Proyecto de Simulación de Eventos Discretos

Aeropuerto de Barajas

Marcos Adrián Valdivié Rodríguez C412

1. Orden del problema

En el Aeropuerto de Barajas, se desea conocer cuánto tiempo se encuentran vacías las pistas de aterrizaje. Se conoce que el aeropuerto cuenta con un máximo de 5 pistas de aterrizaje dedicadas a aviones de carga y que se considera que una pista está ocupada cuando hay un avión aterrizando, despegando o cuando se encuentra cargando o descargando mercancía o el abordaje o aterrizaje de cada pasajero.

Se conoce que el tiempo cada avión que arriba al aeropuerto distribuye mediante una función de distribución exponencial con λ=20 minutos.

Si un avión arriba al aeropuerto y no existen pistas vacías, se mantiene esperando hasta que se vacíe una de ellas (en caso de que existan varios aviones en esta situación, pues se establece una suerte de cola para su aterrizaje.

Se conoce además que el tiempo de carga y descarga de un avión distribuye mediante una función de distribución exponencial con λ=30 minutos. Se considera además que el tiempo de aterrizaje y despegue de un avión distribuye normal (N(10,5)) y la probabilidad de que un avión cargue y/o descargue en cada viaje corresponde a una distribución uniforme.

Además de esto se conoce que los aviones tiene una probabilidad de tener una rotura de 0.1. Así, cuando un avión posee alguna rotura debe ser reparado en un tiempo que distribuye exponencial con λ=15 minutos. Las roturas se identifican justo antes del despegue de cada avión.

Igualmente cada avión, durante el tiempo que está en la pista debe recargar combustible y se conoce que el tiempo de recarga de combustible distribuye exponencial con λ=30 minutos y se comienza justamente cuando el avión aterriza.

Se asume además que los aviones pueden aterrizar en cada pista sin ninguna preferencia o requerimiento.

Simule el comportamiento del aeropuerto por una semana para estimar el tiempo total en que se encuentran vacía cada una de las pistas del aeropuerto.

2. Solución al problema

Se decidió implementar un modelo con 5 servidores colocados en paralelo para simular las pistas, los cuales a su vez poseen 4 servidores en serie para simular, en este orden, el aterrizaje, las operaciones sobre la pista, la reparación y el despegue de cada avión.

Las operaciones sobre la pista se dividieron en 3 servidores los cuales son visitados simultáneamente, de los cuales el de reabastecimiento de combustible siempre es visitado, mientras que para los de carga y descarga se generan dos variables aleatorias uniformes p1 y p2 y se visita cada uno con probabilidad p1 y p2 respectivamente. El evento de operaciones en la pista no concluye hasta que estos 3 servidores terminen.

Una vez terminadas todas las operaciones sobre la pista el avión es dirigido al evento de reparación con una probabilidad de 0.1, y con una probabilidad de 0.9 comienza entonces

directamente el proceso de despegue.

Se generalizó además el problema para n pistas y se compararon los resultados obtenidos.

El modelo usado fue el siguiente:

- Variable de tiempo:
  - **current_time**
- Lista de eventos:
  - **next_arrival_time** corresponde al evento del arribo del próximo avión.
  - **started_landing_time**$_i$ corresponde al evento del comienzo del aterrizaje del avión en la pista i.
  - **finished_landing_time**$_i$ corresponde al evento de la finalización del aterrizaje del avión en la pista i.
  - **started_loading_time**$_i$ corresponde al evento del comienzo de la carga del avión en la pista i.
  - **finished_loading_time**$_i$ corresponde al evento de la finalización de la carga del avión en la pista i.
  - **started_downloading_time**$_i$ corresponde al evento del comienzo de la descarga del avión en la pista i.
  - **finished_downloading_time**$_i$ corresponde al evento de la finalización de la descarga del avión en la pista i.
  - **started_fueling_time**$_i$ corresponde al evento del comienzo del abastecimiento de combustible del avión en la pista i.
  - **finished_fueling_time**$_i$ corresponde al evento de la finalización del abastecimiento de combustible del avión en la pista i.
  - **finished_onfloor_time**$_i$ corresponde al evento de la finalización de las acciones sobre la pista.
  - **started_repairing_time**$_i$ corresponde al evento del comienzo de la reparación del avión en la pista i.
  - **finished_repairing_time**$_i$ corresponde al evento de la finalización de la reparación del avión en la pista i.
  - **started_takingoff_time**$_i$ corresponde al evento del comienzo del proceso de despegue en la pista i.
  - **departure_time**$_i$ corresponde al evento de la finalización del proceso de despegue en la pista i.
- Variables de estado del sistema:

