Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ingeniería Industrial y de Sistemas
ICS2123 – Modelos Estocásticos
Profesores Maximiliano González, Nicolás Vargas y Andrés Navarro
Ayudante Jefe Francisco Tamarín
Segundo Semestre 2024

Tarea 4:
CMTC y Sistemas de Espera

Integrantes

◇ **Integrante 1:** Jacinta Ortiz Santander (Sección 4)

◇ **Integrante 2:** Vicente Lavagnino Gatica (Sección 2)

Instrucciones

◇ Esta tarea debe ser realizada **individualmente** o **en parejas** (pueden ser de diferentes secciones dentro de las que son coordinadas). Deben indicar en la primera plana del PDF los nombres de los/as integrantes, junto con las secciones respectivas de c/u.

◇ Tienen plazo hasta el **Viernes 29 de octubre a las 23:59 hrs** para entregar sus soluciones.

◇ Deben entregar un archivo reporte, **en formato PDF**. En el archivo PDF deben entregar sus respuestas de los 3 problemas. Los códigos implementados deben adjuntarse como un archivo .zip aparte en el buzón. En el PDF deberá estar el desarrollo, resumen de resultados y correspondiente análisis. Se recomienda fuertemente utilizar LaTeX. Sin embargo, estas también pueden ser escritas en algún otro editor de texto o a mano y escaneadas, pero **debe estar ordenado y legible**, de lo contrario, se descontarán 3 décimas a la nota final de la tarea. Cualquier código que utilicen de internet debe ser debidamente citado insertando el *link* asociado.

◇ Si realizan la tarea completa en LaTeX, incluyendo todas las fórmulas y ecuaciones, tendrán **3 décimas extras**.

◇ La tarea tiene un total de 60 puntos. La nota final será calculada como la suma de las tres preguntas mediante la fórmula

$$Nota = \frac{P_1 + P_2 + P_3}{60} \cdot 6 + 1$$

Donde P_i corresponde al puntaje de la pregunta i. Se aproxima a dos decimales.

◇ Las preguntas se harán a través del foro de discusión asociado a la Tarea 3 en Canvas, por lo que les pedimos que **NO** manden dudas por mail sobre la tarea. De esta manera, todos tienen acceso a las respuestas.

◇ Seremos estrictos en que **no se aceptarán envíos por mail**. Evite problemas de internet, del sistema Canvas, de la calidad de la foto o cualquier otro que pudiese ocurrir al entregar sobre la hora. El/La alumno/a que envíe su tarea después de la hora límite, tendrá **nota 1.0**.

Problema 1 (20 puntos)

a)

Definimos este sistema como $X_t = (i, j)$ siendo i la cantidad de correos de Metodos de Optimización y j como la cantidad de correos de Modelos Estocásticos, de esta forma el espacio de estados sería $\Omega = \{(0, ..., O), (0, ..., E)\}$.
Notamos que este es un sistema de nacimiento y muerte ya que las transiciones desde el estado (i,j) sólo pueden ser al estado $(i-1, j), (i, j-1), (i+1, j)$ o al estado $(i, j+1)$. Las tasas de transición instantáneas son:

- $$q_{(i,j)(i+1,j)} = \lambda_1 \qquad \forall i < O, \forall j \in [0, E]$$
- $$q_{(i,j)(i-1,j)} = \mu_1 \qquad \forall i \in [1, O], \forall j \in [0, E]$$
- $$q_{(i,j)(i,j+1)} = \lambda_2 \qquad \forall j < E, \forall i \in [0, O]$$
- $$q_{(i,j)(i,j-1)} = \mu_2 \qquad \forall j \in [1, E], \forall i \in [0, O]$$

Para las tasas de permanencia tenemos los siguiente:

- $$v_{(0,0)} = \lambda_1 + \lambda_2$$
- $$v_{(i,0)} = \lambda_1 + \lambda_2 + \mu_1 \qquad \forall i \in [1, O-1]$$
- $$v_{(0,j)} = \lambda_1 + \lambda_2 + \mu_2 \qquad \forall j \in [1, E-1]$$
- $$v_{(i,j)} = \lambda_1 + \lambda_2 + \mu_1 + \mu_2 \qquad \forall i \in [1, O-1], \forall j \in [1, E-1]$$
- $$v_{(i,E)} = \lambda_1 + \mu_1 + \mu_2 \qquad \forall i \in [1, O-1]$$
- $$v_{(O,j)} = \lambda_2 + \mu_1 + \mu_2 \qquad \forall j \in [1, E-1]$$
- $$v_{(O,E)} = \mu_1 + \mu_2$$
- $$v_{(0,E)} = \lambda_1 + \mu_2$$
- $$v_{(O,0)} = \lambda_2 + \mu_2$$

Las probabilidades de transición son:

$$P_{(i,j),(i+1,j)} = \begin{cases} \frac{\lambda_1}{\lambda_1+\lambda_2}, & \text{si } (i,j) = (0,0), \\ \frac{\lambda_1}{\lambda_1+\lambda_2+\mu_1}, & \text{si } j = 0, i \in [1, O-1] \\ \frac{\lambda_1}{\lambda_1+\lambda_2+\mu_2}, & \text{si } i = 0, j \in [1, E-1] \\ \frac{\lambda_1}{\lambda_1+\lambda_2+\mu_1+\mu_2}, & \text{si } i \in [1, O-1], j \in [1, E-1] \\ \frac{\lambda_1}{\lambda_1+\mu_1+\mu_2}, & \text{si } j = E, i \in [1, O-1] \\ \frac{\lambda_1}{\lambda_1+\mu_2}, & \text{si } i = 0, j = E \end{cases}$$

