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 Discreto

Nuestro modelo de simulación consistirá en un servidor que presta un determinado servicio a una serie de clientes, los cuales solicitarán dicho servicio periódicamente. Cuando llega un cliente y el servidor no está ocupado, será atendido inmediatamente; en caso contrario, el cliente tendrá que esperar en la cola. Cuando se completa un servicio, el servidor elegirá al siguiente en una forma FIFO.

Al empezar la simulación, no habrá clientes esperando y el servidor está libre. Utilizaremos el mismo generador exponencial tanto para el tiempo que tardarán en llegar los clientes, como el tiempo que tardará el servidor en atender a cada uno.

1.1. Simulación con incremento fijo de tiempo

En esta simulación, vamos a tratar al tiempo incrementándolo de unidad en unidad. Para evitar problemas con el manejo del tiempo, tendremos que modificar los generadores de datos para que nos devuelvan los valores redondeados al entero más próximo. Si obtenemos un valor igual a 0, devolveremos 1 en su lugar, ya que el suceso generado quedaría en un tiempo anterior al actual, que generamos al incrementar en una unidad.

Este será nuestro código resultante, que nos servirá tanto para generar el tiempo de llegada del cliente como para generar el timepo del servicio (sólo tendremos que modificar la variable *tlleg* por *tserv*):

```
float generallegada(float tlleg){
  float u = random();          // o tambien rand() en lugar de random()
  u = ( u / (RAND_MAX+1.0) ); //RAND_MAX es una constante del sistema
  u = round( -tlleg * log(1-u) );

  if (u != 0)
    return u;
  else
    return 1.0;
}
```

Para la simulación vamos a emplear diferentes unidades de medida de tiempo (horas, minutos, segundos...) y con un número de clientes a atender bastante alto, de unos 10.000, para que los resultados sean robustos. Este es el resultado que obtenemos:

tiempo	**tlleg**	**tserv**	**% tiempo ocioso**	**Media clientes en cola**
horas	0.15	0.1	0.019994	0.000000
medias horas	0.3	0.2	0.625621	0.078153
cuartos de hora	0.6	0.4	7.224885	0.356633
minutos	9	6	31.501728	1.647961
segundos	540	360	33.934223	1.282934

Como podemos ver, los porcentajes de tiempo ocioso del servidor tienen una variación bastante grande. Esto se debe a que los valores pequeños (unidades de tiempo más altas como las horas o la medias horas) producen unos valores muy próximos a cero en los generadores y, en consecuencia, son redondeados. Como ya hemos visto, estos valores no serán devueltos como 0, si no como 1.

Con esto estamos diciendo que un suceso dura más de lo que realmente es, y vamos acumulando este error en toda la simulación. Esta es la razón por la cual los modelos de incremento fijo no son los más apropiados, ya que sus valores son bastante diferentes a los que obtendríamos sobre el papel.

A continuación, vamos a mostrar unas gráficas con diversas ejecuciones para comprobar que los resultados obtenidos son coherentes y ver mejor las diferencias que obtenemos al utilizar distintas medidas tiempo:

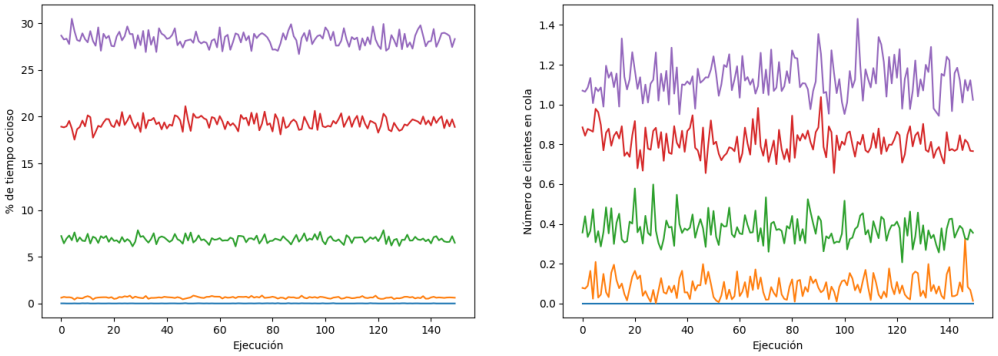

Figura 1.1: Porcentaje de tiempo ocioso del servidor y media de clientes en cola

Aqui podemos observa cómo las gráficas van teniendo un porcentaje de ocupación y un número de clientes en cola más alto a medida que disminuimos las valores *tlleg* y *tserv*. También podemos observar que, pese a que cada una de las líneas rondan un rango de valores, existe cierta variabilidad en los resultados; por lo cual sería un poco irresponsable fiarnos de un solo dato como el de la tabla.

