Proyecto de Simulación. Eventos discretos.

José Carlos Hernández Piñera C411

Problema: La Cocina de Kojo (Kojo's Kitchen)

Descripción del problema:

La cocina de Kojo es uno de los puestos de comida rápida en un centro comercial. El centro comercial está abierto entre las 10:00 am y las 9:00 pm cada día. En este lugar se sirven dos tiposde productos: sándwiches y sushi. Para los objetivos de este proyecto se asumirá que existen solo dos tipos de consumidores: unos consumen solo sándwiches y los otros consumen solo productos de la gama del sushi. En Kojo hay dos períodos de hora pico durante un día de trabajo; uno entre las 11:30 am y la 1:30 pm, y el otro entre las 5:00 pm y las 7:00 pm. El intervalo de tiempo entre el arribo de un consumidor y el de otro no es homogéneo pero, por conveniencia, se asumirá que es homogéneo. El intervalo de tiempo de los segmentos homogéneos, distribuye de forma exponencial. Actualmente dos empleados trabajan todo el día preparando sándwiches y sushi para los consumidores. El tiempo de preparación depende del producto en cuestión. Estos distribuyende forma uniforme, en un rango de 3 a 5 minutos para la preparación de sándwiches y entre 5 y 8 minutos para la preparación de sushi. El administrador de Kojo está muy feliz con el negocio, pero ha estado recibiendo quejas de los consumidores por la demora de sus peticiones. El está interesado en explorar algunas opciones de distribución del personal para reducir el número de quejas. Su interés está centrado en comparar la situación actual con una opción alternativa donde se emplea un tercer empleado durante los períodos más ocupados. La medida del desempeño de estas opciones estará dada por el porciento de consumidores que espera más de 5 minutos por un servicio durante el curso de un día de trabajo. Se desea obtener el porciento de consumidores que esperan más de 5 minutos cuando solo dos empleados están trabajando y este mismo dato agregando un empleado en las horas pico.

Principales ideas seguidas para la solución:

Para darle solución al problema en cuestión se debe simular la llegada de dos tipos de personas a un local donde se prepara comida. Las personas solo pedirán dos tipos de productos (sándwich y sushi), la única diferencia entre los dos productos es el tiempo de preparación de los mismos, el cuál se distribuye de forma uniforme en un rango de 3 a 5 minutos para el primero y de 5 a 8 para el segundo. Se conoce además que el local abre desde un tiempo t_0 hasta T.

Para determinar la orden de cada consumidor se hace uso de la variable aleatoria que distribuye *Bernoulli* de parámetro $p = \frac{1}{2}$, definida de la siguiente forma:

$$ProductType(u) = \begin{cases} sandwich & u \leq \frac{1}{2} \\ sushi & u > \frac{1}{2} \end{cases}$$

Notar que como se conoce que u distribuye uniforme en $[0, 1]$, entonces existe la misma probabilidad que salga para un cliente X un sandwich o un sushi en su pedido.

Para brindar una solución aceptada del problema se hace necesario resolverlo mediante dos variantes y compararlas:

- Simular con dos trabajadores atendiendo los pedidos.
- Simular con dos trabajadores atendiendo los pedidos y uno extra durante los horarios pico definidos en el texto.

Para este manejo de los horarios pico y de los tiempos en general del problema se considera a t_0 como el tiempo 0 y a partir de este se comienzan a sumar minutos, por tanto, el tiempo 90 sería 1 hora y media después de t_0.

Otra cuestión que vale la pena chequear es la obtención de los tiempos de llegada de los clientes al local, estos se obtienen por medio de una variable aleatoria exponencial que presenta por función de distribución $F_x = 1 - e^{-\lambda X}$ y por inversa $X = -(\frac{1}{\lambda}) \ln U$. La obtención de todos los tiempos necesarios para resolver el problema y otras cuestiones importantes se encuentran en el archivo *utils.py*, se recomienda chequearlo para obtener más detalles.

Notemos que la segunda variante para la que se debe resolver el problema es equivalente a la primera, pero con un ligero cambio en los horarios pico; por tanto se puede asegurar que si se conoce la forma de resolver la primera entonces la segunda no ofrece dificultad extra.

A modo de aclaración, se considera como espera por parte de un cliente, si este pasa más de 5 minutos en la cola para ser atendido, es decir, el tiempo en el que el cliente está esperando por un chef libre. En caso de que se quiera calcular todo el tiempo desde la llegada hasta que se va con el producto, es cambiar en el método *run* de *main.py* la forma de obtener la variable *out_time* y en lugar de usar *customer.attended - customer.arrive* emplear *customer.finish - customer.arrive*.

Para hacer un análisis más detallado del problema, usar el *modo debug* fijando en *True* el último parámetro del método *run*.

Modelo de simulación de eventos discretos desarrollado para resolver el problema.

Variables de tiempo:

- elapsed_time: tiempo general.
- attend_customer_time: tiempo que demora en atender un cliente.