$$P_{(i,j),(i-1,j)} = \begin{cases} \frac{\mu_1}{\lambda_1+\lambda_2+\mu_1}, & \text{si } j=0, i \in [1, O-1], \\ \frac{\mu_1}{\lambda_1+\lambda_2+\mu_1+\mu_2}, & \text{si } i \in [1, O-1], j \in [1, E-1] \\ \frac{\mu_1}{\lambda_1+\mu_1+\mu_2}, & \text{si } j=E, i \in [1, O-1] \\ \frac{\mu_1}{\lambda_2+\mu_1+\mu_2}, & \text{si } i=O, j \in [1, E-1] \\ \frac{\mu_1}{\mu_1+\mu_2}, & \text{si } i=O, j=E \\ \frac{\mu_1}{\lambda_2+\mu_2}, & \text{si } i=O, j=0 \end{cases}$$

$$P_{(i,j),(i,j+1)} = \begin{cases} \frac{\lambda_2}{\lambda_1+\lambda_2}, & \text{si } (i,j)=(0,0), \\ \frac{\lambda_2}{\lambda_1+\lambda_2+\mu_1}, & \text{si } j=0, i \in [1, O-1] \\ \frac{\lambda_2}{\lambda_1+\lambda_2+\mu_2}, & \text{si } i=0, j \in [1, E-1] \\ \frac{\lambda_2}{\lambda_1+\lambda_2+\mu_1+\mu_2}, & \text{si } i \in [1, O-1], j \in [1, E-1] \\ \frac{\lambda_2}{\lambda_2+\mu_1+\mu_2}, & \text{si } i=O, j \in [1, E-1] \\ \frac{\lambda_2}{\lambda_2+\mu_2}, & \text{si } i=O, j=0 \end{cases}$$

$$P_{(i,j),(i,j-1)} = \begin{cases} \frac{\mu_2}{\lambda_1+\lambda_2+\mu_2}, & \text{si } i=0, j \in [1, E-1], \\ \frac{\mu_2}{\lambda_1+\lambda_2+\mu_1+\mu_2}, & \text{si } i \in [1, O-1], j \in [1, E-1] \\ \frac{\mu_2}{\lambda_1+\mu_1+\mu_2}, & \text{si } j=E, i \in [1, O-1] \\ \frac{\mu_2}{\lambda_2+\mu_1+\mu_2}, & \text{si } i=O, j \in [1, E-1] \\ \frac{\mu_2}{\mu_1+\mu_2}, & \text{si } i=O, j=E \\ \frac{\mu_2}{\lambda_1+\mu_2}, & \text{si } i=0, j=E \end{cases}$$

b)

La probabilidad límite P_j se define como:

$$P_j = \lim_{t \to \infty} P_{i,j}(t)$$

donde para una CTMC, se cumple que

$$v_j \cdot P_j = \sum_{k \neq j} q_{kj} \cdot P_k, \quad \sum_{n=0}^{\infty} P_n = 1$$

donde la distribución límite corresponde a la solución de estas ecuaciones de equilibrio.

En ese sentido, podemos definir $\pi_{(i,j)}$ como la probabilidad límite de que el sistema esté en el estado (i,j). Luego planteamos las ecuaciones de equilibrio de largo plazo basándonos en las transiciones de entrada y salida de cada estado. Donde para cada estado, la tasa de entrada debe igualar la tasa de salida:

$$\pi_{(i,j)} \cdot v_{(i,j)} = \begin{cases} \pi_{(i-1,j)} \cdot \lambda_1 & \text{si } i > 0 \\ +\pi_{(i,j-1)} \cdot \lambda_2 & \text{si } j > 0 \\ +\pi_{(i+1,j)} \cdot (i+1)\mu_1 & \text{si } i < O \\ +\pi_{(i,j+1)} \cdot (j+1)\mu_2 & \text{si } j < E. \end{cases}$$

La tasa de salida desde el estado (i,j) es:

$$v_{(i,j)} = \lambda_1 + \lambda_2 + i\mu_1 + j\mu_2$$

Donde la suma de todas las probabilidades límite debe ser igual a 1:

$$\sum_{i=0}^{O} \sum_{j=0}^{E} \pi_{(i,j)} = 1$$

Finalmente, si resolvemos el sistema, entedemos que los valores de $\pi_{(i,j)}$ representan las probabilidades de equilibrio en el largo plazo para cada estado.

c)

Consideremos k el booleano que representa si el ayudante está en un descanso o durmiendo, entonces el espacio de estados está dado por

$$S = \{(i, j, k) \mid 0 \leq i \leq O,\ 0 \leq j \leq E,\ k \in \{0, 1\}\}.$$

Luego las transiciones posibles son:

- Si $i + j < K$, los correos entran al sistema:

$$q_{(i,j,0)\to(i+1,j,0)} = \lambda_1 \quad \text{para } i < O.$$

$$q_{(i,j,0)\to(i,j+1,0)} = \lambda_2 \quad \text{para } j < E.$$

- Desarrollo normal:

$$q_{(i,j,0)\to(i-1,j,0)} = \mu_1 \quad \text{si } i \leq M.$$

$$q_{(i,j,0)\to(i,j-1,0)} = \mu_2.$$

- Speedrun:

$$q_{(i,j,0)\to(i-1,j,0)} = 3\mu_1 \quad \text{si } i > M.$$

- Si $i + j = K$, hay descanso:

$$q_{(i,j,0)\to(i,j,1)} = \tau.$$

- Si $i + j > J$, hay siesta y elimina correos:

$$q_{(i,j,0)\to(0,0,1)} = \gamma.$$

- Vuelta al trabajo

$$q_{(i,j,1)\to(i,j,0)} = \tau \quad \text{desde descanso.}$$

$$q_{(0,0,1)\to(0,0,0)} = \gamma \quad \text{desde siesta.}$$

d)

Parámetros

- $\lambda_1, \lambda_2, \tau, \gamma$: tasas de llegada de correos, descanso y siesta.
- U: conjutno de valores para μ_1 y μ_2.
- ϵ, δ: Límites mínimos y máximos de permanencia en cada estado.
- α: mínimo de tiempo en estado activo (ayudando).
- β: mínimo de tiempo en estado de descanso o siesta.
- Z_s: desincentivo en cada estado $s \in S$.
- S_{w}: estados activos.
- S_{r}: estados de siesta o descanso.

