La Cocina de Kojo. Solución

Ariel Coto Santiesteban, C-412

March 24, 2019

1 Orden 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 tipos de 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 periódos 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 distribuyen de 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. Él 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 periódos 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.

2 Solución

2.1 Principales variables

El problema señala tres variables aleatorias a generar fundamentales:

- t_{arribo} : el tiempo del próximo arribo al sistema
- s : la selección entre sándwich o sushi en el menú
- t_p : el tiempo de preparación del plato.

Teniendo en cuenta que solamente existen dos tipos de platos podemos tambien usar una Bernoulli en vez de una uniforme de 0 a 1. Como existen horarios picos, la solución que se presenta recibe como parámetro los lambdas para ambos horarios. Como el cliente desea testear además si emplear otro trabajador en los horarios picos, en la entrada del modelo se encuentra un booleano q representa si emplear un trabajador extra o no. El tiempo se tomó en segundos. El sistema cuenta con dos colas:

- S_{entry} : representa la cola de clientes a antender en la que se guarda el tiempo de llegada de cada cliente, y
- S_{out} : almacena los tiempos que tuvo que esperar cada cliente para que lo antendieran.

La respuesta al problema es el por ciento de los clientes de S_{out} que esperaron más 5 minutos. Además, se cuenta con un array $worker\_busy$ booleano, que muestra si un trabajador está ocupado o no.

2.2 Principales ideas

Básicamente el algoritmo es el siguiente:

1. Se abre la tienda
2. Si hay alguien en la cola, pasa a ser atendido por el primer trabajador libre.
3. Si ya terminó el trabajador i, se pasa al paso 2.
4. Si es hora pico se agrega si se desea medir con un trabajador más, si no, se elimina al tercer trabajador si existe.
5. Si es hora de cerrar se terminan de atender a los clientes de la cola y no se admiten más arribos.

Los dos eventos esenciales en este problema son:

- Llegada de un cliente al restaurante
- Salida de un cliente del restaurante

Por tanto, el sistema se basa en ejecutar el primero que ocurra en el tiempo, teniendo en cuenta las condiciones del momento, como por ejemplo, si $t_{arribo} < t_s$(tiempo de salida), pero $t_{arribo} > T$ (tiempo de cierre), entonces no hay más arribos, y se pasa a la salida del cliente.

3 El Modelo

Estados:

- Llegada de un cliente al restaurante.
- Entrega al cliente de su pedido por el trabajador i-ésimo (salida del sistema).

Variables:

- t (tiempo de la simulación)
- t_{arribo} (tiempo del próximo arribo de un cliente al restaurante)
- $t_{out}(i)$ (tiempo de salida del cliente del trabajador i-ésimo)
- $W(i)$ (estado del trabajador i-ésimo,ocupado o libre)
- *opened* (boolean, estado de la tienda, abierta o cerrada)
- S_{entry} (cola de clientes a atender en la que se guarda el tiempo de llegada de cada cliente)
- S_{out} (cola, tiempo que estuvo cada cliente en la cola)
- $C(i)$ (cantidad de clientes atendidos por el trabajador i)
- N_a (número de arribos al sistema)
- p (proporción para escoger entre sushi y sándwich)

Modelo:

Se utilizará la notación I_0 para significar $1 \leq i \leq 2$ e I_1 para $1 \leq i \leq 3$. Se utilizará la función $is\_rush\_hour(t)$ para determinar si el tiempo t está en el intervalo de hora pico.

- Inicialización
  1. $opended = True$
  2. $t = N_a = 0$
  3. $W(i) = False$ para I_0
  4. $S_{entry} = S_{out} = (0)$
  5. Generar T_i y hacer $t_{arribo} = T_i$
  6. $t_{out}(i) = \infty$ para I_1
- Caso: Llega un cliente al restaurante (si $t_{arribo} = \min(t_{arribo}, t_{out}(i) \text{ para } I_1)$ y $t_{arribo} < T$)
  1. $t = t_{arribo}$
  2. $N_a = N_a + 1$
  3. S_{entry} añadir t momentáneamente
  4. Si $\exists i\ W(i) = False$, sea i el mínimo
     - $W(i) = True$
     - $C(i) = C(i) + 1$
     - $k = S_{entry}[0]$ se extrae su primer elemento

