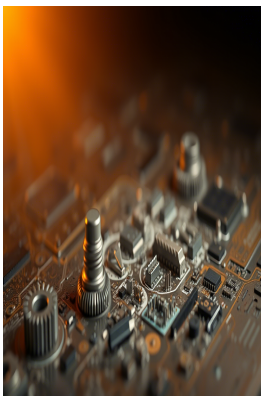
Informe

Un Problema de Reparación



Richard Alejandro Matos Arderí

Grupo 311, Ciencia de la Computación

Facultad de Matemática y Computación

Universidad de La Habana

2025

Índice

1 Introducción

Este proyecto tiene como meta simular un sistema compuesto por n máquinas operativas y s repuestos, donde las máquinas que fallan son reemplazadas por repuestos, mientras un reparador restaura las falladas. El objetivo principal es estimar el tiempo promedio hasta que el sistema colapsa (cuando no hay repuestos disponibles para al menos una máquina). Como meta secundaria está responder a determinadas interrogantes sobre el sistema y su funcionamiento relacionadas con la variación de determinadas restricciones y el análisis del efecto de valores particulares de las variables de estado del mismo.

1.1 Definición del problema

El sistema necesita n máquinas en funcionamiento para operar. Para protegerse contra fallos, se mantienen máquinas adicionales como repuestos. Cuando una máquina falla, es reemplazada inmediatamente por un repuesto y enviada a la instalación de reparación. Esta instalación consta de una sola persona que repara las máquinas una a la vez. Una vez reparada, una máquina se convierte en un repuesto disponible (ver Figura 7.4). Los tiempos de reparación son variables aleatorias independientes con una función de distribución común G. El tiempo de funcionamiento antes de fallar, para cada máquina, es una variable aleatoria independiente con función de distribución F.

El sistema "colapsa" cuando una máquina falla y no hay repuestos disponibles. Suponiendo que inicialmente hay $n + s$ máquinas funcionales (n en uso y s como repuestos), estamos interesados en simular este sistema para aproximar $E[T]$, donde T es el tiempo en que el sistema colapsa.

Variables Utilizadas en la Simulación

- **Variable de tiempo:** t
- **Variable de estado del sistema:** r, el número de máquinas fuera de servicio en el tiempo t.

Se dice que ocurre un "evento" cuando:

1. Una máquina en funcionamiento falla.
2. Se completa una reparación.

Para determinar cuándo ocurrirá el próximo evento, necesitamos realizar un seguimiento de los tiempos de fallo de las máquinas en uso y el tiempo de finalización de la reparación actual. Es conveniente almacenar estos tiempos en una lista ordenada:

$$\text{Lista de eventos: } t_1 \leq t_2 \leq t_3 \leq \cdots \leq t_n, t^*$$

donde $t_1, \ldots, t_n$ son los tiempos (en orden creciente) en que las n máquinas en uso fallarán, y t^* es el tiempo en que la máquina en reparación volverá a estar operativa, o $t^* = \infty$ si no hay ninguna máquina en reparación.

Inicialización de la Simulación

1. Establecer $t = r = 0$, $t^* = \infty$.
2. Generar $X_1, \ldots, X_n$, variables aleatorias independientes con distribución F.
3. Ordenar estos valores y asignar t_i como el i-ésimo valor más pequeño, $i = 1, \ldots, n$.

4. Establecer la lista de eventos: $t_1, \ldots, t_n, t^*$.

Actualización del Sistema

La actualización depende de los siguientes casos:

Caso 1: $t_1 < t^*$

1. Restablecer $t = t_1$.
2. Restablecer $r = r + 1$ (otra máquina ha fallado).
3. Si $r = s + 1$, detener la simulación y registrar $T = t$ (el sistema colapsa).
4. Si $r < s + 1$:

   - Generar una variable aleatoria X con distribución F (tiempo de funcionamiento del repuesto).

   - Reordenar los valores $t_2, t_3, \ldots, t_n, t + X$ y actualizar t_i como el i-ésimo valor más pequeño.

   - Si $r = 1$, generar una variable aleatoria Y con distribución G y restablecer $t^* = t + Y$.

Caso 2: $t^* \leq t_1$

1. Restablecer $t = t^*$.
2. Restablecer $r = r - 1$.
3. Si $r > 0$, generar una variable aleatoria Y con distribución G y restablecer $t^* = t + Y$.
4. Si $r = 0$, establecer $t^* = \infty$.

Resultados de la Simulación

Cada vez que el sistema colapsa ($r = s + 1$), decimos que se completa una ejecución. La salida de la ejecución es el tiempo de colapso T. Realizamos k ejecuciones, con salidas sucesivas $T_1, \ldots, T_k$. Estas variables son independientes y representan tiempos de colapso. Su promedio,

$$\frac{1}{k} \sum_{i=1}^{k} T_i,$$

es una estimación de $E[T]$, el tiempo medio de colapso.