Variables de decisión

- μ_1, μ_2: Tasas de respuesta para ambos ramos.
- π_s: Probabilidad límite de estar en el estado $s \in S$.

Restricciones

- Restricciones de permanencia mínima y máxima para todos los estados:

$$\epsilon \leq \pi_s \leq \delta, \quad \forall s \in S.$$

- Tiempo mínimo total ayudantía:

$$\sum_{s \in S_{\text{activo}}} \pi_s \geq \alpha.$$

- Tiempos de descanso:
$$\sum_{s \in S_{\text{descanso}}} \pi_s \geq \beta.$$
- Relación tasas de respuesta:
$$\mu_2 \geq 1.2 \cdot \mu_1.$$
- Probabilidades estacionarias ajustadas a la normalización:
$$\sum_{s \in S} \pi_s = 1.$$
- Naturaleza de las variables 1:
$$\mu_1, \mu_2 \in U.$$
- Naturaleza de las variables 2:
$$\pi_s \in [0,1] \quad \forall s \in S$$

Función Objetivo
Minimizar el desincentivo total esperado: $Z = \sum_{s \in S} \pi_s \cdot \mathbb{E}[Z_s]$

Finalemente, para calcular este resultado, lo que podemos hacer es en primer lugar llevarlo a Gurobi y calcular matemáticamente el resultado para valores reales.

Además, podemos hacer simulaciones para evaluar el valor de π_s y así no tener que incurrir en un modelo complejo para encontrar un valor óptimo y suficientemente correcto..

Problema 2 (20 puntos)

Parte 1

i3 2028-02

a)

Para este caso tenemos un sistema M/M/1 , es decir, un servidor con capacidad infinita en el sistema. Para encontrar cantidad promedio de clientes preferenciales y el tiempo promedio de estos en el sistema, buscamos L, tal que:

$$L = \sum_{n=0}^{\infty} n \cdot P_n \tag{1}$$

Y que caso de los clientes preferenciales lo podemos reescribir de la siguiente forma:

$$L_1 = \frac{\lambda_1}{\mu - \lambda_1}$$

Ahora para el tiempo promedio de clientes preferenciales utilizamos la ecuación de Little, es decir

$$L_1 = \lambda_1 \cdot W_1 \tag{2}$$

Siendo W_1 el tiempo promedio que los clientes preferenciales pasan en el sistema, resulta

$$W_1 = \frac{1}{\mu - \lambda_1}$$

b)

Si consideramos un sistema M/M/1 para la fila de clientes preferenciales, con tasas de llegada de clientes igual a λ_1, podemos calcular la proporción del tiempo en la cual se está atendiendo a los clientes preferenciales como la proporción del tiempo en que el servidor de este sistema está ocupado, es decir :

$$P_{preferenciales} = \frac{\lambda_1}{\mu}$$

c)

En primer lugar, tenemos que la atención de clientes normales depende si es que hay clientes preferenciales en el sistema. Por lo tanto, calcularemos la cantidad promedio de clientes normales como la resta entre la cantidad promedio de clientes en total menos la cantidad promedio de clientes preferenciales (L_1 calculada en el insico a). De la siguiente manera:

$$L_2 = L - L_1$$

Para calcular L (total) del sistema general M/M/1 ocupamos ambas tasas de llegadas (de las 2 filas) y las sumamos :

$$L = \frac{\lambda_1 + \lambda_2}{\mu - (\lambda_1 + \lambda_2)}$$

Así, L_2 resulta

$$L_2 = \frac{\lambda_1 + \lambda_2}{\mu - (\lambda_1 + \lambda_2)} - \frac{\lambda_1}{\mu - \lambda_1}$$

Ahora, para calcular el tiempo promedio de los clientes normales recordamos la ecuación de Little y análogamente al inciso a) calculamos W_2:

$$W_2 = \frac{1}{\lambda_2} \cdot (\frac{\lambda_1 + \lambda_2}{\mu - (\lambda_1 + \lambda_2)} - \frac{\lambda_1}{\mu - \lambda_1})$$

$$W_2 = \frac{\lambda_1 + \lambda_2}{\lambda_2(\mu - (\lambda_1 + \lambda_2))} - \frac{\lambda_1}{\lambda_2(\mu - \lambda_1})$$

d)

Para calcular la proporción del tiempo en que los clientes normales están siendo atendidos nos guiamos de acuerdo al inciso anterior y ocupamos la proporción de tiempo en el que el sistema completo está siendo ocupado (lo llamaremos P) y lo restamos con el resultado encontrado en el insico b), es decir, la proporción del tiempo que los clientes preferenciales están siendo atendidos, de esta manera:

$$P = \frac{\lambda_1 + \lambda_2}{\mu}$$

Y

$$P_{normales} = \frac{\lambda_1 + \lambda_2}{\mu} - P_{preferenciales} = \frac{\lambda_1 + \lambda_2}{\mu} - \frac{\lambda_1}{\mu} = \frac{\lambda_2}{\mu}$$

Es decir,

$$P_{normales} = \frac{\lambda_2}{\mu}$$

e)

La cantidad total promedio de clientes en el sistema fue calculada en el inciso c), donde ocupamos ambas tasas de llegadas sumándolas, lo que resultó en :

$$L = \frac{\lambda_1 + \lambda_2}{\mu - (\lambda_1 + \lambda_2)}$$

Ahora, de la misma manera que hemos calculado los tiempos promedios, ocupamos la ecuación de Little, sólo que ahora para calcular el tiempo promedio de un cliente cualquiera ocupamos ambas tasas de llegada:

$$W = \frac{L}{(\lambda_1 + \lambda_2)} = \frac{1}{\mu - (\lambda_1 + \lambda_2)}$$

f)

Con esta nueva forma de atención, notamos que ahora los clientes preferenciales no interrumpen la atención de los normales aunque sí siguen siendo preferentes. Esto quiere decir que los clientes normales serán atendidos completamente si es que llegan al final de la fila y no tendrán que volver a esta. Por lo tanto, esto incide directamente en que los clientes normales pasaran menos tiempo en el sistema por lo que W_2 disminuirá, al igual que L_2 ya que son proporcionales.