Variables contadoras:

- arrives_number: número de arribos.
- arrive_time: tiempo que indica cuando se va a realizar el próximo arribo.
- attended_by_chef: arreglo que muestra cuantos clientes han sido atendidos por cada chefs.

Variables de estado del sistema:

- amount_customers_now: cantidad de clientes actualmente en el sistema.
- pending_customers: cola de clientes que están esperando a ser atendidos.
- chefs: arreglo que tiene en la posición i el id del cliente que está siendo atendido por ese chef.
- bad_time: indica si estamos en horario pico, solo se usa para simular el segundo caso (2 trabajadores y uno extra en el horario pico)

Flujo de la simulación:

- Inicialización:
  - elapsed_time = arrives_number = amount_customers_now
  - attended_by_chef = [0] * amount_chefs
  - chefs = [0] * amount_chefs

    - service_time = ∞
    - generar T_0
    - arrive_time = T_0
- 1er Caso [$arrive\_time = \min(arrive\_time, service\_time_i) \text{and} arrive\_time \leq total\_time\_work$]
    - elapsed_time = arrive_time
    - arrives_number += 1
    - amount_customers_now += 1
    - generar customer { id = arrives_number, order = generar orden, arrive = arrive_time }
    - generar T_k
    - obtener free_chef
        - chefs_i = customer.id
        - customer.attended = elapsed_time
        - attended_by_chef$_i$ += 1
        - generar attend_customer_time
        - service_time$_i$ = elpased_time + attend_customer_time
    - si free_chef = $\emptyset$
        - pending_customer.add(customer)
- 2do Caso [$n\_out = \min(arrive\_time, service\_time_i)$]
    - elapsed_time = n_out
    - customer.finish = n_out
    - amount_customers_now -= 1
    - si pending_customer.any()
        - customer = pending_customers.pop()
        - chefs$_i$ = customer.id
        - customer.attend = n_out
        - attended_by_chef$_i$ += 1
        - generar attend_customer_time
        - service_time$_i$ = elapsed_time + attend_customer_time
    - si not pending_customer.any()
        - chefs$_i$ = 0
        - service_time$_i$ = ∞

Para simular el caso donde hay un chef extra atendiendo pedidos en horarios pico se mantienen todos los elementos y se agregan unas pequeñas funcionalidades extras, pues como ya se comentó en este enfoque el caso 2 contiene al caso 1, entonces la simulación tendría además:

- 3er Caso [$bad\_time = True$]
    - chefs$_{n+1}$ = 0, se activa el chef extra
- 4to Caso [$bad\_time = False$]
    - chefs$_{n+1}$ = ∞

Consideraciones obtenidas a partir de la simulación.

Como en el texto no se hacía referencia al parámetro λ (usado en la exponencial que define los arribos de los consumidores al negocio) , entonces para realizar una prueba lo más certera posible se realizaron alrededor de 1000 simulaciones para cada valor distinto del parámetro.

Se obtivieron los siguientes resultados.

```
------------------Testing with lambda 0.5-------------------------
Dos chefs:  96.69883513306692 %
Tres chefs:  81.94807241256008 %


------------------Testing with lambda 0.3333333333333333-------------------------
Dos chefs:  50.18990825270762 %
Tres chefs:  28.016210168196416 %


------------------Testing with lambda 0.25-------------------------
Dos chefs:  15.503448777031567 %
Tres chefs:  9.776514301993613 %


------------------Testing with lambda 0.2-------------------------
Dos chefs:  6.985216313450435 %
Tres chefs:  4.093974990526715 %


------------------Testing with lambda 0.16666666666666666-------------------------
-
Dos chefs:  3.420202847136945 %
Tres chefs:  2.1482052911013296 %


------------------Testing with lambda 0.14285714285714285-------------------------
-
Dos chefs:  2.119456419273178 %
Tres chefs:  1.2974720564932096 %


------------------Testing with lambda 0.125-------------------------
Dos chefs:  1.4152418055455356 %
Tres chefs:  0.8966080130590298 %
```

El cuadro anterior indica para cada valor de lambda usado y con dos o tres chefs, el porciento de los clientes que esperaron más de cinco minutos en ser atendidos.

A nivel de simulación el tiempo que demora un cliente en ser atendido está dado por *customer.attended - customer.arrive*, lo cuál no es difícil notar. Como conclusión se deriva que el servicio de atención es mejor cuando, en los horarios pico, se añade un cliente extra. A medida que lambda se hace más pequeño, como aumenta el valor esperado del tiempo en el arribo de clientes, es más probable que al llegar alguno exista un chef que se encuentre desocupado y por tanto la diferencia entre el porciento cuando se usa 2 chefs y 3 es menor.

Luego, por los datos antes expuestos, aseguramos que es mucho mejor tener un cliente extra en los horarios picos, en la situación que se presenta.