Simulation Project 1 Spring2024

Jorge Alejandro Pichardo Cabrera C312
Universidad de La Habana

1. Introducción

1.1. Proposito

Resolver y analizar el Problema 3 del epígrafe 7 del Libro Simulation 5th Edition by Sheldon M Ross

1.2. Trasfondo

Utilizaremos los contenidos del libro referido para el trabajo, en especial los metodos para simular un proceso Poisson no homogéneo descrito en el epígrafe 6, asi como el lenguaje de programacion python debido a su utilidad a la hora de graficar tablas.

1.3. El Problema

Suppose that jobs arrive at a single server queueing system according to a nonhomogeneous Poisson process, whose rate is initially 4 per hour, increases steadily until it hits 19 per hour after 5 hours, and then decreases steadily until it hits 4 per hour after an additional 5 hours. The rate then repeats indefinitely in this fashion—that is, $\lambda(t + 10) = \lambda(t)$. Suppose that the service distribution is exponential with rate 25 per hour. Suppose also that whenever the server completes a service and finds no jobs waiting, he goes on break for a time that is uniformly distributed on (0, 0.3). If upon returning from his break there are no jobs waiting, then he goes on another break. Use simulation to estimate the expected amount of time that the server is on break in the first 100 hours of operation. Do 500 simulation runs.

1.4. Paramtros Principales

Aqui decidimos tomar como parametros el ratio minimo(Rat_Mi) y maximo(Rat_Ma) del proceso Poisson que describe las llegadas a la cola, tambien denotamos alfa como el tiempo que demora el ratio en fluctuar desde el valor mínimo al máximo, y beta el tiempo que demora el proceso en ir del máximo al mínimo. Definimos G_rat como el ratio de la distribución exponencial G que describe el proceso de servicio. Tenemos el parámetro Break_MaxTime como el tiempo máximo que puede durar un descanso del servidor. Por último maxSimTime es el tiempo que dura la simulación Valores de los parámetros en el problema dado:

- Rat_Mi=4
- Rat_Ma=19
- alfa=5
- beta=5
- G_rat=25
- Break_MaxTime=0.3
- maxSimTime=100

2. Simulación

2.1. Código

```
def G(ratio :float):#exponencial distribution generator
    d=-1/ratio*log(uniform(0,1))
    return d
}
```

Este metodo se usa simplemente para generar el tiempo que demora el servidor en realizar un servicio, se usa el algortimo de la transformación inversa para calcular el valor a partir de una variable uniforme.

```
def T(Ts,Rat_Mi,Rat_Ma,alfa:int , beta:int):
    rest=Ts-int(Ts/(alfa+beta))*(alfa+beta)
    rat=0 #lamda(Ts)
    Ts=Ts+G(Rat_Ma)
    if(rest<alfa):
        rat=Rat_Mi+((Rat_Ma-Rat_Mi)/alfa)*rest
    else:
        rest-=alfa
        rat=Rat_Ma-((Rat_Ma-Rat_Mi)/beta)*rest
    if(uniform(0,1)<=rat/Rat_Ma):
        return Ts
    else:
        return T(Ts,Rat_Mi,Rat_Ma,alfa,beta)
```

Este metodo se usa para simular el proceso Poisson descrito en el problema usando el algoritmo del epígrafe 7.1 del Simulation 5th Edition, para calcular el $\lambda(t)$ simplemente se halla el resto de la divison entera de t por el tamanno del periodo(alfa+beta) y a partir de ahí en dependencia de si ese resto es mayor o menor que alfa se hacen los calulos correspondientes.

```
def Simulate(Rat_Mi,Rat_Ma,alfa,beta,G_rat,Break_MaxTime,maxSimTime):
    curT=0
    nextArrive=T(curT,Rat_Mi,Rat_Ma,alfa,beta)
    nextTask=0
    cont=int(0)
    onBreakTime=0
    while(min(nextArrive,nextTask)<maxSimTime):
        if(nextArrive<=nextTask):
            cont+=1
            curT=nextArrive
            nextArrive=T(curT,Rat_Mi,Rat_Ma,alfa,beta)
        else:
            curT=nextTask
            if(cont==0):
                nextTask=curT+uniform(0,Break_MaxTime)
                onBreakTime+=min(nextTask-curT,maxSimTime-curT)
            else:
                cont-=1
                nextTask=curT+G(G_rat)

    return onBreakTime
```

El codigo anterior es la simulacion en sí, se usa la variable curT para referirse al momento actual de la simulación, nextArrive es el momento en el que llegará el proximo trabajo a la cola, y nextTask es el tiempo en el que el servidor terminará su tarea actual que puede ser tanto el break time como el trabajo actual que esté realizando. La variable cont resume la cantidad de trabajos pendientes y OnBreak es el tiempo que el servidor ha pasado descansando y la variable que se quiere estimar. El algoritmo consiste en que mientras el tiempo actual de la simulacion(curT) sea menor que el tiempo a simular(maxSimTime), se avance hacia el próximo evento agendado, que puede ser el caso 1(agregar un nuevo trabajo a la cola) o el caso 2(esperar a que el servidor termine su tarea actual).Si es el primero de estos entonces se annade un nuevo elemento a la cola(cont++) y se avanza curT hasta ese tiempo. Si es el segundo caso entonces se revisa que la cola este vacia o no, en caso de estar vacía el servidor se toma un descanso y ese tiempo es añadido a la variable a estimar, sino se procesa uno de los trabajos pendientes.

