UNIVERSIDAD NACIONAL DEL ALTIPLANO

INGENIERÍA ESTADÍSTICA E INFORMÁTICA

METODOS DE OPTIMIZACION

Actividad N° 2 - Restricciones

Desarrolle los siguientes enunciados de acuerdo con el objetivo planteado en cada problema, ya sea maximización o minimización. Además, implemente una solución computacional utilizando el lenguaje de programación o framework de su preferencia, con el propósito de generar una solución genérica que permita la visualización gráfica de los resultados en todos los casos.

Andree Alessandro Chili Lima
[229071]

30/09/2024

Problema 1

Un algoritmo necesita procesar datos en lotes. Cada lote requiere x MB de memoria, pero la capacidad total de memoria disponible es de 1024 MB. El algoritmo puede procesar un máximo de 8 lotes. El objetivo es maximizar la cantidad de datos procesados, pero cada lote más allá del quinto reduce su eficiencia en un 20%.

Restricciones

- **Capacidad total de memoria disponible**:

$$n \cdot x \leq 1024$$

- **Número máximo de lotes**:

$$n \leq 8$$

Función

- Para los primeros 5 lotes: $5 \cdot x$.
- Para los lotes adicionales (del 6 al 8), la eficiencia es del 80%, es decir, $0.8 \cdot x$ por lote.

Entonces, si procesamos n lotes, donde $n \leq 5$, los datos procesados son $n \cdot x$. Si $n > 5$, los datos procesados son:

$$D(n) = \begin{cases} n \cdot x & \text{si } n \leq 5 \\ 5 \cdot x + (n-5) \cdot 0.8 \cdot x & \text{si } n > 5 \end{cases}$$

Solución

El algoritmo deberá probar distintos valores de x y n dentro de las restricciones, evaluando en cada caso cuántos datos pueden procesarse de manera óptima.

Ejemplo: $n = 5$

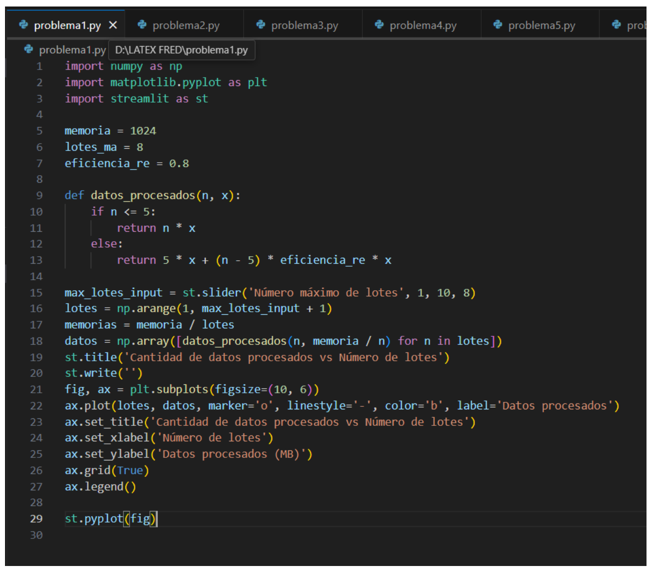
$$5x \leq 1024 \implies x \leq 204.8\,\text{MB por lote}$$

Si $x = 204.8$, se procesarán:

$$5 \cdot 204.8 = 1024\,\text{MB de datos}$$

Rpta.

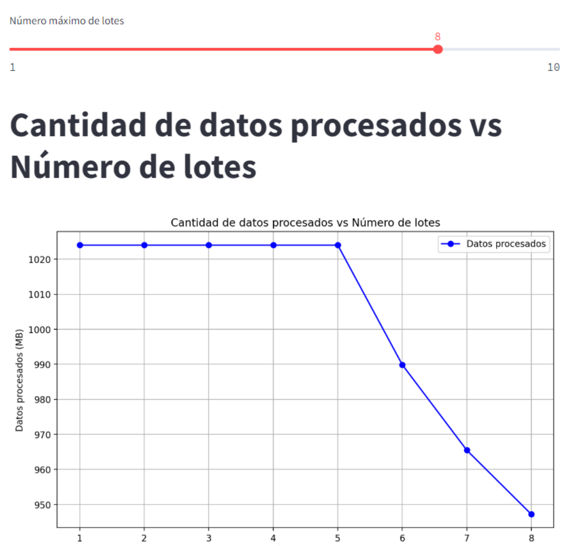
La solución óptima es procesar 5 lotes de 204.8 MB, lo que maximiza los datos procesados sin perder eficiencia. Alternativamente, procesar 8 lotes de 128 MB da una menor cantidad de datos procesados debido a la reducción de eficiencia.