Por otro lado, ya que los parámetros del sistema completo no cambian, es que L y W tampoco cambiarán de acuerdo a esta nueva forma de atención.

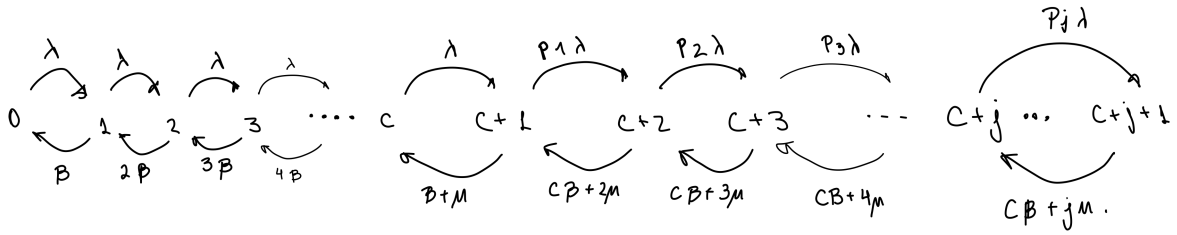
Ya que los valores totales no cambian pero L_2 y W_2 disminuyen, entonces es necesario que L_1 y W_1 aumenten. Esto tiene sentido ya que al no interrumpir la atención de los clientes normales estos pasarán más tiempo en la cola y por lo tanto tambien será mayor la cantidad promedio de estos en el sistema.

Parte 2

I3 2016 02

g)

Definimos esta Cadena de Markov en Tiempo Continuo como X(t) igual a la cantidad de personas en el sistema y $\Omega : (0, 1, 2, 3...)$. De acuerdo al enunciado tenemos el siguiete diagrama:



Notamos que tenemos las siguientes tasas de transición instantáneas:

- $q_{i,i+1} = \lambda \qquad \forall i \in [0, C]$
- $q_{i,i-1} = i \cdot \beta \qquad \forall i \in [1, C]$
- $q_{i,i+1} = P_{i-C} \cdot \lambda \qquad i > C$
- $q_{i,i-1} = C \cdot \beta + (i - C) \cdot \mu \qquad i > C$

h)

Para que el sistema converja dado que tenemos una fila ilimitada, necesitamos que la tasa de llegada sea menor a la tasa de atención de este sistema. Notamos que la atención aumenta mientras hay más clientes en el sistema y por otro lado, la tasa de llegada de clientes se mantiene siempre igual o menor a λ. Es por esto que con grandes cantidades de clientes en el sistema en algún momento se cumplirá que la tasa de atención será mayor a la llegada de los clientes.

De esta forma, las expresiones se describen de la siguiente manera:

$$\pi_i = \begin{cases} \frac{1}{i!}\left(\frac{\lambda}{\beta}\right)^i \pi_0, & \text{si } i \leq C, \\ \frac{\lambda^i}{C! \cdot \beta^C} \prod_{j=0}^{i-C-1} \frac{P_j}{C\beta + (j+1)\mu} \pi_0, & \text{si } i > C. \end{cases}$$

Donde:

$$\pi_0 = \frac{1}{\sum_{i=0}^{C} \frac{1}{i!}\left(\frac{\lambda}{\beta}\right)^i + \sum_{i=C+1}^{\infty} \frac{\lambda^i}{C!\cdot\beta^C} \prod_{j=0}^{i-C-1} \frac{P_j}{C\beta+(j+1)\mu}}.$$

i)

i. Suponiendo las probabilidades estacionarias son conocidas, podemos asociar este comportamiento a la probabilidad de que un caso sea favorable entre los casos totales.

   Para este caso, todos los eventos son la cantidad de clientes que llegan en una hora y los casos favorables son:

$$\sum_{i=C+1}^{\infty} \pi_i \cdot \lambda \cdot (1 - P_{i-C})$$

   De esta manera la esperanza se calcula de la siguiente manera

$$\sum_{i=C+1}^{\infty} \pi_i \cdot (1 - P_{i-C})$$

ii. Para esto tenemos que calcular la tasa de personas que tanto por atención como por pérdida de paciencia deciden salirse del sistema, lo cual se calcula multiplicando la tasa de Poisson con la tasa de salida junto con la decisión de entrada o salida, de forma tal que la expresión es:

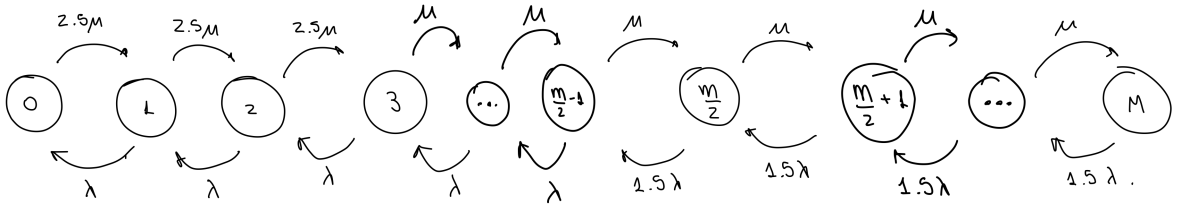
$$\lambda \cdot \left(1 - \sum_{i=C+1}^{\infty} \pi_i \cdot (1 - P_{i-C})\right)$$