1.2. Simulación con incremento variable de tiempo

Ahora vamos a tratar el mismo problema pero con un incremento de tiempo variable, para hacerlo más eficiente y preciso. En este caso, la variable no tendrá ni por qué ser un entero ni tendremos que parsear los generadores de datos, ya que se incrementará su valor hasta el suceso más cercano. Decimos que es más preciso porque porque no tiene que dar los saltos que daba el anterior modelo y, en consecuencia, gana en eficiencia ya que va suceso a suceso.

Esta es la nueva línea de código que tendremos que añadir (también quitaremos la que aumenta el tiempo de manera uniforme $reloj++$):

```
1 reloj = min(tiempo_llegada, tiempo_salida)
```

A continuación, vamos a ejecutar este modelo de la misma forma que hemos hecho con el modelo anterior y óbservar los cambios que se obtienen:

Tabla con los resultados.....

Vemos que los resultados esta vez han sido más regulares, ya que para todas las medidas de tiempo se han obtenido unos valores entre el 33 y el 34 por ciento. A diferencia del modelo anterior, en este podríamos decir que los resultados son bastante más fiables independientemente de si utilizamos horas, minutos, segundos...

Esto es debido a que ahora no tenemos la acumulación del error que teníamos anteriormente, y los sucesos se producen siempre en el momento declarado, sin ponderar.

Imagenes de la ejecucion .....

Si observamos ahora las gráficas con las diversas ejecuciones, vemos que están prácticamente todas unas encima de otras. Como acabamos de decir, ya no tenemos este error en las distintas medidas de tiempo. No obstante, vemos que aún tenemos bastante variabilidad en los datos, llegando algunos a máximos como XX o a mínimos como XX.

Por otra parte, esta mejora no sólo afecta a las medidas de tiempo, si no que también afecta a la eficiencia. En el modelo anterior, los incrementos en el reloj dependen de la medida del tiempo; mientras que en el modelo actual, el número de incrementos dependerá del número de sucesos. Vamos a comprobarlo extrayendo una tabla que compare los tiempos de ejecución para todas las medidas con incremento fijo, con las mismas medidas en incremento variable:

Table de tiempos.....

Podemos ver que el tiempo de ejecución en el modelo con incremento fijo va aumentando lo que parece lineal o exponencialemente, mientras que en el modelo con incremento variable tenemos unos tiempos prácticamente iguales en unas ejecuciones y otras.

Gráfica de teimpos??????????????

En la gráfica lo podemos confirmar?. Tras estudiar los dos modelos, concluímos que utilizar uno de incremento variable sería lo acertado, no sólo por la mayor eficiencia en tiempo que nos da, si no por algo más importante como es la precisión y calidad al trabajar con distintos tipos de datos o unidades de medida.

1.3. Programa de simulación dinámico y discreto

Ejecucion con m=1 para ver si son iguales a los teoricos

- Tiempo medio de espera en cola = $\frac{tserv^2}{tlleg-tserv} = \frac{6^2}{9-6} = \frac{36}{3} = 12$
- Tiempo medio de estancia en el sistema = $\frac{tserv \cdot tlleg}{tlleg-tserv} = \frac{6 \cdot 9}{9-6} = \frac{54}{3} = 18$
- Número medio de clientes en cola = $\frac{tserv^2}{tlleg(tlleg-tserv)} = \frac{6^2}{9 \cdot (9-6)} = \frac{36}{27} = 1.\widehat{3}$

- Número medio de clientes en el sistema $= \frac{tserv}{tlleg - tserv} = \frac{6}{9-6} = \frac{6}{3} = 2$
- Longitud media de colas no vacías $= \frac{tlleg}{tlleg - tserv} = \frac{9}{9-6} = \frac{9}{3} = 3$
- Porcentaje de tiempo de ocio del servidor $= \left(1 - \frac{tserv}{tlleg}\right) \cdot 100 = \left(1 - \frac{6}{9}\right) \cdot 100 = \frac{100}{3} = 33.\widehat{3}$

Aumentamos el número de servidores pero manteniendo un equilibrio.

Modificad el programa para que pueda repetir varias veces la simulación.

Reemplazar generadores exponenciales por determinísticos y/o uniformes.

Capítulo 2

Mi tercer modelo de simulación Discreto

2.1. Remolcador de un puerto

2.2. Mejoras propuestas para el remolcador

Capítulo 3

Análisis de Salidas y Experimentación

3.1. ¿Cuánto hay que simular?

3.2. Intervalos de confianza

3.3. Comparación de más de dos sistemas