[h] 

problema1.py problema2.py problema3.py problema4.py problema5.py

problema1.py D:\LATEX FRED\problema1.py

```python
import numpy as np
import matplotlib.pyplot as plt
import streamlit as st

memoria = 1024
lotes_ma = 8
eficiencia_re = 0.8

def datos_procesados(n, x):
    if n <= 5:
        return n * x
    else:
        return 5 * x + (n - 5) * eficiencia_re * x

max_lotes_input = st.slider('Número máximo de lotes', 1, 10, 8)
lotes = np.arange(1, max_lotes_input + 1)
memorias = memoria / lotes
datos = np.array([datos_procesados(n, memoria / n) for n in lotes])
st.title('Cantidad de datos procesados vs Número de lotes')
st.write('')
fig, ax = plt.subplots(figsize=(10, 6))
ax.plot(lotes, datos, marker='o', linestyle='-', color='b', label='Datos procesados')
ax.set_title('Cantidad de datos procesados vs Número de lotes')
ax.set_xlabel('Número de lotes')
ax.set_ylabel('Datos procesados (MB)')
ax.grid(True)
ax.legend()

st.pyplot(fig)
```

[h] 

Problema 2

Un sistema distribuido tiene 20 nodos. Cada nodo puede procesar x peticiones por segundo. El sistema en su conjunto no puede procesar más de 400 peticiones por segundo debido a limitaciones de red. Maximiza el número de peticiones procesadas sin exceder la capacidad de la red.

Restricciones

- **Capacidad total de procesamiento**:

$$n \cdot x \leq 400$$

- **Número de nodos**:

$$n = 20$$

Función

Maximizar el número total de peticiones procesadas por el sistema, P, donde:

$$P = n \cdot x$$

Sujeto a las restricciones:

$$n \cdot x \leq 400$$
$$n = 20$$

Solución

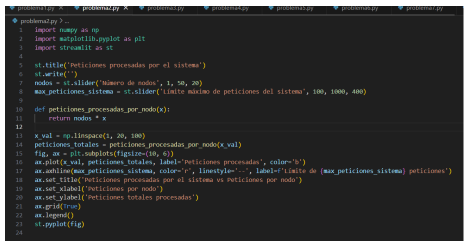
Sustituyendo $n = 20$ en la restricción:

$$20 \cdot x \leq 400$$
$$x \leq \frac{400}{20} = 20 \, \text{peticiones/segundo por nodo}$$

Por lo tanto, el valor máximo de x es 20 peticiones por segundo por nodo. Esto significa que cada nodo puede procesar hasta 20 peticiones por segundo sin exceder el límite de 400 peticiones por segundo para el sistema completo.

Rpta

Cada nodo debe procesar $x = 20$ peticiones por segundo, lo que maximiza el número total de peticiones procesadas por el sistema sin exceder la capacidad de la red.

[h] 

[h] 

Problema 3

Un script de Python tarda $5x+2$ segundos en procesar x datos. Por cada dato adicional, el tiempo de ejecución crece linealmente. Sin embargo, el sistema tiene un límite de tiempo de ejecución de 50 segundos. ¿Cuál es el número máximo de datos que puede procesar el script?

Restricciones

$$5x + 2 \leq 50$$

Función

Maximizar x, es decir, encontrar el número máximo de datos que se pueden procesar sin exceder el tiempo permitido.

Solución

Despejamos x de la ecuación:

$$5x + 2 \leq 50$$

$$5x \leq 50 - 2$$

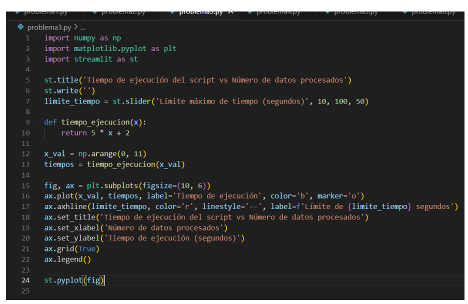
$$5x \leq 48$$

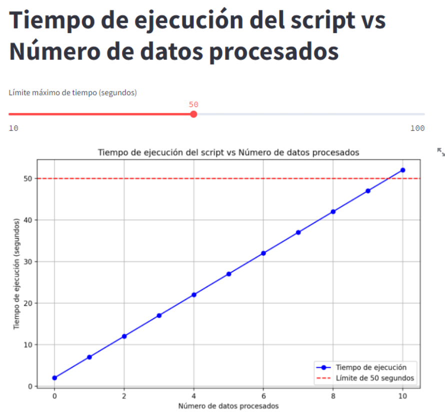
$$x \leq \frac{48}{5} = 9.6$$

Dado que x debe ser un número entero, el valor máximo de x es 9.

