Informe del Proyecto de Simulación

February 27, 2024

1 Introducción

En el marco de este proyecto, nos proponemos la tarea de desarrollar una simulación de eventos discretos con el propósito de analizar diversos fenómenos.

La simulación que llevaremos a cabo se basa en un sistema que requiere un número específico de máquinas en funcionamiento para mantenerse operativo. Para mitigar el impacto de averías, se mantiene un conjunto de máquinas adicionales como repuestos. En caso de una avería, una máquina en uso es reemplazada de inmediato por un repuesto, y la averiada es enviada a reparación, atendida por un único reparador.

Nuestro principal objetivo radica en simular este sistema para aproximar el tiempo de falla total (representado por la variable T) del sistema. Deseamos estimar $E[T]$ utilizando los tiempos de reparación aleatorios y los tiempos de falla de las máquinas en uso. Buscamos comprender cómo se comporta el sistema ante este escenario simulado.

1.1 Variables que Describen el Problema

- n: Número de máquinas necesarias para el funcionamiento óptimo del sistema.
- s: Cantidad de máquinas de repuesto inicialmente disponibles.
- G: Función de distribución común para los tiempos de reparación de las máquinas.
- F: Distribución de los tiempos de funcionamiento antes de una falla para las máquinas en uso.
- T: Tiempo en el que el sistema experimenta una falla debido a la falta de repuestos.

El análisis detallado de estas variables nos permitirá comprender mejor la dinámica del sistema y arrojar luz sobre cómo mitigar posibles fallas mediante la gestión efectiva de repuestos y tiempos de reparación.

2 Detalles de Implementación

Para la implementación de este proyecto utilizamos 3 spripts de python: main.py, simulation_repairs.py y variables.py

El main.py ejecuta la simulación con distintos parametros de cantidad de máquinas y proporción de repuestos y expresando los resultados en graficas utilizando la libreria de python matplotlib.pyplot.

```
for num in cant:
    y =[]
    for ratio in ratios:
        s = int(num / ratio)
        y.append(simulate(n=num, s=s, get_explosion_time=weibull_distributio
    matrix.append(y)
```

En simulation_repairs.py primeramente se realiza el paso de inicialización en el cual se crea un array con todas las máquinas activas y otro con las de respaldo y para cada máquina activa se crea un evento de fallo.

```
events = []

computers = []

def get_event_time(event: tuple[float, str, Computer]) -> float:
        return event[0]

available_backups = []
on_repair_backlog = []
```

Luego se organizan los eventos en una pila y se ejecuta un bucle en el que ocurrirán dichos eventos y en dependencia del tipo de evento se realizará una acción:

```
def sort_events():
        events.sort(key=get_event_time, reverse=True)
```

Para los eventos de fallo; en caso de no haber respaldo se detiene la simulación, de lo contrario, aumento el contador de roturas de dicha máquina, saco un respaldo, creo un evento de fallo para dicho respaldo y añado la máquina a la lista de espera donde al colocarse de primera se creara su evento de reparación.

```
case type if type == explosion:

        if len(available_backups) == 0:
                return time

        e[2].break_machine()

        backup = available_backups.pop()
    events.append((time + get_explosion_time(backup.times_broken), explosion
```

```
        computers.remove(e[2])
        computers.append(backup)

        if len(on_repair_backlog) == 0:
                events.append((time + get_repair_time(), repair, e[2]))
        on_repair_backlog.append(e[2])
        sort_events()
```

Si ocurre un evento de tipo reparación; quita el primer elemento de la lista de reparación, lo añade a los respaldos y crea un evento de reparación con el próximo elemento de la lista.

```
case type if type == repair:
        available_backups.append(e[2])
        on_repair_backlog.remove(e[2])

        if len(on_repair_backlog) > 0:
                events.append((time + get_repair_time(), repair, on_repair_b
                sort_events()
```

En variables.py tenemos los métodos de las distribuciones de las distribuciones de las reparaciones y de los fallos de las máquinas.

```
def exponential_distribution():
        num = 0.05 + 0.45 * random.random()
    return -math.log(1.0 - random.random()) / num

def weibull_distribution(k=0):
        k += 1
    num = 30 + 90 * random.random()
    return num * ((-math.log(1.0 - random.random())) ** (1 / k))
```

