UNIVERSIDAD DE GRANADA

Simulación de Sistemas

grado en ingeniería informática

Práctica 3

Modelos de Simulación Dinámicos y Discretos

Autor
José María Sánchez Guerrero

Rama
Computación y Sistemas Inteligentes

Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación

Curso 2019-2020

Índice general

Capítulo 1

Mi segundo modelo de simulación de MonteCarlo

Un establecimiento se abastece diariamente de un cierto producto, y necesita decidir cuántas unidades **s** de ese producto pedir cada día Se desea encontrar el valor de s donde se maximice la ganancia esperada. Obtenemos una ganancia de **x** euros por unidad vendida, y una pérdida de **y** euros si no se ha vendido al final del dia. También contaremos con un valor de demanda, el cual serán los productos solicitados cada día y seguirán una distribución de probabilidad **P** con el cual jugaremos para ver los distintos resultados que nos ofrecerán.

1.1. Experimentación inicial

Empezaremos por un modelo de MonteCarlo inicial, en el cual representaremos todas las variables anteriores en el código para calcular la ganancia, y también dispondremos de varias distribuciones para calcular la demanda de productos solicitados. La ganancia viene determinada por lo siguiente:

$$g(s,x,y,d) = \begin{cases} x*s & si \;\; d \geq s \\ x*d-(s-d)*y & si \;\; d \geq s \end{cases} \tag{1.1}$$

Y tendremos estas tres tipos de distribuciones que podremos seleccionar en el código la que queramos. Hay que tener en cuenta que la simulación será para 100 valores de s, por lo que esto influirá también en las distribuciones:

- $P(D=d)$ se distribuye uniformemente entre 0 y 99

- $P(D = d)$ es proporcional a $100 - d$, $\forall d = 0, 1, 2, ..., 99$
- $P(D = d)$ tiene una distribución "triangular"que crece con $d/2500$ entre 0 y 50; y decrece con $(100 - d)/2500$ entre 50 y 99.

Por último, como vamos a ejecutar un modelo varias **veces**, tendremos que obtener la media para cada s y también la desviación típica con la fórmula

$$desviaciont = \sqrt{\frac{sum2 - veces * gananciaesperada * gananciaesperada}{veces - 1}} \quad (1.2)$$

El código de este ejercicio está en el archivo *generadores.c*, proporcionado junto a la memoria. Vamos a ejecutarlo con distintos parámetros para ver cómo afectan al modelo. La ejecución la vamos a dividir en tres: una para $x = 10, y = 1$; otra para $x = 10, y = 5$; y la última para $x = 10, y = 10$. Los valores de las tablas serán tanto la ganancia como la desviación típica para los distintos valores de *veces* y contrastaremos los resultados.

Estos han sido los resultados para la distribución **uniforme**:

Veces	s	**Ganancia**	**Desviación**
100	86	495.68	316.09
1000	91	472.38	308.29
5000	90	457.66	310.85
10000	91	454.09	311.94
100000	87	450.85	304.69

Cuadro 1.1: $x = 10$, $y = 1$

Veces	s	**Ganancia**	**Desviación**
100	77	390.80	388.73
1000	70	347.14	348.71
5000	67	335.88	332.22
10000	65	333.65	323.97
100000	66	330.05	331.99

Cuadro 1.2: $x = 10$, $y = 5$

Veces	s	**Ganancia**	**Desviación**
100	50	290.60	314.91
1000	55	268.84	356.97
5000	48	254.50	306.41
10000	53	249.94	345.77
100000	51	247.03	333.85

Cuadro 1.3: $x = 10$, $y = 10$

Como podemos ver.....

Estos han sido los resultados para la distribución **proporcional**:

Veces	s	Ganancia	Desviación
100	78	322.40	285.75
1000	69	297.04	243.19
5000	71	289.95	244.48
10000	72	287.13	227.74
100000	67	283.75	235.86

Cuadro 1.4: $x = 10$, $y = 1$

Veces	s	Ganancia	Desviación
100	57	233.40	293.49
1000	44	202.47	229.51
5000	39	193.12	203.07
10000	42	193.25	221.17
100000	44	188.60	231.17

Cuadro 1.5: $x = 10$, $y = 5$

Veces	s	Ganancia	Desviación
100	30	167.40	210.69
1000	30	146.60	201.91
5000	27	136.17	179.10
10000	27	135.01	178.65
100000	28	133.66	189.31

Cuadro 1.6: $x = 10$, $y = 10$

Como podemos ver.....