- S_{out} añadir $t - k$
- Generar P que distribuye como $U(0, 1)$. Si $P > p$ hacer $sushi = True$, si no $sushi = False$
- Si $sushi = True$, generar S_u que distribuye $U(5, 8)$ y hacer $t_{out}(i) = t + S_u$, si no generar S_a que distribuye $U(3, 5)$ y hacer $t_{out}(i) = t + S_a$

5. Si $is\_rush\_hour(t) = True$, generar T_{rush} y hacer $t_{arribo} = t + T_{rush}$,
   - Si queremos probar con un 3er trabajador y no se ha agregado, se agrega.
6. Si no, generar T_{no_rush} $t_{arribo} = t + T_{no\_rush}$
   - Si hay trabajando 3 personas y el 3ro ya termió, entonces se remueve

- Caso: Se libera el tarbajador j (si $t_j = \min(t_{arribo}, t_{out}(i)$ para $I_1)$ ó $(\exists i\ W(i) = True$ and $opened = False))$

1. $t = t_j$
2. Si $S_{entry} = (0)$
   - $W(j) = False$ y $t_{out}(j) = \infty$
3. Si no
   - $k = S_{entry}[0]$ se extrae su primer elemento
   - S_{out} añadir $t - k$
   - Generar P que distribuye como $U(0, 1)$. Si $P > p$ hacer $sushi = True$, si no $sushi = False$
   - Si $sushi = True$, generar S_u que distribuye $U(5, 8)$ y hacer $t_{out}(i) = t + S_u$, si no generar S_a que distribuye $U(3, 5)$ y hacer $t_{out}(i) = t + S_a$

- Caso: Cierre de la tienda ($t \geq T$ ó $t_{arribo} \geq T$)

  - $opened = False$

4 Implementación

Para correr el sistema se debe ejecutar en la carpeta del proyecto *python3 main.py*y seguir con las instrucciones que le aparecen. el main.py ejecutará 100 simulaciones del problema y guardaá los resultados en el archivo *Kojo_simulations.db*. Para mostrar los resultados deber ejecutar *python3 kojo_plotter.py*.

5 Consideraciones

Los resultados de las experimentaciones se almacenaron en Kojo_simulations.db, la cual es una base de datos de sqlite. En este .db existen 2 tablas, *three_workers_waits* y *two_workers_waits*, las cuales almacenan los resultados de experimentar con 2 trabajadores y con tres en las horas picos. Los datos q se almacenan son: *waits* (tiempo de espera de un cliente), *cnt_simulations* (cantidad

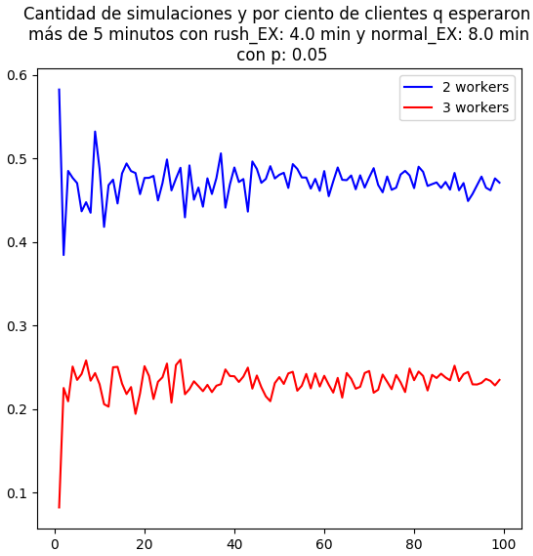

Figure 1: Resultados desde 1 hasta 100 simulaciones

de simulaciones realizadas en el experimento), *rush_EX* (la media utilizada para simular las horas pico), la otra media *normal_EX*, y *sushi_prop* que representa la razón de la seleción entre sushi y sándwich. En general, el comportamiento de los resultados depende mucho de los parámetros del sistema. Si la gente pide mucho sushi, más demora va a haber en la cola, al igual que mientras más distantes estén las medias de las llegadas de los clientes se va a notar más la diferencia.