520	$ 0	$ 0	-1	$ 0
967	58	2	0	492	$ 4.920	$ 0	$ 0	-2	$ 0
968	65	2	0	425	$ 4.250	$ 0	$ 0	-3	$ 0
969	52	1	0	372	$ 3.720	$ 0	$ 0	-4	$ 0
970	63	1	0	308	$ 3.080	$ 300	$ 200	6	$ 0
971	68	3	0	237	$ 2.370	$ 0	$ 0	5	$ 0
972	62	0	0	175	$ 1.750	$ 0	$ 0	4	$ 0
973	62	4	0	109	$ 1.090	$ 0	$ 0	3	$ 0
974	75	2	0	32	$ 320	$ 0	$ 0	2	$ 0
975	54	3	0	0	$ 0	$ 0	$ 0	1	$ 3.300
976	66	1	592	525	$ 5.250	$ 0	$ 0	0	$ 9.900
977	46	1	0	478	$ 4.780	$ 0	$ 0	-1	$ 0
978	48	0	0	430	$ 4.300	$ 0	$ 0	-2	$ 0
979	43	0	0	387	$ 3.870	$ 0	$ 0	-3	$ 0
980	63	1	0	323	$ 3.230	$ 300	$ 200	6	$ 0
981	42	3	0	278	$ 2.780	$ 0	$ 0	5	$ 0
982	68	0	0	210	$ 2.100	$ 0	$ 0	4	$ 0
983	73	0	0	137	$ 1.370	$ 0	$ 0	3	$ 0
984	55	0	0	82	$ 820	$ 0	$ 0	2	$ 0
985	59	1	0	22	$ 220	$ 0	$ 0	1	$ 0
986	57	1	577	541	$ 5.410	$ 0	$ 0	0	$ 5.250
987	58	2	0	481	$ 4.810	$ 0	$ 0	-1	$ 0
988	49	1	0	431	$ 4.310	$ 0	$ 0	-2	$ 0
989	59	1	0	371	$ 3.710	$ 0	$ 0	-3	$ 0
990	54	1	0	316	$ 3.160	$ 300	$ 200	4	$ 0
991	55	0	0	261	$ 2.610	$ 0	$ 0	3	$ 0
992	61	1	0	199	$ 1.990	$ 0	$ 0	2	$ 0
993	51	0	0	148	$ 1.480	$ 0	$ 0	1	$ 0
994	57	2	584	673	$ 6.730	$ 0	$ 0	0	$ 0
995	64	1	0	608	$ 6.080	$ 300	$ 200	6	$ 0
996	58	5	0	545	$ 5.450	$ 0	$ 0	5	$ 0
997	62	0	0	483	$ 4.830	$ 0	$ 0	4	$ 0
998	73	2	0	408	$ 4.080	$ 0	$ 0	3	$ 0
999	74	2	0	332	$ 3.320	$ 0	$ 0	2	$ 0
1000	53	1	0	278	$ 2.780	$ 0	$ 0	1	$ 0

Los resultados con estos nuevos parámetros son los siguientes:

Costo Inventario	$ 624.690
Costo Revisar	$ 6.300
Costo Ordenar	$ 4.200
Costo GoodWill	$ 132.450
Costo Total	**$ 767.640**

Se observa que el costo total ha disminuido. El objetivo entonces es buscar el mejor T y S que de el menor costo total posible.

Bibliografía

[1] "Ciencia Explicada," [Online]. Available: http://www.ciencia-explicada.com/2012/01/pifia-historica-de-ibm-de-cuando-los.html. [Accessed 27 Julio 2012].

[2] M. Matsumoto, "Mersenne Twister Home Page," [Online]. Available: http://www.math.sci.hiroshima-u.ac.jp/~m-mat/MT/emt.html. [Accessed 27 Julio 2012].

[3] M. Matsumoto and M. Saito, "SIMD-oriented Fast Mersenne Twister (SFMT): twice faster than Mersenne Twister," [Online]. Available: http://www.math.sci.hiroshima-u.ac.jp/~m-mat/MT/SFMT/index.html. [Accessed 27 Julio 2012].

[4] Hvass Laboratories, "RandomOps," [Online]. Available: http://www.hvass-labs.org/projects/randomops/cs/. [Accessed 27 Julio 2012].

[5] F. Panneton, P. L'Ecuyer and M. Matsumoto, "Maximally equidistributed pseudorandom number generators via linear output transformations," [Online]. Available: http://www3.ocn.ne.jp/~harase/megenerators.html. [Accessed 27 Julio 2012].

[6] J. S. C. B. L. N. D. M. N. Jerry Banks, Discrete-Event System Simulation, Prentice Hall, 5 edition (July 6, 2009).

[7] Fox News, "Can You Crack the World's Toughest Sudoku?," 19 Agosto 2010. [Online]. Available: http://www.foxnews.com/scitech/2010/08/19/crack-worlds-toughest-sudoku/. [Accessed 27 Julio 2012].