3. Resultados obtenidos

La simulación raw nos dio un resultado de 54 horas aproximadamente con un error de ± 0.07 entre distintas corridas del algoritmo.

3.1. Variación de los parámetros

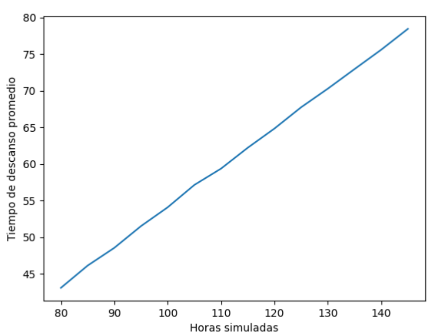
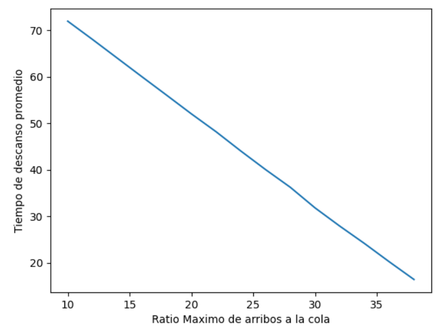


Figure 1. Gráfica donde varía la cantidad de horas simuladas

En la gráfica de la Figura 1 se puede observar como el tiempo de descanso es lineal con respecto a la cantidad de horas que dura la simulacion, lo cual tiene sentido.

Aqui en la Figura 2 podemos ver como el aumento en el ratio de arribos de tareas a la cola disminuye

Figure 2. Gráfica donde varía el pico maximo de intensidad de arribo de tareas a la cola



el tiempo de descanso del servidor de forma lineal también.

Figure 3. Gráfica donde varía el pico maximo de intensidad de arribo de tareas a la cola

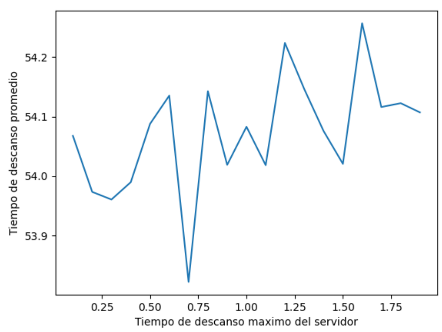
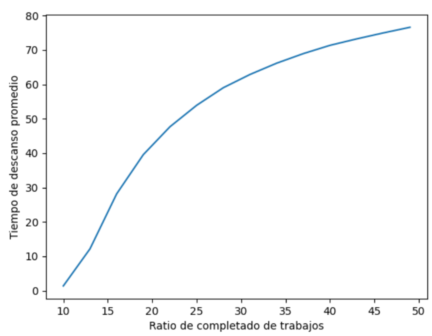


Figure 4. Gráfica donde varía el ratio de completado de tareas por parte del servidor



Aunque la gráfica de la Figura 3 parezca sin sentido debido a los valores atípicos, si nos fijamos mejor observamos que la diferencia entre el máximo y el mínimo es de solo 0.4 horas y que los valores rondan las 54 horas de descanso, estos ”errores” son más a causa de la simulación que de la variacion del parámetro en si, podemos afirmar entonces que los periodos de descanso que se toma el servidor no tienen tanto impacto en el tiempo total que descansa. Esto desde un punto de vista lógico tiene sentido, ya que en el tiempo que descansa los trabajos se acomulan, incurriendo en una carga laboral mayor que luego tiene que cumplir, tarde o temprano. O sea que si analizamos el tiempo que pasa trabajando nos damos cuenta de que esto depende mayormente de la cantidad de tareas que le llegan y la velocidad para completarlas y el tiempo de descanso es precisamente el complemento del tiempo de trabajo.

La ultima grafica(figura 4) se asemeja a una función logarítmica. No encontramos razón aparente para esto a pesar de que se esperaba un aumento lineal.

3.2. Criterio de parada

Un pequeño análisis sobre el criterio de parada arrojó que si bien puede ser menor llegando a los mismos resultados, aumentar el número de simulaciones no aumenta la precision y usando la formula $n = (Z^2 * \sigma^2)/(e^2 * \mu^2)$ donde n es la cantidad de simulaciones requeridas, nos arroja que con poco más de 50 simulaciones ya tenemos una aproximación bastante buena con un error de ± 0.2 y una certeza de un 95% tuvimos en cuenta por su puesto lo que pedía el ejercicio y tomando también la influencia de los errores numéricos los cuales no estan contemplados en los errores matemáticos.