pythHappy Computing

Richard García De la Osa C412

Git link: https://github.com/Regnod/Happy-Computing/tree/master.

Happy Computing es un taller de reparaciones electrónicas, donde se realizan las siguientes actividades:

1. Reparación por garantía (Gratis)
2. Reparación fuera de garantía (350$)
3. Cambio de equipo (500$)
4. Venta de equipos reparados (750$)

Se conoce además que el taller cuenta con 3 tipos de empleados: Vendedor, Técnico y Técnico Especializado.

Para su funcionamiento, cuando un cliente llega al taller, es atendido por un vendedor y en caso de que el servicio que requiera sea una Reparación (sea de tipo 1 o 2) el cliente debe ser atendido por un técnico (especializado o no). Además en caso de que el cliente quiera un cambio de equipo este debe ser atendido por un técnico especializado. Si todos los empleados que pueden atender al cliente están ocupados, entonces se establece una cola para sus servicios. Un técnico especializado solo realizará Reparaciones si no hay ningún cliente que desee un cambio de equipo en la cola.

Se conoce que los clientes arriban al local con un intervalo de tiempo que distribuye poisson con **λ=20** minutos y que el tipo de servicios que requieren pueden ser descrito mediante la tabla de probabilidades:

Tipos de Servicios	**Probabilidad**
1	0.45
2	0.25
3	0.1
4	0.2

Además se conoce que un técnico tarda un tiempo que distribuye exponencial con **λ = 20** minutos, en realizar una reparación cualquiera. Un técnico especializado tarda un tiempo que distribuye exponencial con **λ = 15** minutos, para realizar un cambio de equipos y los vendedores pueden atender cualquier servicio en un tiempo que distribuye normal (N(5 min, 2 min)).

El dueño del lugar desea realizar una simulación de la ganancia que tendría en una jornada laboral si tuviera 2 vendedores, 3 técnicos y 1 técnico especializado.

Modelo de Simulación de Eventos Discretos.

Para realizar el modelo de la simulación se declararon las siguientes variables:

- **duration**: Tiempo total de simulación.
- **time**: Tiempo actual de simulación.
- **client_count**: Cantidad de clientes atendidos en el Taller.

- **profit**: Ganacia durante el día.
- **next_client**: Tiempo de llegada del próximo cliente.
- **third_s**: Variable para saber si hay alguien para el servicio 3 en la cola.
- **vendedores**: Lista de vendedores.
- **tecnicos**: Lista de técnicos.
- **tecnicos_especializados**: Lista de técnicos especializados.
- **cola**: Para llevar los tiempos de cada cliente.

Hay una clase Client, la cual tiene las siguientes variables:

- **time**: Tiempo actual que ha transcurrido desde el comienzo de la simulación hasta la última operación que se ha realizado sobre este.
- **service**: El tipo de servicio.
- **exit_time**: Tiempo con que el cliente sale del Taller.
- **worker**: Para saber que trabajador los atendió.
- **attanded_by**: Para conocer que tipo de empleado lo atendió.
- **state**: Para saber en que estado está el cliente.

Posibles eventos a tratar durante la simulación:

1. LLega un cliente a la tienda
2. El cliente quiere una reparación por garantía
3. El cliente quiere una reparación fuera de garantía
4. El cliente quiere un cambio de equipo
5. El cliente quiere comprar un equipo reparado
6. El cliente tiene que esperar en la cola.

Idea General:

Para la simulación, elejí una jornada laboral de 8 horas(480 minutos) y la filosofía de atender a todos los clientes que quedaban en la cola incluso después de pasados las 8 horas(los clientes que entraron al taller antes de que cerrara). Los clientes son recibidos por los vendedores, si ambos están libres se elige uno random para atenderlo. El vendedor es el que se encarga de redirigir a los clientes a los técnicos o atenderlos ellos mismos, de manera que la cantidad de clientes total se puede obtener sumando la cantidad total atendida por ambos vendedores. los técnicos reciben a los clientes si hay alguno libre, si los 3 están ocupados entonces se verifica si se le puede asignar el especializado(si no está ocupado y no hay nadie en la cola esperando por él). Realizaré varias veces una cantidad determinada de simulaciones, con distintos n(cantidad de simulaciones), y recogeré los promedios de profit, tiempo y cantidad de clientes que se obtuvo en las simulaciones. También brindaré otros datos auxiliares por si llegan a interesar, como la cantidad de clientes atendidos por cada cliente así como el intervalo de tiempo de llegada y de salida de estos clientes.