3 Resultados y Experimentos

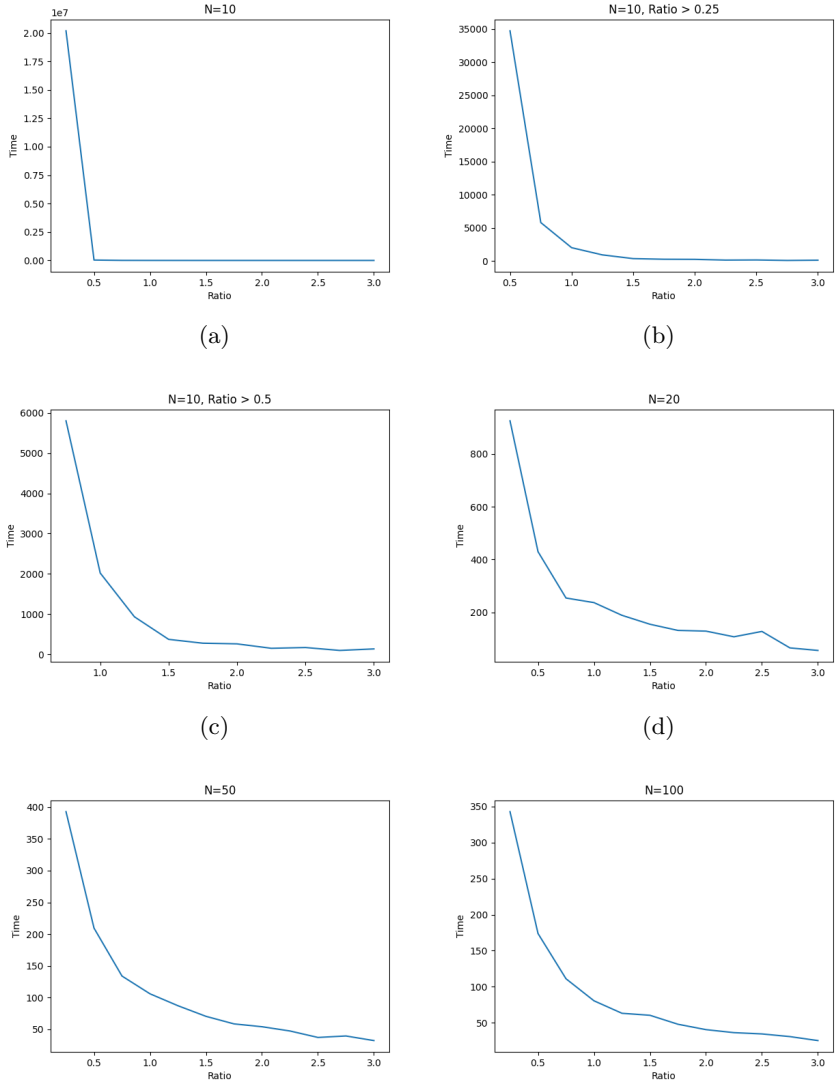
Para incorporar los resultados de la simulación de una manera que facilitara su observación e interpretación los representamos en unas gráficas que se encuentran en la sección de anexos. En esas graficas se muestra el tiempodque tarda el sistema en fallar en funcion de la proporción máquina/respuestos con distintos valores de N, En la figura 1 se muestran los resultados sin normalización, en la figura 2 con normalización logarítmica y la figura 3 con normalización ZScore.
Interpretando los resultados llegamos a la conclusion de que con una cantidad mayor de repuestos en proporción con la cantidad de máquinas operativas se alcanzan mayores tiempos de funcionamiento, por el contrario, con un mayor número de máquinas el sistema colapsa en un tiempo menor.
El analisis estadístico fue necesario para la interpretar los resultados de la simulación, nos basamos en las vaiables Cantidad de Máquinas[N], Ratio de Repuestos[N/S] y el Tiempo de Funcionamiento[T]. Algunas utilidades de estos

resultados puede ser saber con un número fijo de máquinas, cuantos repuestos debes conseguir para alcanzar un tiempo deseado, minimizando así el posible coste de comprar dichos repuestos.
En nuestro proyecto decidimos finalizar la simulación cuando una máquina se rompe y no quedan repuestos disponibles para esa máquina, momento en el cual el sistema deja de estar operativo y se alcanza el tiempo máximo de funcionamiento del sistema, que es el parámetro que queremos analiazar.