Rpta

El número máximo de datos que el script puede procesar sin exceder los 50 segundos es $x = 9$.

[h] 



[h]

Problema 4

Un servidor web procesa x peticiones por segundo, y el uso de CPU sigue la fórmula $2x^2 + 10x$.La CPU no puede exceder el 80 % de uso. Minimiza el uso de CPU sin caer por debajo del umbral de procesamiento de 10 peticiones por segundo.

Restricciones

- El uso de CPU no puede superar el 80%:

$$2x^2 + 10x \leq 80$$

- El servidor debe procesar al menos 10 peticiones por segundo:

$$x \geq 10$$

Solución

Resolviendo la ecuación cuadrática:

$$2x^2 + 10x - 80 \leq 0$$

Simplificamos:

$$x^2 + 5x - 40 \leq 0$$

Resolviendo la ecuación cuadrática:

$$x = \frac{-5 \pm \sqrt{5^2 - 4(1)(-40)}}{2(1)}$$

$$x = \frac{-5 \pm \sqrt{185}}{2}$$

$$x_1 = 4.3 \quad (\text{no factible porque } x \geq 10)$$

$$x_2 = -9.3 \quad (\text{no factible porque } x \geq 10)$$

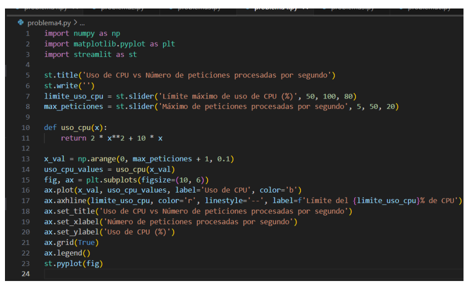
Como las soluciones no cumplen la restricción de $x \geq 10$, evaluamos directamente en $x = 10$:

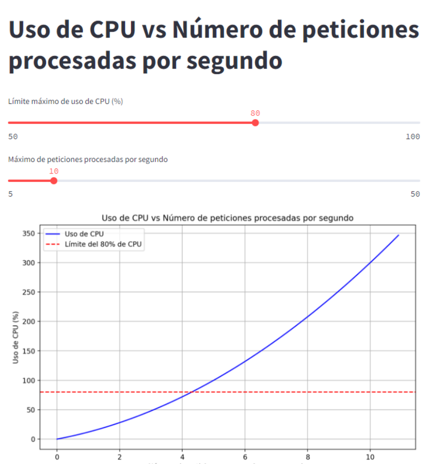
$$U(10) = 2(10)^2 + 10(10) = 300$$

El uso de CPU excede el límite del 80%, por lo que no existe una solución factible que cumpla con ambas restricciones.

Rpta

Este problema no tiene una solución factible que satisfaga ambas restricciones de procesamiento mínimo y límite de uso de CPU. Se necesita ajustar alguna de las restricciones para que el sistema pueda funcionar correctamente.

[h] 

[h] 

Problema 5

Durante el entrenamiento de un modelo de machine learning, el batch size x afecta el tiempo de entrenamiento $T(x) = \frac{1000}{x} + 0.1x$. El tamaño del lote debe estar entre 16 y 128. Encuentra el batch size que minimiza el tiempo de entrenamiento.

Restricciones

El tamaño del lote debe estar entre 16 y 128:

$$16 \leq x \leq 128$$

Función

Minimizar $T(x) = \frac{1000}{x} + 0.1x$

Solución

Para encontrar el valor de x que minimiza $T(x)$, tomamos la derivada de $T(x)$:

$$T'(x) = -\frac{1000}{x^2} + 0.1$$

Igualamos a cero:

$$-\frac{1000}{x^2} + 0.1 = 0$$
$$\frac{1000}{x^2} = 0.1$$
$$x^2 = 10000$$
$$x = \sqrt{10000} = 100$$

Evaluación en los extremos

Evaluamos en los extremos del intervalo:

