Simulación de eventos discretos

Ana Karla Caballero González C-411
Alejandro Camacho Pérez C-412

3 de marzo de 2024

1. Introducción

En el presente proyecto, se buscaba simular el funcionamiento de una fábrica, la cuál para estar activa, necesita de unas n máquinas funcionando. La fábrica cuenta con una cantidad s de máquinas de repuesto, las cuales se ponen en funcionamiento inmediatamente de necesitarse. Una máquina en funcionamiento puede fallar en cualquier momento, y ser llevada a mantenimiento, donde luego de ser reparada, pasaría a ser una máquina de repuesto. Tanto el tiempo de funcionamiento de una máquina, como el tiempo de reparación, son variables aleatorias independientes con sus propias funciones de distribución.

Dada una configuración inicial, se quiere saber la esperanza de vida del tiempo de funcionamiento de la fábrica, la cuál se detiene en caso de que se rompa alguna de las n máquinas en funcionamiento, y no haya máquinas de repuesto.

Las funciones de distribución utilizadas fueron la exponencial y la uniforme para representar el tiempo de funcionamiento y reparación de las máquinas, respectivamente.

Para ello, se definieron las siguientes variables representativas del problema:

- n: Número de máquinas en funcionamiento.
- s: Número de máquinas de repuesto.
- T: Tiempo de funcionamiento de la fábrica.
- TFM_i: Tiempo de funcionamiento de la máquina i.
- TRM_i: Tiempo de reparación de la máquina i.
- F: Distribución de probabilidad de los tiempos de funcionamiento de las máquinas.
- λ: Parámetro de la distribución exponencial.
- G: Distribución de probabilidad de los tiempos de reparación de las máquinas.

- μ: Parámetro de la distribución normal. Representa la media.
- σ: Parámetro de la distribución normal. Representa la desviación estándar.
- M: Número de máquinas en la fábrica.
- M_i: Máquina i.
- r: Número de máquinas rotas
- t: Tiempo actual.

2. Detalles de implementación

Para la implementa de la simulación, se utilizó el lenguaje de programación Python y las librerías *numpy* y *scipy*.

El proyecto tiene la siguiente estructura de archivos:

```
/
|-- src/
|   |-- code/
|   |   |-- auxiliar_functions.py
|   |   |-- factory_data_collector.py
|   |   |-- factory.py
|   |   |-- machines.py
|   |   |-- simulator.py
|   |
|   |-- gui/
|   |   |-- gui.py
|
|-- .gitignore
|-- LICENSE
|-- main.py
|-- README.md
|-- requirements.txt
```

2.1. main.py

Este archivo es el punto de entrada del programa. En él se inicializa la interfaz gráfica y se ejecuta el programa.

2.2. src/code/auxiliar_functions.py

Este archivo contiene funciones de distribución de probabilidad y las variables que estas utilizan.

2.3. src/code/factory_data_collector.py

Contiene las clases *FactoryDataCollector*, *MachineData* y MachineState, donde la última hace función de Enum.

La clase *FactoryDataCollector* tiene la funcionalidad de llevar todos los datos relevantes de la fábrica, en forma de como especie de log de la misma.

La clase *MachineData* es una clase de datos, que contiene el tiempo de funcionamiento y reparación de una máquina, el id de la misma y su estado actuál. Además, es capaz de dar el estado actuál de la máquina en formato de texto.

La clase *MachineState* es una clase de datos, que contiene los estados posibles de una máquina, los cuales son: *WORKING*, *REPAIRING*, *IDLE* y *BROKEN*.

2.4. src/code/factory.py

Contiene la clase *Factory*, la cuál es la encargada de simular el funcionamiento de la fábrica, llevar el control de las máquinas en funcionamiento y de repuesto y es capaz de llevar un registro de los tiempos de funcionamiento y reparación de las máquinas.

2.5. src/code/machines.py

Contiene la clase *Machine*, la cuál es la encargada de simular el funcionamiento de una máquina, y de llevar el control de su tiempo de funcionamiento y reparación.

2.6. src/code/simulator.py

Contiene la clase *Simulator*, la cuál es la que recibe los parámetros necesarios para la simulación, ejecuta la misma y devuelve los resultados.

2.7. src/gui/gui.py

Contiene la clase *FactoryGUI*, la cuál es la encargada de mostrar la interfaz gráfica de la simulación.