- **queue_size** corresponde a la cantidad de aviones que esperan por que se libere alguna pista para poder aterrizar.
- **running_events**$_i$ corresponde a la cantidad de eventos que están ocurriendo simultáneamente en la pista i, se utiliza para conocer cuando se han terminado todas las acciones sobre esta y comenzar la reparación o el despegue del avión que en ella se encuentra.

- Variables contadoras:
  - **airplane_count** corresponde a la cantidad de aviones que han llegado al sistema.
  - **procesed_airplanes** corresponde a la cantidad de aviones que han, al menos, comenzado su proceso de aterrizaje en alguna pista.

- Variables de salida:
  - **occupied_time**$_i$ corresponde al tiempo total que ha estado ocupada la pista i, se utiliza para calcular el tiempo libre.
  - **airplane_arrival**$_i$ corresponde al tiempo en que el avión i comenzó su proceso de aterrizaje.
  - **airplane_departure**$_i$ corresponde al tiempo en que el avión i terminó su proceso de despegue.

Evento 1: Llegada de un avión (next_arrival_time es mínimo).

```
next_arrival_time = generar nuevo tiempo de arribo
airplane_count++
Si existe una pista vacia entonces:
        starting_landing_time[pista] = current_time
sino:
        queue_size++
```

Evento 2: Comienzo del proceso de aterrizaje (started_landing_time$_i$ es mínimo).

```
started_landing_time(i) = infinito
finished_landing_time(i) = generar tiempo de aterrizaje
```

Evento 3: Finalización del proceso de aterrizaje (finished_landing_time$_i$ es mínimo).

```
finished_landing_time(i) = infinito
started_fueling_time(i) = generar tiempo de reabastecimiento
p1 = Generar variable uniforme(0,1)
p2 = Generar variable uniforme(0,1)
u = Generar variable uniforme(0,1)
Si u <= p1 entonces started_loading_time(i) = current_time
```

```
u = Generar variable uniforme(0,1)
Si u <= p2 entonces started_downloading_time(i) = current_time
Actualizar running_events(i) segun corresponda
```

Evento 4: Comienzo del proceso de carga (started_loading_time$_i$ es mínimo).

```
started_loading_time(i) = infinito
finished_loading_time(i) = generar tiempo de carga
```

Evento 5: Finalización del proceso de carga (finished_loading_time$_i$ es mínimo).

```
finished_loading_time(i) = infinito
running_events(i)--
Si running_events(i) es 0 entonces finished_onfloor_time(i) = current_time
```

Evento 6: Comienzo del proceso de descarga (started_downloading_time$_i$ es mínimo).

```
started_downloading_time(i) = infinito
finished_downloading_time(i) = generar tiempo de descarga
```

Evento 7: Finalización del proceso de descarga (finished_downloading_time$_i$ es mínimo).

```
finished_downloading_time(i) = infinito
running_events(i)--
Si running_events(i) es 0 entonces finished_onfloor_time(i) = current_time
```

Evento 8: Comienzo del proceso de reabastecimiento (started_fueling_time$_i$ es mínimo).

```
started_fueling_time(i) = infinito
finished_fueling_time(i) = generar tiempo de reabastecimiento
```

Evento 9: Finalización del proceso de reabastecimiento (finished_fueling_time$_i$ es mínimo).

```
finished_fueling_time(i) = infinito
running_events(i)--
Si running_events(i) es 0 entonces finished_onfloor_time(i) = current_time
```

Evento 10: Finalización de las operaciones sobre la pista (finished_onfloor_time$_i$ es mínimo).

```
finished_onfloor_time(i) = infinito
u = Generar variable uniforme(0,1)
Si u <= 0.1 entonces started_repairing_time(i) = current_time
Sino started_takingoff_time(i) = current_time
```

Evento 11: Comienzo del proceso de reparación(started_repairing_time$_i$ es mínimo).

```
started_repairing_time(i) = infinito
finished_repairing_time(i) = generar tiempo de reparacion
```

Evento 12: Finalización del proceso de reparación(finished_repairing_time$_i$ es mínimo).

```
finished_repairing_time(i) = infinito
started_takingoff_time(i) = current_time
```

Evento 13: Comienzo del proceso de despegue(started_takingoff_time$_i$ es mínimo).