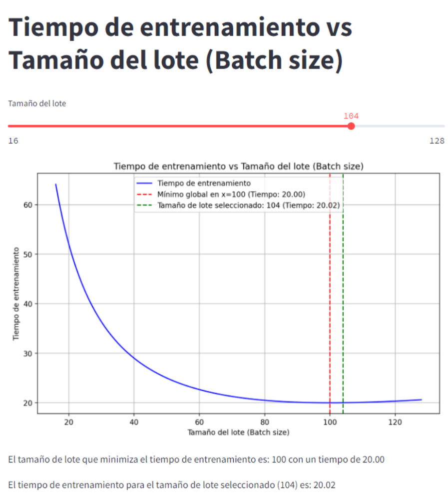
- Para $x = 16$:
$$T(16) = \frac{1000}{16} + 0.1(16) = 64.1$$
- Para $x = 128$:
$$T(128) = \frac{1000}{128} + 0.1(128) = 20.61$$
- Para $x = 100$:
$$T(100) = \frac{1000}{100} + 0.1(100) = 20$$

Rpta

El tamaño del lote que minimiza el tiempo de entrenamiento es $x = 100$, con un tiempo mínimo de entrenamiento de $T(100) = 20$ segundos.

```python
import numpy as np
import matplotlib.pyplot as plt
import streamlit as st

st.title('Tiempo de entrenamiento vs Tamaño del lote (Batch size)')
st.write('')
batch_size = st.slider('Tamaño del lote', 16, 128, 16)

def tiempo_entrenamiento(x):
    return (1000 / x) + 0.1 * x

x_val = np.arange(16, 129, 1)
tiempos = tiempo_entrenamiento(x_val)
min_index = np.argmin(tiempos)
batch_size_min = x_val[min_index]
tiempo_min = tiempos[min_index]
tiempo_actual = tiempo_entrenamiento(batch_size)

fig, ax = plt.subplots(figsize=(10, 6))
ax.plot(x_val, tiempos, label='Tiempo de entrenamiento', color='b')
ax.axvline(x=batch_size_min, color='r', linestyle='--', label=f'Mínimo global en x={batch_size_min} (Tiempo: {tiempo_min:.2f})')
ax.axvline(x=batch_size, color='g', linestyle='--', label=f'Tamaño de lote seleccionado: {batch_size} (Tiempo: {tiempo_actual:.2f})')

ax.set_title('Tiempo de entrenamiento vs Tamaño del lote (Batch size)')
ax.set_xlabel('Tamaño del lote (Batch size)')
ax.set_ylabel('Tiempo de entrenamiento')
ax.grid(True)
ax.legend()
st.pyplot(fig)
st.write(f'El tamaño de lote que minimiza el tiempo de entrenamiento es: {batch_size_min} con un tiempo de {tiempo_min:.2f} ')
st.write(f'El tiempo de entrenamiento para el tamaño de lote seleccionado ({batch_size}) es: {tiempo_actual:.2f}')
```

[h]



[h]

Problema 6

Un sistema de transmisión de datos tiene un ancho de banda total de 1000 Mbps. Cada archivo que se transmite utiliza x Mbps. El sistema puede transmitir un máximo de 50 archivos a la vez, y cada archivo adicional más allá de 30 reduce el ancho de banda disponible en un 5 %. Maximiza el número de archivos transmitidos.

Restricciones

Existen dos casos para las restricciones del problema:

- **Si** $n \leq 30$: No hay penalización en el ancho de banda, por lo que la restricción es:
$$n \times x \leq 1000$$
- **Si** $n > 30$: Se aplica una penalización del 5% por cada archivo adicional más allá de 30. El ancho de banda disponible es:
$$A = 1000 \times (1 - 0.05 \times (n - 30))$$
En este caso, la restricción se vuelve:
$$n \times x \leq A$$

Función

Queremos maximizar el número de archivos n sujetos a las restricciones del ancho de banda.

sujeta a:

$$\begin{aligned} &n \times x \leq 1000, \quad \text{si } n \leq 30 \\ &n \times x \leq 1000 \times (1 - 0.05 \times (n - 30)), \quad \text{si } n > 30 \end{aligned}$$

Solucion

Caso 1: $n \leq 30$

Cuando $n \leq 30$, la restricción es:

$$n \times x \leq 1000$$

Esto implica que:
$$n \leq \frac{1000}{x}$$

Entonces: Si $x = 20$ Mbps (ancho de banda por archivo), tenemos:

$$n \leq \frac{1000}{20} = 50$$

Pero dado que solo podemos transmitir hasta 30 archivos sin penalización, podemos transmitir hasta 30 archivos en este caso.