2.8. Flujo de la simulación

Al iniciar la aplicación, se muestra la interfaz gráfica, la cuál permite al usuario ingresar los parámetros de la simulación y que cuenta con unos parámetros establecidos por defecto. La simulación se ejecutar al precionar el botón *Start Simulation*

El proceso de la simulación comienza por la ejecución del método *__start_simulation*, el cuál obtiene los parámetros introducidos por el usuario y crea una instancia

de la clase *Simulator*, para luego ejecutar el método *run* de la clase *Simulator*.

El proceso de inicialización de *Simulator* consta por inicializar las variables de la simulación. Una ves se ejecuta el método *run*, se entra en un ciclo que se ejecuta *self.__iterations* veces, es decir, la cantidad de iteraciones de la simulación que se desea realizar. Por cada ciclo, se crea una instancia de la clase *Factory*, y se utiliza el método *start_factory* para iniciar la simulación de la fábrica y luego se guardan los resultados de la simulación.

Una inicialización de la clase *Factory* consta primeramente de crear las M máquinas de la fábrica, a las cuales se les asigna cada una un tiempo de funcionamiento y de reparación en el momento de la inicialización. Luego se crea una cola para las máquinas de repuesto que comienza con s máquinas y una cola para las máquinas rotas que comienza vacía. Las máquinas en funcionamiento se guardan en una lista que será tratada como un heap, donde el peso sería el tiempo de funcionamiento de la máquina, así se garantiza que siempre que se tome una máquina, sea la que tenga el menor tiempo de funcionamiento.

El método *start_factory*, comienza creando una hebra que va a estar ejecutando el método *__repair_machine*, del cuál se hablará más adelante. Luego, pone en funcionamiento todas las máquinas, lo cuál no es más que marcar un tiempo de inicio en cada una de ellas. Con las máquinas funcionando, se marca el tiempo de inicio de la fábrica y se entra en el método *__perform_work_routine*, el cuál es el ciclo encargado de simular el funcionamiento de la fábrica.

Cuando se ejecuta *__perform_work_routine*, se toma el primer elemento del heap de máquinas funcionando, y se entra en un ciclo infinito. Para comprobar si la máquina seleccionada se rompe, se le resta al tiempo actual, el tiempo de inicio de la máquina, y se compara con el tiempo máximo de funcionamiento de la misma. En caso de que esta esté rota, pasa a la cola de máquinas rotas, y se toma la primera máquina de la cola de máquinas de repuesto, se inicia, y se coloca en el heap de máquinas funcionando. En caso de que no existieran máquinas de repuesto en la cola, se sale del ciclo.

El método *__repair_machine* es el encargado de simular el mantenimiento de las máquinas. Consta de un ciclo que se detiene si la fabria no está funcionando. Por cada iteración, se comprueba si $r > 0$, y se toma una de las máquinas rotas. Luego, se pone un tiempo de espera a la hebra igual al tiempo de reparación de la máquina, simulando el proceso de reparar la misma. Una vez terminado, se ejecuta el método *__repair* de la máquina en cuestión, el cuál obiene nuevos valores para el tiempo de funcionamiento y reparación de la máquina, y se coloca en la cola de máquinas de repuesto.

Al terminar el método *__perform_work_routine*, se ejecuta el método *__factory_crashed*, el cuál marca el tiempo de fin de la fábrica, guarda los resultados de la simulación y marca el fin de la simulación.

Tras cada simulación realizada, se actualiza el log de las simulaciones, para ser mostrados en la interfaz gráfica.

3. Resultados y experimentos

A continuación se muestra una tabla con varias corridas de la simulación, con diferentes parámetros de entrada.

Cuadro 1: Resultados de la simulación.

N	S	Lambda	Mu	Sigma	Iterations	E[T]
10	10	1.0	1.0	0.5	10	3.17
10	10	1.0	1.0	0.5	25	3.28
10	10	1.0	1.0	0.5	40	3.21
10	25	1.0	1.0	0.5	10	12.80
10	25	1.0	1.0	0.5	25	14.03
10	25	1.0	1.0	0.5	40	14.90
10	40	1.0	1.0	0.5	10	40.43
10	40	1.0	1.0	0.5	25	46.56
25	10	1.0	1.0	0.5	10	1.49
40	10	1.0	1.0	0.5	40	1.27
10	10	1.0	1.0	0.8	40	2.58
10	10	0.5	0.5	0.5	40	3.02

Cómo se puede observar, la relación $\frac{n}{s}$ es un factor importante en el tiempo de vida de la fábrica. Cuando esta tiende a 0, $E[T]$ aumenta, y cuando tiende a infinito, $E[T]$ disminuye.