Problema 3 (20 puntos)

Tarea 3 2020-01

a)

El sistema lo definimos como X(t): cantidad de pruebas sin destruir en el punto de destrucción en el tiempo t. El espacio de estados es $\Omega = (0, ..., M)$. Notamos que es un proceso de nacimiento y muerte ya que las transiciones desde el estado i sólo pueden ser al estado i 1 o al estado i + 1. Para el sistema tenemos el siguiente grafo:



Las tasas de transición instantáneas son:

$$q_{i,i+1} = \begin{cases} 2.5 \cdot \mu, & \text{si } i \leq 2, \\ \mu, & \text{si } 2 < i < M, \\ 0, & \text{si } i = M \end{cases}$$

$$q_{i,i-1} = \begin{cases} \lambda, & \text{si } 0 < i < \frac{M}{2}, \\ 1.5 \cdot \lambda, & \text{si } i \geq \frac{M}{2}, \\ 0, & \text{si } i = 0 \end{cases}$$

b)

Para la existencia de las probabilidades en el largo plazo necesitamos que las tasas de salida sean mayores a las tasas de llegada, es decir:

$$\frac{\lambda}{\mu} > 1 \quad \forall i \in [3, \frac{M}{2}[$$
$$\frac{\lambda}{2.5 * \mu} > 1 \quad \forall i \in [0, 2]$$
$$\frac{1.5 * \lambda}{\mu} > 1 \quad \forall i \in [\frac{M}{2}, M]$$

Las ecuaciones de equilibrio del sistema son:

$$P_0 \cdot 2.5\mu = P_1 \cdot \lambda,$$

$$P_1 \cdot (2.5\mu + \lambda) = P_0 \cdot 2.5\mu + P_2 \cdot \lambda,$$

$$P_2 \cdot (2.5\mu + \lambda) = P_1 \cdot 2.5\mu + P_3 \cdot \lambda,$$
$$P_3 \cdot (\mu + \lambda) = P_2 \cdot 2.5\mu + P_4 \cdot \lambda,$$
$$P_i \cdot (\mu + \lambda) = P_{i-1} \cdot \mu + P_{i+1} \cdot \lambda, \quad \forall i \in [4, \frac{M}{2} - 1[,$$

$$P_{(\frac{M}{2}-1)} \cdot (\mu + \lambda) = P_{\frac{M}{2}-2} \cdot \mu + P_{\frac{M}{2}} \cdot 1.5\lambda,$$

$$P_i \cdot (\mu + 1.5\lambda) = P_{i-1} \cdot \mu + P_{i+1} \cdot 1.5\lambda, \quad \forall i \in [\frac{M}{2}, M[,$$

$$P_M \cdot (1.5\lambda) = P_{M-1} \cdot \mu.$$

$$\sum_{i=0}^{M} P_i = 1.$$

c)

Resolvemos el sistema de ecuaciones mediante la siguiente función en Pyhton:

```python
#c)
import numpy as np

def solucion(m,lambda_,mu):
    matriz=[]

    for i in range(m+2):
        fila_aux=[]
        if i==0:
            fila_aux.extend([-2.5*mu,lambda_])
            for j in range(m-1):
                fila_aux.extend([0])
            matriz.append(fila_aux)

        elif i==1:
            fila_aux.extend([2.5*mu,-(2.5*mu+ lambda_), lambda_])
            for j in range(m-2):
                fila_aux.extend([0])
            matriz.append(fila_aux)

        elif i==2:
            fila_aux.extend([0,2.5*mu,-(2.5*mu+lambda_),lambda_])
            for j in range(m-3):
                fila_aux.extend([0])
            matriz.append(fila_aux)

        elif i==3:
            fila_aux.extend([0,0,2.5*mu,-(mu+lambda_),lambda_])
            for j in range(m-4):
                fila_aux.extend([0])
            matriz.append(fila_aux)

        elif i>=4 and i<(m//2 -1):
            for j in range(0,i-1):
                fila_aux.extend([0])
            fila_aux.extend([mu,-(mu+lambda_),lambda_])
            for j in range(i+2,m+1):
                fila_aux.extend([0])
            matriz.append(fila_aux)
```

```
def solucion(m,lambda_,mu):

        elif i==(m//2 -1):
            for j in range(0,i-1):
                fila_aux.extend([0])
            fila_aux.extend([mu,-(mu+lambda_),1.5*lambda_])
            for j in range(i+2,m+1):
                fila_aux.extend([0])
            matriz.append(fila_aux)

        elif i>=m//2 and i<m:
            for j in range(0,i-1):
                fila_aux.extend([0])
            fila_aux.extend([mu,-(mu+1.5*lambda_),1.5*lambda_])
            for j in range(i+2,m+1):
                fila_aux.extend([0])
            matriz.append(fila_aux)

        elif i==m:
            for j in range(0, i-1):
                fila_aux.extend([0])
            fila_aux.extend([mu,-1.5*lambda_])
            matriz.append(fila_aux)

        elif i== m+1:
            for j in range(0,m+1):
                fila_aux.extend([1])
            matriz.append(fila_aux)

    matriz.remove(matriz[1])

    b=[0 for j in range(m)]
    b.extend([1])

    A=np.array(matriz)
    B=np.array(b)
    x=np.linalg.solve(A,B)

    return x
```

Ahora, para calcular la proporción del tiempo en el largo plazo que Aníbal se encuentra destruyendo a una tasa de destrucción aumentada en un 50 %, es decir,1.5λ, utilizamos la siguiente función:

```
print("A",solucion(24,0.5,0.1))
print("B",solucion(24,0.5,0.3))
print("C",solucion(24,0.5,0.5))
print("D",solucion(24,0.5,0.7))

def sumar(x):
    p=0
    for i in range(len(x)):
        if i>=((len(x)-1)//2):
            p+=x[i]

    return p

print("A",sumar(solucion(24,0.5,0.1)))
print("B",sumar(solucion(24,0.5,0.3)))
print("C",sumar(solucion(24,0.5,0.5)))
print("D",sumar(solucion(24,0.5,0.7)))
```