Caso 2: $n > 30$

Cuando $n > 30$, la penalización reduce el ancho de banda disponible. La nueva restricción es:
$$n \times x \leq 1000 \times (1 - 0.05 \times (n - 30))$$

Simplificamos el término de la derecha:

$$A = 1000 \times (1 - 0.05 \times (n - 30)) = 1000 \times (1.5 - 0.05 \times n)$$

Por lo tanto, la restricción se convierte en:

$$n \times x \leq 1000 \times (1.5 - 0.05 \times n)$$

Entonces: Si $x = 20$ Mbps (ancho de banda por archivo):

- Para $n = 35$ (más de 30 archivos):

  $$35 \times 20 \leq 1000 \times (1.5 - 0.05 \times 35)$$

  $$700 \leq 1000 \times (1.5 - 1.75) = 1000 \times 0.75 = 750$$

  Aquí, el ancho de banda disponible es suficiente, por lo que $n = 35$ es una solución válida.

- Para $n = 40$:

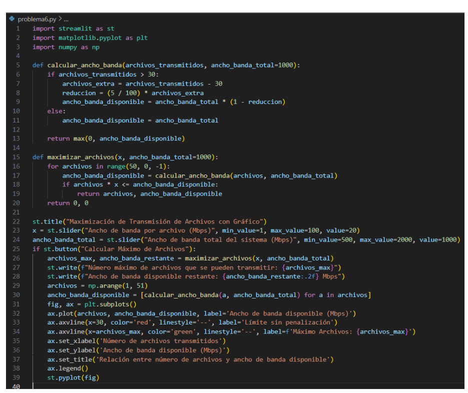
  $$40 \times 20 \leq 1000 \times (1.5 - 0.05 \times 40)$$

  $$800 \leq 1000 \times (1.5 - 2) = 1000 \times 0.5 = 500$$

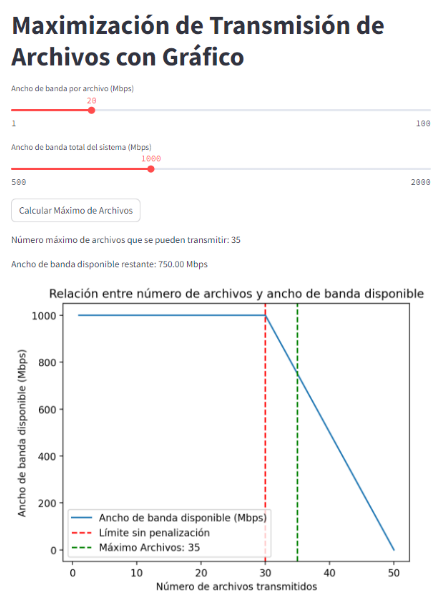
  En este caso, no se cumple la restricción porque el ancho de banda requerido (800 Mbps) excede el disponible (500 Mbps). Entonces, $n = 40$ no es una solución válida.

Rpta

Si $x = 20$ Mbps, el número máximo de archivos que se pueden transmitir es 35 antes de que el ancho de banda disponible se reduzca demasiado.

[h] 

```
problema6.py > ...
import streamlit as st
import matplotlib.pyplot as plt
import numpy as np

def calcular_ancho_banda(archivos_transmitidos, ancho_banda_total=1000):
    if archivos_transmitidos > 30:
        archivos_extra = archivos_transmitidos - 30
        reduccion = (5 / 100) * archivos_extra
        ancho_banda_disponible = ancho_banda_total * (1 - reduccion)
    else:
        ancho_banda_disponible = ancho_banda_total

    return max(0, ancho_banda_disponible)

def maximizar_archivos(x, ancho_banda_total=1000):
    for archivos in range(50, 0, -1):
        ancho_banda_disponible = calcular_ancho_banda(archivos, ancho_banda_total)
        if archivos * x <= ancho_banda_disponible:
            return archivos, ancho_banda_disponible
    return 0, 0

st.title("Maximización de Transmisión de Archivos con Gráfico")
x = st.slider("Ancho de banda por archivo (Mbps)", min_value=1, max_value=100, value=20)
ancho_banda_total = st.slider("Ancho de banda total del sistema (Mbps)", min_value=500, max_value=2000, value=1000)
if st.button("Calcular Máximo de Archivos"):
    archivos_max, ancho_banda_restante = maximizar_archivos(x, ancho_banda_total)
    st.write(f"Número máximo de archivos que se pueden transmitir: {archivos_max}")
    st.write(f"Ancho de banda disponible restante: {ancho_banda_restante:.2f} Mbps")
    archivos = np.arange(1, 51)
    ancho_banda_disponible = [calcular_ancho_banda(a, ancho_banda_total) for a in archivos]
    fig, ax = plt.subplots()
    ax.plot(archivos, ancho_banda_disponible, label='Ancho de banda disponible (Mbps)')
    ax.axvline(x=30, color='red', linestyle='--', label='Límite sin penalización')
    ax.axvline(x=archivos_max, color='green', linestyle='--', label=f'Máximo Archivos: {archivos_max}')
    ax.set_xlabel('Número de archivos transmitidos')
    ax.set_ylabel('Ancho de banda disponible (Mbps)')
    ax.set_title('Relación entre número de archivos y ancho de banda disponible')
    ax.legend()
    st.pyplot(fig)
```