4 Modelo Matemático

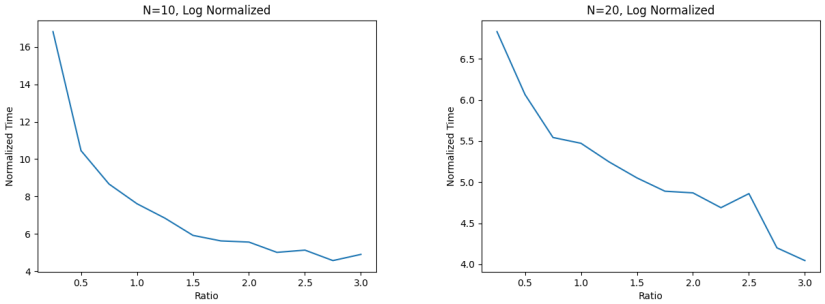
Para la realización de este proyecto decidimos utilizar un Modelo de Simulación de Eventos Discretos en el cual tomamos la posible ocurrencia en cada intervalo de tiempo (día) de un evento de tipo fallo o uno de tipo reparación. con una condición de parada sujeta a la ocurrencia de un evento de tipo fallo mientras se cumple la condición de que mo se encuentren elementos en los repuestos en esa instancia de tiempo. Para cada uno de estos eventos utiliamos una función de distribución, en el caso del evento de fallo utilizamos la distribución de Weibull, la cual mediante observaciones se ha llaegado a la conclusión que se comporta de manera similar a los datos reales de este tipo de eventos además de ser bastante flexible, lo cual nos permite simular en distintos escenerios. Para el de reparación utilizamos la distribución exponencial la cual es frecuentemente usada para modelar una sucesión de eventos independientes como lo son las reparaciones de una máquina.
En nuestro modelo interpretamos la unidad de tiempo como un día. El valor esperado del numero de días que tarda una maquina en romperse lo tomamos entre 30 y 120. El técnico tiene una probabilidad de entre 0.05 y 0.5 de reperar una máquina en una unidad de tiempo. En la distribución de los fallos de las máquinas el tiempo para romperse disminuye mientras más reparaciones se hayan realizado en la máquina.

5 Anexos



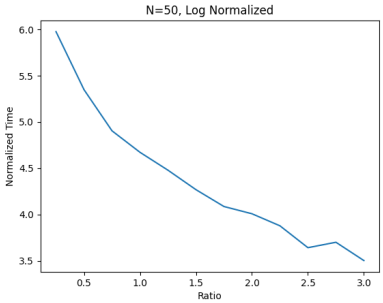
(a) (b) (c) (d) (e) (f)

Figure 1: Resutados de la simulación para distintos valores de N

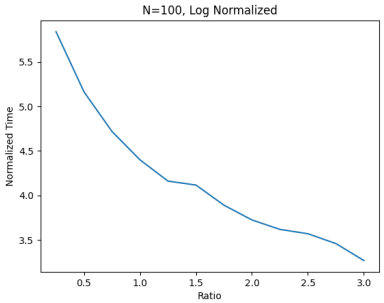


(a)

(b)

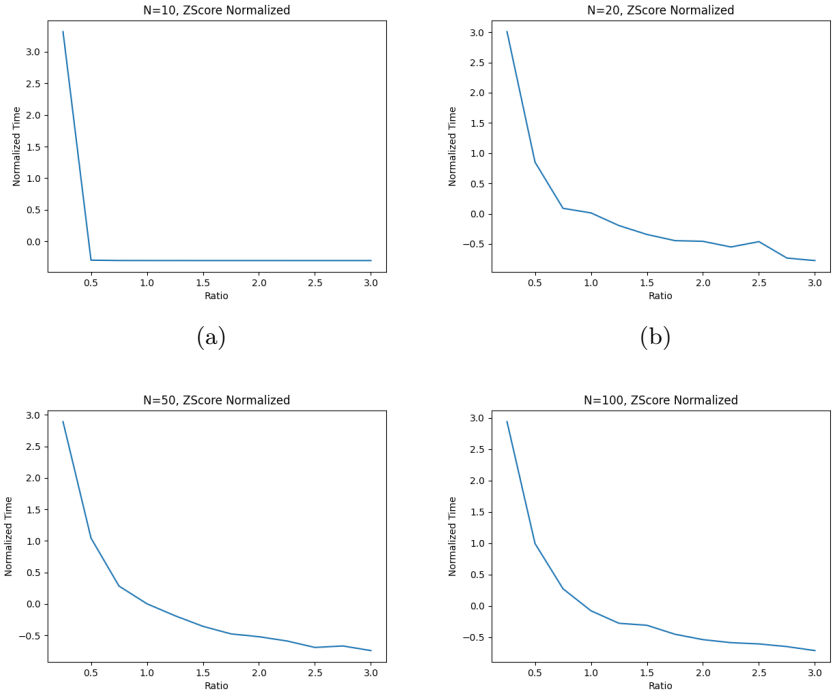


(c)



(d)

Figure 2: Resultados con normalización logarítmica



(a) (b)

(c) (d)

Figure 3: Resultados con normalización ZScore