Mostraremos un vector obtenido, en este caso es el vector obtenido para la primera iteración ($M = 24, \lambda = 0.5, \mu = 0.1$) fue el siguiente:

```
A [5.24590172e-01 2.62295086e-01 1.31147543e-01 6.55737716e-02
 1.31147543e-02 2.62295086e-03 5.24590172e-04 1.04918034e-04
 2.09836069e-05 4.19672138e-06 8.39344276e-07 1.67868855e-07
 2.23825140e-08 2.98433520e-09 3.97911360e-10 5.30548481e-11
 7.07397974e-12 9.43197299e-13 1.25759640e-13 1.67679520e-14
 2.23572693e-15 2.98096924e-16 3.97462565e-17 5.29950087e-18
 7.06600116e-19]
```

Y los resultados obtenidos a través de la función definida sumar(x) para cada μ, fueron:

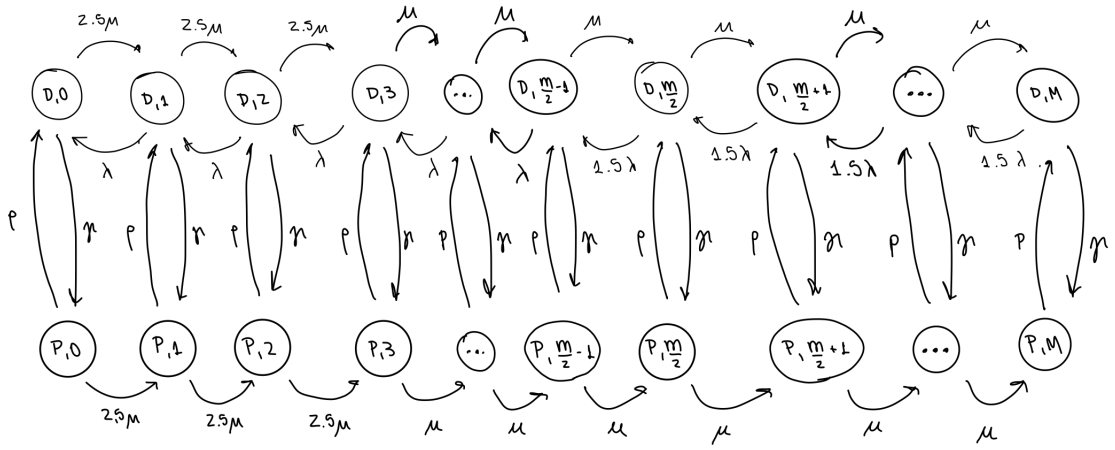
- $\mu_{0.1} = 2.5825977718141643e - 08$
- $\mu_{0.3} = 0.002875978467327154$
- $\mu_{0.5} = 0.17132520258010486$
- $\mu_{0.7} = 0.711776680928138$

Analizando los resultados notamos que,

- Para $\mu = 0.1$, $P = 2.58 \times 10^{-8}$:
  Con una tasa de llegada baja ($\mu = 0.1$), la proporción del tiempo en que Aníbal encuentra destruyendo a una tasa de destrucción de 1.5λ es muy baja. Esto quiere decir, que con tasas de llegadas muy bajas es muy poco probable que las pruebas se acumulen y superen los $\frac{M}{2}$. Esto debido a que la capacidad de destrucción es mucho mayor que la llegada.
- Para $\mu = 0.3$, $P = 0.0029$:
  Con una tasa de llegada moderadamente baja ($\mu = 0.3$), la proporción del tiempo en que Aníbal encuentra destruyendo pruebas a una tasa de 1.5λ sigue siendo muy baja. Esto implica que, aunque las pruebas se acumulan un poco más que en el caso de $\mu = 0.1$, sigue siendo poco probable que las pruebas acumuladas superen $\frac{M}{2}$.
- Para $\mu = 0.5$, $P = 0.1713$:
  Con una tasa de llegada intermedia ($\mu = 0.5$), la proporción del tiempo en que Aníbal encuentra destruyendo pruebas a una tasa de 1.5λ aumenta considerablemente. Esto significa que el flujo de pruebas entrantes empieza a utilizar mejor la capacidad del sistema, y la acumulación ocasional de pruebas supera más frecuentemente $\frac{M}{2}$. En este punto, el sistema se acerca a un equilibrio donde las tasas de llegada y destrucción están mejor balanceadas.
- Para $\mu = 0.7$, $P = 0.7118$:
  Con una tasa de llegada alta ($\mu = 0.7$), la proporción del tiempo en que Aníbal encuentra destruyendo pruebas a una tasa de 1.5λ es significativamente mayor. Esto indica que, con tasas de llegada altas, el sistema opera de manera eficiente y constantemente en el límite de su capacidad. Es mucho más probable que las pruebas acumuladas superen $\frac{M}{2}$, pero gracias a la capacidad de destrucción de Aníbal, el sistema se mantiene estable y logra manejar el flujo elevado de pruebas.

d)

El nuevo sistema lo definimos como X(t)=(estado de Aníbal, cantidad de pruebas), con el siguiente espacio de estados $\Omega = \{(P, D), (0, ..., M)\}$, donde P siginifca que Aníbal esta en pausa y D que está destruyendo. Tenemos el siguiente grafo para esta cadena:



Las tasas de transición instantáneas son:

$$q_{(D,i),(D,i+1)} = \begin{cases} 2.5 \cdot \mu, & \text{si } i \leq 2, \\ \mu, & \text{si } 2 < i < M, \\ 0, & \text{si } i = M \end{cases}$$

$$q_{(D,i),(D,i-1)} = \begin{cases} \lambda, & \text{si } 0 < i < \frac{M}{2}, \\ 1.5 \cdot \lambda, & \text{si } i \geq \frac{M}{2}, \\ 0, & \text{si } i = 0 \end{cases}$$

$$q_{(P,i),(P,i+1)} = \begin{cases} 2.5 \cdot \mu, & \text{si } i \leq 2, \\ \mu, & \text{si } 2 < i < M, \\ 0, & \text{si } i = M \end{cases}$$

$$q_{(P,i),(P,i-1)} = 0 \quad \forall i \in [0, M]$$

$$q_{(D,i),(P,i)} = \gamma \quad \forall i \in [0, M]$$
$$q_{(P,i),(D,i)} = \rho \quad \forall i \in [0, M]$$

e)

Para la existencia de las probabilidades en el largo plazo necesitamos que las tasas de salida sean mayores a las tasas de llegada, es decir:

$$\frac{\lambda}{\mu} > 1 \quad \forall i \in [3, \frac{M}{2}[$$
$$\frac{\lambda}{2.5 * \mu} > 1 \quad \forall i \in [0, 2]$$
$$\frac{1.5 * \lambda}{\mu} > 1 \quad \forall i \in [\frac{M}{2}, M]$$

Las ecuaciones de equilibrio son las siguientes:

$$P_{D,0} \cdot (2.5\mu + \gamma) = P_{D,1} \cdot \lambda + P_{P,0} \cdot \rho$$

$$P_{D,1} \cdot (2.5\mu + \lambda + \gamma) = P_{D,0} \cdot 2.5\mu + P_{D,2} \cdot \lambda + P_{P,1} \cdot \rho$$

$$P_{D,2} \cdot (2.5\mu + \lambda + \gamma) = P_{D,1} \cdot 2.5\mu + P_{D,3} \cdot \lambda + P_{P,2} \cdot \rho$$

$$P_{D,3} \cdot (\mu + \lambda + \gamma) = P_{D,2} \cdot 2.5\mu + P_{D,4} \cdot \lambda + P_{P,3} \cdot \rho$$

$$P_{D,i} \cdot (\mu + \lambda + \gamma) = P_{D,i-1} \cdot \mu + P_{M,i+1} \cdot \lambda + P_{P,i} \cdot \rho \quad \forall i \in [4, \frac{M}{2} - 1[$$

$$P_{D,\frac{M}{2}-1} \cdot (\mu + \lambda + \gamma) = P_{D,\frac{M}{2}-2} \cdot \mu + P_{D,\frac{M}{2}} \cdot 1.5\lambda + P_{P,\frac{M}{2}-1} \cdot \rho$$

$$P_{D,i} \cdot (\mu + 1.5\lambda + \gamma) = P_{D,i-1} \cdot \mu + P_{D,i+1} \cdot 1.5\lambda + P_{P,i} \cdot \rho \quad \forall i \in [\frac{M}{2}, M[$$

$$P_{D,M} \cdot (1.5\lambda + \gamma) = P_{D,M-1} \cdot \mu + P_{P,M} \cdot \rho$$

$$P_{P,0} \cdot (\rho + 2.5\mu) = P_{D,0} \cdot \gamma$$

$$P_{P,1} \cdot (\rho + 2.5\mu) = P_{D,1} \cdot \gamma + P_{P,0} \cdot 2.5\mu$$

$$P_{P,2} \cdot (\rho + 2.5\mu) = P_{D,2} \cdot \gamma + P_{P,1} \cdot 2.5\mu$$

$$P_{P,3} \cdot (\rho + \mu) = P_{D,3} \cdot \gamma + P_{P,2} \cdot 2.5\mu$$

$$P_{P,i} \cdot (\rho + \mu) = P_{D,i} \cdot \gamma + P_{P,i-1} \cdot \mu \quad \forall i \in [4, M-1]$$

$$P_{P,M} \cdot (\rho) = P_{D,M} \cdot \gamma + P_{P,M-1} \cdot \mu \quad \forall i \in [4, M-1]$$

