UNIVERSIDAD DE GRANADA

E.T.S. DE INGENIERÍAS INFORMÁTICA y DE TELECOMUNICACIÓN

Departamento de Ciencias de la Computación e Inteligencia Artificial

SIMULACIÓN DE SISTEMAS

Guión de Prácticas

Práctica 1: Diferentes Modelos de Simulación

Curso 2022-2023

Grado en Informática

Capítulo 1

Mi Primer Modelo de Simulación de MonteCarlo

1.1. Planteamiento del Sistema a Estudiar

Se pretende analizar si puede resultar beneficioso jugar al siguiente juego: el jugador va lanzando sucesivamente una moneda y se va anotando el resultado obtenido (cara, c, o cruz, x). Por cada lanzamiento el jugador debe pagar y (p.e. 10) euros. El juego continua hasta que la diferencia entre el número de caras y de cruces (en valor absoluto) obtenidas sea igual a 3, y entonces el jugador recibe z (p.e. 100) euros como premio.

Obviamente interesará jugar a este juego si se obtiene beneficio, o sea si las ganancias obtenidas, z, superan a las pérdidas, y multiplicado por el número de lanzamientos realizados. Teniendo en cuenta la aleatoriedad intrínseca que hay en este sistema, cada vez que se juegue se obtendrán resultados diferentes, pudiendo ganar unas veces y perder otras (p.e. la secuencia cxccc para el juego tras realizar 5 lanzamientos, 4 caras y 1 cruz, con lo que el beneficio es 100-5*10=50; en cambio la secuencia xxcxcccxxxcxx usa 13 lanzamientos, 5 caras y 8 cruces, con beneficio 100-13*10=-30). Para determinar si es ventajoso jugar o no, tenemos que decidirlo en términos de alguna cantidad estable, en este caso el beneficio *promedio*, que nos indicará si a largo plazo el juego resulta ventajoso o no. Vamos a utilizar un modelo de Monte Carlo para resolver este problema.

1.2. Tareas a Realizar

El programa `lanzamonedas-generico.cpp`, que puede encontrarse en la plataforma de docencia de la asignatura, implementa tal modelo. Básicamente realiza una estimación del beneficio medio obtenido con este juego, jugándolo muchas veces, anotando la ganancia (o pérdida) obtenida cada vez y calculando la media. También calcula el número medio de lanzamientos necesario para detener el juego. El programa toma como entrada el número de repeticiones (simulaciones) que se hacen del juego, así como el valor de y y el valor de z.

- Utilizad este programa (o una modificación del mismo que facilite los cálculos), *varias veces* con el mismo número de repeticiones, y observad los resultados. Probad con valores

diferentes para el número de repeticiones (valores relativamente pequeños, medianos y grandes). Por ejemplo probad 100 veces distintas con número de repeticiones igual a 10, otras 100 veces con repeticiones igual a 500, y otras 100 veces con 10000 repeticiones. Representad gráficamente esos resultados. ¿Qué conclusiones se pueden obtener?

- Probad con diferentes valores de y y z. ¿Se puede obtener una conclusión clara sobre cuándo conviene jugar, según los valores de y y z?

Los resultados obtenidos dependen implícitamente de un factor, que es la probabilidad de obtener cara o cruz al lanzar la moneda (0.5 para una moneda no trucada) y explícitamente del hecho de que la diferencia entre el número de caras y de cruces obtenidas debe ser igual a 3 para detener la partida. Tales factores pueden afectar enormemente a los resultados.

El mismo programa anteriormente comentado, si se ejecuta suministrándole además esos parámetros, también calcula el beneficio medio obtenido y el número medio de lanzamientos necesario. Realmente el programa con solo tres parámetros equivale a la llamada `lanzamonedas-generico numero_de_iteraciones pago_por_lanzamiento pago_al_finalizar 0.5 3`, los dos últimos valores son la probabilidad de obtener cara y la diferencia entre caras y cruces que debe darse para detener la partida.

- Ahora se trata de ver cómo cambios en estos nuevos parámetros afectan al resultado. ¿Qué ocurre si manteneis todos los parámetros menos uno, y ése lo vais variando? Realizad diversas pruebas, tabulad los resultados y sacad conclusiones. Probad también a cambiar los dos parámetros a la vez a ver qué pasa. Haced representaciones gráficas bidimensionales (número medio de lanzamientos versus probabilidad, y número de lanzamientos versus diferencia entre caras y cruces) y tridimensionales.