Estos han sido los resultados para la distribución **triangular**:

Veces	s	Ganancia	Desviación
100	83	493.40	217.66
1000	77	476.39	212.23
5000	76	466.11	205.99
10000	76	466.78	206.22
100000	79	464.94	212.76

Cuadro 1.7: $x = 10$, $y = 1$

Veces	s	Ganancia	Desviación
100	74	432.50	282.03
1000	62	408.09	225.36
5000	59	389.14	218.78
10000	58	388.80	215.32
100000	59	386.79	221.56

Cuadro 1.8: $x = 10$, $y = 5$

Veces	s	Ganancia	Desviación
100	51	367.00	225.96
1000	44	335.26	187.24
5000	53	336.00	251.46
10000	48	337.11	217.58
100000	49	333.64	228.09

Cuadro 1.9: $x = 10$, $y = 10$

Vemos que-.....

1.2. Modificaciones del modelo

1.2.1. Cantidad fija de devolución

En este apartado vamos a modificar el modelo construido anteriormente, de tal forma que el establecimiento pueda devolver las unidades no vendidas. De esta forma hay que pagar una cantidad fija de z euros de gastos de devolución de las unidades no vendidas, en vez de tener una pérdida de y euros por unidad. Esta cantidad no varía, a menos que la $z = 0$.

La función de ganancia ahora sería:

$$g(s,x,y,d) = \begin{cases} x * s & si \;\; d \geq s \\ x * d - z & si \;\; d \geq s \end{cases} \tag{1.3}$$

Estará implementada en el código *generadoresModificados.c* y las pruebas que vamos a realizar con él van a ser las mismas que hicimos anteriormente, y asi poder comparar unos resultados con otros. Las ejecuciones nos han dado las siguientes tablas:

z	Distribución	s	Ganancia	Desviación
5	uniforme	95	490.45	297.18
5	proporcional	91	326.41	236.09
5	triangular	95	495.37	203.63
400	uniforme	59	178.91	367.37
400	proporcional	20	18.58	246.17
400	triangular	44	233.82	273.39
200	uniforme	79	318.07	317.13
200	proporcional	61	142.47	255.30
200	triangular	55	323.75	214.21

Cuadro 1.10: Resultados para todas las distribuciones con $x = 10$ y $veces = 100000$

1.2.2. Cantidad relativa de devolución

Por último, vamos a 'fusionar' los dos últimos casos en uno. Si el valor z es relativamente grande, no interesará pagar esa cantidad de dinero cuando queden pocas unidades sin vender. Por otro lado, cuando el número de unidades no vendidas sea pequeño, es preferible asumir la pérdidas de y que tener que pagar los gastos de devolución.

La función de la ganancia se nos quedaría de la siguiente forma:

$$g(s,x,y,d) = \begin{cases} x * s & si \;\; d \geq s \\ x * d - min\{z, (s-d) * y\} & si \;\; d \geq s \end{cases} \tag{1.4}$$

También estará implementada en *generadoresModificados.c*, y para ejecutarla, tendremos que cambiar el parámetro $modificacion = 2$. El resto de valores, se mantendrán exactamente iguales que en las ejjecuciones anteriores. Los resultados han sido los siguientes:

z	Distribución	s	Ganancia	Desviación
100	uniforme	92	411.21	309.86
100	proporcional	72	238.44	247.52
100	triangular	74	415.98	217.21
150	uniforme	86	380.06	327.23
150	proporcional	56	206.16	249.46
150	triangular	61	396.93	211.83
200	uniforme	81	356.57	317.13
200	proporcional	43	189.64	225.30
200	triangular	60	390.21	219.63

Cuadro 1.11: Resultados para todas las distribuciones con $x = 10$, $y = 5$ y $veces = 100000$

Como podemos ver....

y	Distribución	s	Ganancia	Desviación
3	uniforme	79	381.69	334.20
3	proporcional	54	225.98	240.16
3	triangular	64	419.09	210.75
7	uniforme	83	344.55	337.87
7	proporcional	51	172.44	258.79
7	triangular	59	373.18	225.49
10	uniforme	81	336.49	335.62
10	proporcional	53	162.06	260.66
10	triangular	57	358.21	222.97

Cuadro 1.12: Resultados para todas las distribuciones con $x = 10$, $z = 200$ y $veces = 100000$

Podemos ver que las tablas.....

Capítulo 2

Generadores de datos

Este modelo de simulación lo vamos a tratar con el problema del *aparcamiento*, en el cual un coche se dispone a aparcar a una distancia x de su destino. También dispondremos de variables como el número de plazas que alcanza a ver el conductor desde su posición o la probabilidad de que esa plaza esté ocupada o no. El ejercicio consiste en elegir una plaza de aparcamiento c en la cual el conductor, ni se qu

2.1. Mejorando los generadores

Para hacernos una idea de los valores que obtendremos y los parámetros que más afectan al rendimiento de este modelo, vamos a realizar una ejecución inici

2.1.1. Generadores ordenados

2.1.2. Implementación con búsqueda binaria

2.1.3. Mejora de eficiencia en el generador a

2.2. Generadores congruenciales

Para hacernos una idea de los valores que obtendremos y los parámetros que más afectan al rendimiento de este modelo, vamos a realizar una ejecución inici