Generar un cliente:

El método **generate_new_client** se encarga de generar los clientes usando el tiempo actual de la simulación y la distribución de poisson con lambda = 20.

1. **next_client** = **time** + **poisson(20)**.
2. **client.service** ← Tipo de servicio seleccionado para ese cliente.
3. Luego encolamos el cliente creado en la **cola**.

Caso 1 - Llega un cliente a la tienda

1. **client** = **cola.pop()**
2. si **client.state** == **Seller** significa que todavía no lo ha atendido un vendedor
   - **time** = **client.time**
   - El método **attend_seller** se encarga de asignar el cliente a su respectivo trabajador, manejando todos los casos posibles de los vendedores.
   - Y generamos el próximo cliente que entrará a la tienda **next_client** = **generate_new_client()**.
3. si **client.state** == **Tech** significa que ya lo atendió un vendedor y lo redirigió a un técnico.
   - **time** = **client.ctime**
   - El método **attend_tech(client)** maneja a que técnico le asigna el nuevo cliente.
4. si **client.state** == **Spec Tech** significa que ya lo atendió un vendedor y lo redirigió a un técnico especializado.
   - **time** = **client.time**.
   - El método **attend_spectech(client)** maneja a que técnico le asigna el nuevo cliente.
5. si **client.state** == **Completed** significa que ya terminó la operación asignada. Después actualizar el tiempo atual de la simulación.
   - **time** = **min(time, client.time)** siendo time el que va a salir próximo de la cola.
   - Guardamos los datos que queremos tener registrados.
   - En dependencia de su tipo de servicio actualizamos la ganancia.

Caso General: Manejo del cliente:

El método **attend_seller(client)** asigna el cliente a un vendedor, y este usa el método **action_seller** el cual en dependencia del tipo de servicio le actualiza el **state** y el **time** y vuelve a ser encolado, lsuego se repite el caso 1.

1. Si el cliente sacado de la cola tiene **state** == **Tech**, se llama al método que maneja a la asignación de técnicos. **attend_tec(client)** mira la disponibilidad de los técnicos y asigna al primero libre que encuentra, en caso que ninguno esté libre, verifica si no hay ningún cliente que requiera el servicio de **Cambio de equipo** en la cola, y además que el técnico especializado no esté ocupado. En este caso se lo manda a al técnico especializado, sino lo encola otra vez.
2. Si el cliente sacado de la cola tiene **state** == **Spec Tech**, se llama al método **attend_spectech(client)** el cual hace lo mismo que el anterior pero solo maneja los servicios para técnicos especializados.

Sobre las distribuciones de variables aleatorias utilizadas:

En este problema utilizo 3 tipos de variables aleatorias, la **normal**, la **exponencial** y la **poisson**.

La distribución de poisson utilizada se basa en:

Utilizo el método visto en la conferencia de generación de variables aleatorias, $N = max\{n : U_1 * \ldots * U_n \geq e^{-\lambda}\}$.

```
def poisson(lamb):
    u = random.random()
    cnt = 0
    while u >= numpy.e**-lamb:
        u = u * (random.random())
        cnt += 1
    return cnt
```

$F_x(X) = 1 - e^{-\lambda x}$.

$X = -(1/\lambda)ln(U) = -ln(U)/\lambda.$

```
def exp_distribution(lamb):
    u = uniform(0, 1)
    return (-math.log( u)) / (lamb)
```

Utilice el método de rechazos para la distribución normal.

```
def normal_distribution(mu, sigma_square):
    u = uniform(0, 1)
    y1 = 0
    y2 = 0
    while y2 - (((y1-1)**2)/2) <= 0:
        y1 = exp_distribution(1)
        y2 = exp_distribution(1)
    u = uniform(0, 1)
    ans = y1 if u > 0.5 else -y1
    return ans* math.sqrt(sigma_square) + mu
```

Uniform es utilizado de random.

Resultados Generales:

Después de correr la simulación con distintos **n**(1, 10, 100, 1000, 10000), llegamos a los siguientes resultados:

El promedio de profit obtenido ronda los 6800~6900 en su mayoría y tomo valores cercanos a 7000 en pocas ocasiones.

La cantidad de clientes promedio siempre rondaba los 23.5 con una variación muy corta de décimas.

El promedio de minutos de la jornada laboral rondaba los 495 ~496 minutos, vemos que sobrepasan los 480 minutos dado que atendemos a los clientes restantes después de cerrada la tienda.