Capítulo 2

Mi primer Modelo de Simulación Discreto

2.1. Planteamiento del Sistema a Estudiar

Supongamos una compañía que vende un único producto, y quisiera decidir cuantos ítems mantener en inventario durante los siguientes n meses, de cara a minimizar gastos. Los tiempos entre demandas del producto son variables aleatorias independientes e idénticamente distribuidas, exponenciales con media 0.1 por mes (1 demanda cada tres días, en media). El tamaño de las demandas, D, es una variable aleatoria (independiente de cuando ocurre una demanda) con la siguiente distribución:

$$D = \begin{cases} 1 & \text{con probabilidad } 1/6 \\ 2 & \text{con probabilidad } 1/3 \\ 3 & \text{con probabilidad } 1/3 \\ 4 & \text{con probabilidad } 1/6 \end{cases}$$

Al principio de cada mes, la compañía revisa el nivel del inventario, y decide cuántos ítems pedir a su proveedor. Si la compañía pide Z ítems, incurre en un costo de $K + iZ$, donde $K = 32$ es el costo por hacer el pedido, e $i = 3$ es el costo por unidad (si $Z = 0$, no se incurre en ningún costo). Cuando se hace un pedido, el tiempo necesario para que llegue (llamado retraso de envío) es una variable aleatoria uniformemente distribuida entre 0.5 y 1 mes.

La compañía usa lo que se denomina una política estacionaria (s, S) para decidir cuanto pedir, es decir

$$Z = \begin{cases} S - I & \text{si } I < s \\ 0 & \text{si } I \geq s \end{cases}$$

donde I es el nivel de inventario al principio del mes (antes de hacer el pedido).

Cuando se produce una demanda, ésta se satisface inmediatamente si el nivel de inventario es mayor o igual que el tamaño de la demanda. Si no es así, el exceso de demanda solicitado se satisfará en futuros envíos (en cuyo caso el nivel de inventario se hace negativo). Cuando llega un pedido, se emplea en primer lugar para satisfacer el déficit que pudiera haber, y el resto (si queda) se añade al inventario.

Hay otros dos tipos de costos, además del de pedido: de mantenimiento y de déficit, que comentaremos ahora, después de introducir notación adicional. Sea $I(t)$ el nivel del inventario

en tiempo t (que puede ser positivo, negativo o cero). Sea $I^+(t) = \text{máx}(I(t), 0)$ el número de ítems que hay en el inventario en realidad, y $I^-(t) = \text{máx}(-I(t), 0)$ el número de ítems por satisfacer de demandas previas. Supondremos para nuestro modelo que la compañía incurre en un costo promedio de mantenimiento para los n meses de

$$h\frac{\int_0^n I^+(t)dt}{n}$$

donde $h = 1$ por ítem y por mes, y $\int_0^n I^+(t)dt/n$ es el número medio real de ítems en el inventario. Los costos de mantenimiento incluyen alquileres, seguros, tasas,...

También supondremos que la compañía tiene un costo medio de déficit para los n meses de

$$\pi\frac{\int_0^n I^-(t)dt}{n}$$

donde $\pi = 5$ por ítem y por mes, y $\int_0^n I^-(t)dt/n$ es el número medio de ítems vendidos pero no servidos. Este costo incluye pérdidas de confianza de clientes, descuentos...

Vamos a estudiar este problema empleando un modelo de simulación dinámico y discreto. El objetivo es determinar la política de pedido estacionaria que proporcione menor costo total promedio.

Supongamos ahora que algunos responsables de la compañía plantean la posible utilización de otras políticas de pedido, diferentes de la política estacionaria comentada antes. Concretamente, se han propuesto dos posibilidades más:

- Pedir mensualmente el número de unidades vendidas durante el mes anterior.
- Utilizar una política de gestión del inventario estacionaria pero más dinámica, de forma que no se hacen pedidos mensuales, sino que en cualquier momento en que el nivel actual de inventario I llegue a ser menor que s, se hace un pedido de $S - I$ unidades (y no se vuelve a pedir hasta que llegue ese pedido).

2.2. Tareas a Realizar

El programa `inventario.cpp`, disponible en la plataforma de docencia de la asignatura, implementa un modelo de simulación para este sistema funcionado durante 120 meses (10 años). El programa tiene como parámetros de entrada el número de simulaciones (de 10 años cada una) que se van a realizar, así como los parámetros s y S de la política estacionaria que se va a usar. Tiene la capacidad de estimar los costos medios de pedido, mantenimiento y déficit, así como el costo total.

Supondremos que inicialmente $I(0) = 60$, y que no hay ningún pedido en curso.

Queremos comparar las siguientes trece políticas de pedido y elegir la más conveniente:

s	0	0	0	0	20	20	20	20	40	40	40	60	60
S	40	60	80	100	40	60	80	100	60	80	100	80	100

- Investigad experimentalmente, utilizando este programa (o una modificación del mismo que facilite los cálculos), cuál es la política estacionaria más conveniente de las propuestas.

Realizad el estudio varias veces, fijando el parámetro relativo a número de simulaciones a diferentes valores cada vez (por ejemplo 1, 5, 10, 50, 100 y 500). ¿Qué conclusiones se obtienen?

El programa `inventario2.cpp` implementa un modelo de simulación para la política de pedir lo vendido el mes anterior, y el programa `inventario3.cpp` el modelo de política estacionaria pero dinámica.

- Comparad experimentalmente, utilizando esos dos programas si las nuevas propuestas pueden dar lugar a menores costes que con la política estacionaria original.