$$\sum_{i=0}^{M} P_{D,i} + P_{P,i} = 1$$

Con el siguiente código en Pyhton pudimos calcular, con los valores solicitados, la proporción del tiempo en el largo plazo que Aníbal se encuentra destruyendo a una tasa de destrucción aumentada en un 50 %, es decir se encuentra en cualquier estado entre $(D, \frac{M}{2})$ y (D, M):

```python
import numpy as np
# e)
m = 30
lambda_ = 0.7
mu = 0.8
gamma = 0.2
rho = 0.5

# 62 nodos     hasta el 31 son (D,i) y desde el 32 parte (P,0) hasta el 62 : (P,30)
matriz = []
for i in range((m+1)*2+1):
    fila_aux = []
    if i >= 0 and i <= 30:
        if i == 0:
            fila_aux.extend([-(2.5*mu+gamma), lambda_])
            fila_aux.extend([0 for j in range(29)])
            fila_aux.extend([rho])  # nodo 32
            fila_aux.extend([0 for j in range(30)])
            matriz.append(fila_aux)

        elif i == 1:
            fila_aux.extend([2.5*mu, -(2.5*mu+lambda_+gamma), lambda_])
            fila_aux.extend([0 for j in range(29)])
            fila_aux.extend([rho])  # nodo 33
            fila_aux.extend([0 for j in range(29)])
            matriz.append(fila_aux)

        elif i == 2:
            fila_aux.extend([0, 2.5*mu, -(2.5*mu+lambda_+gamma), lambda_])
            fila_aux.extend([0 for j in range(29)])
            fila_aux.extend([rho])  # nodo 34
            fila_aux.extend([0 for j in range(28)])
            matriz.append(fila_aux)
```

```python
    elif i == 3:
        fila_aux.extend([0, 0, 2.5*mu, -(mu+lambda_+gamma), lambda_])
        fila_aux.extend([0 for j in range(29)])
        fila_aux.extend([rho])  # nodo 35
        fila_aux.extend([0 for j in range(27)])
        matriz.append(fila_aux)

    elif i >= 4 and i < (m//2-1):
        for j in range(0, i-1):
            fila_aux.extend([0])

        fila_aux.extend([mu, -(mu+lambda_+gamma), lambda_])
        fila_aux.extend([0 for j in range(29)])
        fila_aux.extend([rho])  # nodo 29+i+2
        fila_aux.extend([0 for j in range(30-i)])
        matriz.append(fila_aux)

    elif i >= m//2 and i < m:
        for j in range(0, i-1):
            fila_aux.extend([0])

        fila_aux.extend([mu, -(mu+1.5*lambda_+gamma), 1.5*lambda_])
        fila_aux.extend([0 for j in range(29)])
        fila_aux.extend([rho])
        fila_aux.extend([0 for j in range(30-i)])
        matriz.append(fila_aux)

    elif i == m:
        for j in range(0, i-1):
            fila_aux.extend([0])

        fila_aux.extend([mu, -(1.5*lambda_+gamma)])
        fila_aux.extend([0 for j in range(i)])
        fila_aux.extend([rho])
        matriz.append(fila_aux)
```

```
    elif i > 30:
        if i == 31:  # (P,0)
            fila_aux.extend([0 for j in range(i - 31)])
            fila_aux.extend([gamma])
            fila_aux.extend([0 for j in range(61-i)])
            fila_aux.extend([-(rho+2.5*mu)])
            fila_aux.extend([0 for j in range(61-i)])
            matriz.append(fila_aux)

        elif i == 32:  # (P,1)
            fila_aux.extend([0 for j in range(i - 31)])
            fila_aux.extend([gamma])
            fila_aux.extend([0 for j in range(61-i)])
            fila_aux.extend([2.5*mu, -(rho+2.5*mu)])
            fila_aux.extend([0 for j in range(61-i)])
            matriz.append(fila_aux)

        elif i == 33:  # (P,2)
            fila_aux.extend([0 for j in range(i - 31)])
            fila_aux.extend([gamma])
            fila_aux.extend([0 for j in range(61-i)])
            fila_aux.extend([0, 2.5*mu, -(rho+2.5*mu)])
            fila_aux.extend([0 for j in range(61-i)])
            matriz.append(fila_aux)

        elif i == 34:  # (P,3)
            fila_aux.extend([0 for j in range(i - 31)])
            fila_aux.extend([gamma])
            fila_aux.extend([0 for j in range(61-i)])
            fila_aux.extend([0, 0, mu, -(rho+mu)])
            fila_aux.extend([0 for j in range(61-i)])
            matriz.append(fila_aux)

        elif i > 34 and i < 61:
            fila_aux.extend([0 for j in range(i - 31)])
            fila_aux.extend([gamma])
            fila_aux.extend([0 for j in range(61-i)])
            fila_aux.extend([0 for j in range(i-32)])
            fila_aux.extend([mu, -(rho+mu)])
            fila_aux.extend([0 for j in range(61-i)])
            matriz.append(fila_aux)
```

```
        elif i == 61:
            fila_aux.extend([0 for j in range(i - 31)])
            fila_aux.extend([gamma])
            fila_aux.extend([0 for j in range(i-32)])
            fila_aux.extend([mu, -(rho)])
            matriz.append(fila_aux)

        elif i == 62:
            for j in range(0, 62):
                fila_aux.extend([1])
            matriz.append(fila_aux)


b = [0 for j in range(61)]
b.extend([1])
A = np.array(matriz)
B = np.array(b)
x = np.linalg.solve(A, B)

print(x)

def sumar_probabilidades(x):
    suma = 0
    for i in range(len(x)):
        if i >= 15 and i <= 30:
            suma += x[i]
    return suma

print(sumar_probabilidades(x))
```

Con la función "sumar_probabilidades"se obtuvo que la proporción en el largo plazo en que Aníbal se encuentra destruyendo a una tasa de destrucción aumentada en un 50 % es 0.6198235828341643.

i. Se obtuvo que la probabilidad en el largo plazo de que Aníbal se encuentre sobornando árbitros es $p = 0.28568540783522106$. Se ocupó el siguiente código

```
# i)
p = 0
for i in range(62):
    if i >= 31:
        p += x[i]
print(p)
```

ii. Se obtuvo que la tasa efectiva, en el largo plazo, de entrada de pruebas al punto de destrucciónn es $t = 0.7167246051466737$. Mediante el siguiente código:

```
# ii)
tasa = 0
for i in range(30):
    if i <= 2:
        tasa += (2.5*mu)*x[i]
    else:
        tasa += mu*x[i]
for i in range(30):
    if (i+31) <= 33:
        tasa += (2.5*mu)*x[i+31]
    else:
        tasa += (mu)*x[i+31]
print(tasa)
```

iii. Se obtuvo que la tasa efectiva, en el largo plazo, de destrucción de pruebas es $t = 0.7169584685073024$. Mediante el siguiente código:

```
# iii)
tasa_2 = 0
for i in range(31):
    if i < 15:
        tasa_2 += lambda_*x[i]
    else:
        tasa_2 += 1.5*lambda_*x[i]
print(tasa_2)
```

iv. Se obtuvo que el valor esperado, en el largo plazo, de pruebas en el punto de destrucción es 22.268 pruebas. Se ocupó el siguiente código:

```
# iv)
p_destruccion = 0
for i in range(31):
    p_destruccion += i*x[i]

for j in range(31):
    p_destruccion += j*x[j+31]
print(p_destruccion)
```

v. El tiempo promedio que pasa una prueba en el sistema es 4276 minutos. Calculado según el siguiente código:

```
# v)

w = 1/(tasa_2-tasa)
print(w)
```