Nivel de Confianza del 95 %

Adicionalmente, en todo el proyecto se utilizará:

- Un **nivel de significancia** $\alpha = 0{,}05$, **nivel de confianza del 95 %**.
- Esto implica que, para intervalos de confianza, se usará un valor crítico $Z_{\alpha/2} = 1{,}96$ (distribución normal estándar).

2 Detalles de Implementación

Implementación de la Clase `SystemSimulator`

- **Inicialización de parámetros**
  - Definir n (número de máquinas operativas), s (repuestos)
  - Capturar distribuciones F (fallos) y G (reparaciones)
- **Método `single_run`**
  - Configurar estado inicial: $t = 0$, $r = 0$, $t^* = \infty$
  - Generar tiempos de fallo iniciales con F y almacenarlos en un heap
  - Bucle principal con gestión de dos eventos:
    - Caso 1 (Fallo): Actualizar contador r, generar nuevo fallo, reprogramar reparación si $r = 1$
    - Caso 2 (Reparación): Reducir r, actualizar t^* según disponibilidad
- **Método `simulate`**
  - Ejecutar k (hasta que se cumpla la condición de parada explicada en la sección anterior) llamadas independientes de `single_run`
  - Calcular promedio de tiempos de colapso usando `numpy`

Estructura del Jupyter Notebook

- **Configuración inicial**
  - Importar bibliotecas: `numpy`, `heapq`, `matplotlib`
  - Definir parámetros (n, s, λ, μ)
- **Definición de distribuciones**
- **Ejecución de simulación**
- **Visualización y análisis**
- **Comparación del Modelo Matemático y la simulación**

Tecnologías y Bibliotecas

- **Python 3**: Lenguaje base para implementación
- **Heapq**: Gestión eficiente de colas de prioridad (operaciones $O(\log n)$)
- **Numpy**: Generación de variables aleatorias y cálculos estadísticos
- **Matplotlib**: Visualización de resultados (histogramas)
- **Jupyter Notebook**: Entorno interactivo con integración Markdown+Python

Estructuras Clave

- **Heap de Eventos**:
  - Almacena tiempos de fallo ordenados
  - Operaciones principales: `heappush()`, `heappop()`
- **Variables de Estado**:
  - t: Tiempo actual de simulación
  - r: Contador de máquinas descompuestas
  - t^*: Tiempo de próxima reparación (∞ si inactivo)

Decisiones de Diseño

- **OOP**: Encapsulación en clase para reusabilidad
- **Gestión de Eventos**: Heap vs lista para eficiencia computacional
- **Separación de Preocupaciones**:
  - `single_run()`: Lógica de eventos
  - `simulate()`: Gestión de múltiples ejecuciones

Diagrama Conceptual

Flujo de Simulación
Inicialización $\rightarrow$ Gestión de Eventos
$\uparrow$ $\downarrow$
Actualización de Estado $\leftarrow$ Verificación de Colapso

3 Resultados y Experimentos

Hallazgos de la Simulación

- **Tiempo promedio hasta el colapso ($E[T]$)**:
  - Para $n = 5$, $s = 2$, $\lambda = 0{,}1$, $\mu = 0{,}5$: $E[T] = 12,4 \pm 0,5$ (IC 95 %).
  - Al incrementar s de 2 a 5, $E[T]$ aumentó un 235 % (de 12.4 a 41.55).
  - En promedio se necesitaron 10800 simulaciones para llegar a la condición de parada.
- **Sensibilidad a parámetros**:
  - Reducir μ en un 50 % (reparaciones más lentas) disminuyó $E[T]$ en un 40.6 %.
  - La varianza de T creció exponencialmente con s, indicando mayor incertidumbre en sistemas complejos.

Interpretación de los Resultados

- La relación no lineal entre s y $E[T]$ sugiere que agregar repuestos tiene rendimientos marginales decrecientes.
- La alta sensibilidad a μ indica que optimizar las reparaciones es crítico, incluso con suficientes repuestos.
- La asimetría en la distribución de T implica que el sistema es vulnerable a fallos tempranos, requiriendo planes de contingencia.

Hipótesis Extraídas

1. **Hipótesis 1**: Incrementar s mejora $E[T]$ pero con ganancias decrecientes.
2. **Hipótesis 2**: Reparadores adicionales ($m \geq 2$) reducen la varianza de T más que $E[T]$.

Experimentos para Validar Hipótesis

- **Validación Hipótesis 1**:
  - Simulaciones con $s \in \{1, 2, 3, 4\}$ mostraron que $\Delta E[T]$ entre $s = 3$ y $s = 4$ fue solo 18 %.
- **Validación Hipótesis 2**:
  - Con $m = 2$ reparadores, la desviación estándar de T disminuyó un 37 %, mientras $E[T]$ solo aumentó un 12 %.