Capítulo 3

Mi primer Modelo de Simulación Continuo

3.1. Planteamiento del Sistema a Estudiar

Hay una especie de pez pequeño que vive en un lago. Si notamos por x el número de esos peces, entonces su tasa de incremento, en ausencia de factores limitativos, se puede describir aproximadamente mediante la ecuación

$$\frac{dx}{dt} = ax$$

donde a es una constante relacionada con su tasa natural de natalidad. Esta ecuación indica que x se incrementará exponencialmente, sin ningún límite. En cualquier sistema natural habrá una cantidad limitada de recursos, tales como espacio o comida, para mantener a cualquier especie. Si suponemos que el máximo número de individuos de esta especie que podría existir en el lago es igual a la constante b, entonces una ecuación de crecimiento más razonable es:

$$\frac{dx}{dt} = ax\left(1 - \frac{x}{b}\right)$$

donde el término $(1 - x/b)$ refleja la competición entre miembros de la especie por los recursos. Para distintos valores de a y b, la ecuación anterior resulta en patrones de crecimiento más o menos rápido.

Supongamos ahora que en el mismo lago también vive otra especie de pez más grande, cuya comida principal es la especie de pez pequeño. El número de peces grandes, y, tendría una ecuación de crecimiento similar a la anterior, pero el máximo número de individuos de esta especie está limitado por la disponibilidad de comida, que está relacionada directamente con x. Por tanto una ecuación de crecimiento para la especie de pez grande es

$$\frac{dy}{dt} = cy\left(\frac{x}{d} - \frac{y}{e}\right) \tag{3.1}$$

Por supuesto, si los peces pequeños son comidos por los grandes, la tasa de crecimiento de x debería modificarse para reflejar este hecho. Por tanto obtenemos

$$\frac{dx}{dt} = ax\left(1 - \frac{x}{b}\right) - fy \tag{3.2}$$

El sistema de dos ecuaciones anterior describe las interacciones primarias entre las dos especies de peces. Son ecuaciones no lineales y no pueden resolverse analíticamente.

Para ilustrar las características de esas ecuaciones, supongamos que los coeficientes tienen los siguientes valores:

- $a = \frac{2}{30} = 0{,}0667$ días^{-1}
- $b = 5000000$
- $c = \frac{2}{90} = 0{,}0222$ días^{-1}
- $d = 5000000$
- $e = 36000$
- $f = 5$ días^{-1} (en los experimentos posteriores probad también con $f = 4$).

Estos valores implican que el número de peces pequeños x crece a una tasa tres veces mayor que el número de peces grandes, que el máximo número de peces de cada tipo que las condiciones del lago permiten es de 5000000 y 36000, y que la "voracidad" de los peces grandes es alta (cada pez grande devora unos 5 peces pequeños cada día). Se tiene interés en estudiar si ese sistema es estable, en ausencia de factores externos (si alcanza un "equilibrio ecológico").

Supongamos ahora que la especie de pez depredador tiene interés comercial, y que, en algún instante de tiempo, por ejemplo el 50 % de esos peces, o un determinado número de ellos, son pescados. Se desea conocer cómo afectará esta intervención al posible equilibrio del sistema, si se podrá recuperar o no, y cuándo.

Por último, se podría plantear el diseño de una política de pesca óptima, en el sentido de decidir cada cuánto tiempo hay que pescar y cuántos peces deben pescarse cada vez (asumiendo que en este proceso no se capturan peces pequeños, se usan redes de tamaño apropiado), de forma que no se agoten los recursos y se maximice el número de capturas.

Vamos a estudiar este sistema empleando un modelo de simulación dinámico y continuo.

3.2. Tareas a Realizar

El programa `Simulacion_lago2.C`, disponible en la plataforma de docencia de la asignatura, implementa un modelo de simulación para este sistema. El programa devuelve el número de peces de cada especie en diferentes instantes de tiempo. Al ejecutarlo solicita como parámetros de entrada el tiempo total durante el que se estudiará el sistema (en días) y el número inicial de individuos de cada especie.

- Investigad, empleando un tiempo de simulación de varios años, y diferentes valores para el número de individuos de cada especie, si el sistema es capaz de alcanzar un punto estable de equilibrio. Resulta útil representar gráficamente la evolución del número de peces de cada especie en función del tiempo[1].

[1]Por ejemplo se puede usar gnuplot para esto de forma muy sencilla: ejecutar gnuplot, y luego lanzar el comando `plot ''fichero-de-datos'' using 1:2` (o `1:3`). También es interesante representar la evolución de una especie frente a la otra; para ello se puede usar `plot ''fichero-de-datos'' using 2:3`

- Supuesto que el sistema está en equilibrio, investigad cómo afecta una campaña de pesca que capture determinado número de peces grandes, para diferentes valores posibles.
- Investigad diferentes políticas de pesca (cada cuánto tiempo hay que pescar y cuántos peces), con el objetivo de maximizar el número de capturas. Esto requiere algunas modificaciones en el código del programa, si se quiere hacer de forma automatizada.