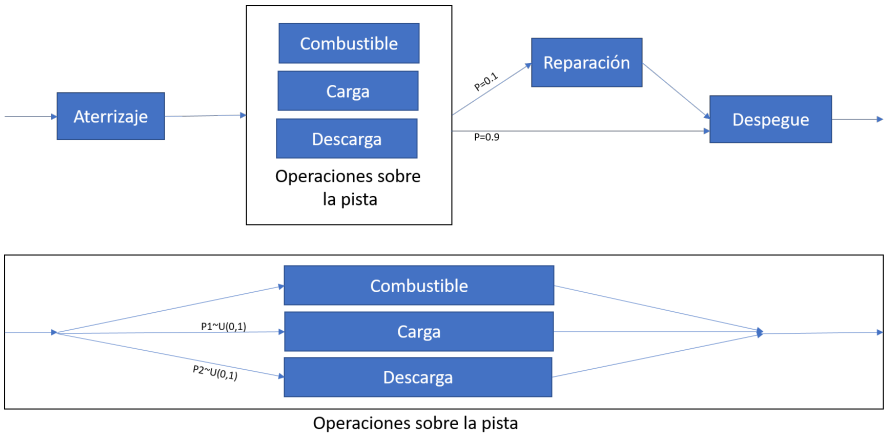
```
started_takingoff_time(i) = infinito
departure_time(i) = generar tiempo de despegue
```

Evento 14: Finalización del proceso de despegue(departure_time$_i$ es mínimo).

```
departure_time(i) = infinito
occupied_time(i) += tiempo que estuvo la pista ocupada
Si hay aviones en la cola entonces traer un nuevo avion.
```

El ciclo principal consiste en buscar el próximo evento, actualizar el tiempo actual según corresponda a dicho evento y procesarlo. El ciclo se detiene una vez haya pasado el tiempo de simulación solicitado y no queden eventos por procesar. En la Figura 1 se puede observar un resumen del flujo de eventos para un avión en una pista.

Figure 1: Diagrama de eventos para una pista



3. Simulación de variables aleatorias

Las variables aleatorias utilizadas en el modelo fueron de tipo uniforme, exponencial y normal.

No. de pistas usadas	Pista 1	Pista 2	Pista 3	Pista 4	Pista 5	Pista 6
3	139	132	230	•	•	•
4	2309	2304	2292	2311	•	•
5	3469	3896	3242	3820	3947	•
6	4163	4447	4209	4214	4737	4033

Table 1: Cantidad de minutos que se mantuvo vacía cada pista

Para la simulación de las variables aleatorias uniformes y exponencial se utilizó el método de la derivada inversa tal como fue explicado en clases, mientras que para la simulación de las variables aleatorias normales se utilizó el método de los rechazos.

4. Resultados

Se simuló el comportamiento del aeropuerto para 3, 4, 5 y 6 pistas, los resultados se pueden observar en la Tabla 1.

En todos los casos se procesaron alrededor de 500 aviones en un tiempo de 7 semanas o 10080 minutos. Podemos observar en la tabla que la cantidad de tiempo libre de las pistas aumenta a medida que aumenta la cantidad de pistas disponibles, como era de esperar. Además, estos tiempos son extremadamente pequeños cuando solo habían 3 pistas, ya que estas permanecían ocupadas la mayor parte del tiempo debido a las largas colas que se formaban en ese caso. Por tanto, podemos concluir que un aeropuerto con un flujo de aviones y de operaciones como el descrito en el problema no puede funcionar correctamente con solo 3 pistas, mientras que para 4, 5 y 6 se puede lograr un equilibrio, siendo este mayor mientras mayor sea la cantidad de pistas disponibles.

5. Anexos

El código del programa usado para la simulación del aeropuerto de barajas se encuentra en el repositorio `github.com/mavaldivie/Barajas_airport`.