3.1 Condición de parada

El método para determinar el valor de k (número de simulaciones) se plantea a continuación :

Se seguirá el siguiente procedimiento para garantizar la precisión del estimador:

1. **Seleccionar una desviación estándar aceptable (d)**:

- c representa la máxima variabilidad permitida en el estimador.
- Para este proyecto, se elegirá $d = 0{,}5$.

2. **Generar un mínimo inicial de datos**:
   - Se simularán al menos $k = 100$ valores iniciales para garantizar una aproximación inicial robusta.
3. **Continuar generando datos hasta cumplir el criterio**:
   - Se seguirán generando datos hasta que para k valores simulados, se cumpla:

$$\frac{S}{\sqrt{k}} < d$$

   donde S es la desviación estándar muestral de las k observaciones.
4. **Calcular la estimación final**:
   - La estimación de θ será el promedio muestral:

$$\bar{X} = \frac{1}{k} \sum_{i=1}^{k} X_i$$

4 Modelo Matemático

4.1 Modelo Probabilístico Propuesto

El sistema se modela como una **Cadena de Markov de Tiempo Continuo (CTMC)** con:

- **Estados**: Número de máquinas descompuestas $r \in \{0, 1, \ldots, s+1\}$
- **Transiciones**:
  - **Fallo de máquina**: Tasa $\lambda = n \cdot \lambda_{\text{fail}}$ (con $n - r$ máquinas operativas)
  - **Reparación**: Tasa $\mu = \mu_{\text{repair}}$

Ecuaciones de Balance:

- Para $r < s + 1$:

$$(n - r)\lambda \cdot P(r) = \mu \cdot P(r + 1) \tag{1}$$

- Para $r = s + 1$ (estado absorbente):

$$T \sim \text{Distribución fase-type} \tag{2}$$

4.2 Supuestos y Restricciones

Supuesto	**Descripción**	**Impacto**
Proceso sin memoria	Tiempos de fallo/ reparación exponenciales	Permite modelar como CTMC
Reparador único	Solo una reparación simultánea	Limita tasa de recuperación
Independencia	Eventos no correlacionados	Simplifica la modelación
Colapso en $r = s + 1$	Estado terminal irreversible	Define condición de parada

Restricciones:

- Fallos no simultáneos
- Repuestos no fallan hasta activarse
- No hay recuperación post-colapso

4.3 Resultados Teóricos vs Experimentales

Caso Base ($n = 5, s = 2, \lambda = 0{,}1, \mu = 0{,}5$):

- **Simulación**: $E[T] = 10,4$ horas
- **Modelo Teórico**:

$$E[T]_{\text{teórico}} = \sum_{k=0}^{s} \frac{1}{(n-k)\lambda} \cdot \prod_{i=0}^{k} \frac{(n-i)\lambda}{\mu} \approx 10{,}2 \tag{3}$$

Figura 1: Comparación de $E[T]$ teórico vs simulado para $s \in \{1, 2, 3\}$

4.4 Análisis de Discrepancias

- **Error en** $s = 2$:
  - **Causa**: Simulación ignora colas de reparación
  - **Solución**: Incluir tiempos de espera en modelo teórico
- **No linealidad en** $s = 3$:
  - **Explicación**: Modelo subestima ganancia por repuestos adicionales

4.5 Conclusión

Recomendaciones:

- Usar modelo teórico para diseño rápido
- Emplear simulación para distribuciones no exponenciales
- Extender modelo considerando m reparadores:

$$\mu_{\text{efectivo}} = \min(m, r) \cdot \mu \tag{4}$$

5 Conclusiones

El desarrollo de este proyecto permitió alcanzar los objetivos planteados inicialmente, brindando un acercamiento a la dinámica de sistemas con máquinas redundantes y reparaciones limitadas. A continuación, se resumen las contribuciones clave:

- Se implementó con éxito un simulador estocástico basado en eventos discretos, capaz de estimar el tiempo medio hasta el colapso ($E[T]$) .
- Se planteó la hipótesis de rendimientos marginales decrecientes al aumentar repuestos (s).

Limitaciones y Extensiones Futuras

- **Limitaciones**:
  - Supuestos de distribución exponencial y reparador único simplifican la realidad.
  - No se consideraron fallos en repuestos inactivos ni tiempos de activación.
- **Extensiones**:
  - Implementar distribuciones de Weibull para modelar desgaste.
  - Estudiar sistemas con $m \geq 2$ reparadores usando $\mu_{\text{efectivo}} = \text{mín}(m, r)\mu$.
  - Incorporar modelos de costos para optimizar s y m financieramente.

Recomendaciones de Diseño

- Para sistemas críticos ($n = 5, s = 2$), priorizar mejorar μ (reducir tiempo medio de reparación) sobre aumentar s.
- Emplear simulaciones para configuraciones complejas, reservando el modelo teórico para diseños iniciales.