[h]

SOLUCION ALTERNATIVA....

```python
import numpy as np
import matplotlib.pyplot as plt
import streamlit as st

ancho_banda_total = 1000
max_archivos = 50
archivos_extras = 30
st.title('Sistema de Transmisión de Datos')
st.write('')
n_archivos = st.slider('Número de Archivos a Transmitir', 1, max_archivos, 1)
ancho_banda_por_archivo = st.slider('Ancho de Banda por Archivo (Mbps)', 1, 100, 10)

if n_archivos <= archivos_extras:
    ancho_banda_usado = n_archivos * ancho_banda_por_archivo
    ancho_banda_disponible = ancho_banda_total - ancho_banda_usado
    d = ancho_banda_total
else:
    ancho_banda_reducido = ancho_banda_total * (1 - 0.05 * (n_archivos - archivos_extras))
    ancho_banda_usado = n_archivos * ancho_banda_por_archivo
    ancho_banda_disponible = ancho_banda_reducido - ancho_banda_usado
    d = ancho_banda_reducido

st.write(f'Ancho de banda utilizado: {ancho_banda_usado:.2f} Mbps')
st.write(f'Ancho de banda disponible: {ancho_banda_disponible:.2f} Mbps')
st.write(f'Ancho de banda TOTAL: {d:.2f} Mbps')

if ancho_banda_usado > d:
    st.write(f'⚠ **Exceso de ancho de banda!** El sistema está utilizando más de {d} Mbps.')
x_values = np.arange(1, max_archivos + 1)
ancho_banda_disponible_list = []
for n in x_values:
    if n <= archivos_extras:
        ancho_banda_disponible_list.append(ancho_banda_total)
    else:
        ancho_banda_disponible_list.append(ancho_banda_total * (1 - 0.05 * (n - archivos_extras)))

fig, ax = plt.subplots(figsize=(10, 6))
ax.plot(x_values, ancho_banda_disponible_list, label='Ancho de Banda Disponible (Mbps)', color='b')
ax.axhline(y=ancho_banda_total, color='r', linestyle='--', label='Ancho de Banda Total (1000 Mbps)')
ax.axvline(x=30, color='red', linestyle='--', label='Límite sin penalización')
ax.axvline(x=n_archivos, color='green', linestyle='--', label=f'Archivos Seleccionados: {n_archivos}')
ax.set_title('Relación entre Número de Archivos y Ancho de Banda Disponible')
ax.set_xlabel('Número de Archivos Transmitidos')
ax.set_ylabel('Ancho de Banda Disponible (Mbps)')
ax.grid(True)
ax.legend()
```

[h]

[h]

Problema 7

Un sistema de colas procesa x trabajos por segundo. La función del tiempo de respuesta T(x) = $\frac{100}{x} + 2x$. Minimiza el tiempo de respuesta del sistema, considerando que el sistema debe procesar al menos 5 trabajos por segundo.

Restricciones

El sistema debe procesar al menos 5 trabajos por segundo:

$$x \geq 5$$

Función

Minimizar el tiempo de respuesta $T(x)$:

$$T(x) = \frac{100}{x} + 2x$$

Solución

Para encontrar el valor de x que minimiza $T(x)$, tomamos la derivada de $T(x)$ respecto a x:

$$T'(x) = -\frac{100}{x^2} + 2$$

Igualamos la derivada a cero para encontrar los puntos críticos:

$$-\frac{100}{x^2} + 2 = 0$$

$$\frac{100}{x^2} = 2$$

$$x^2 = \frac{100}{2} = 50$$

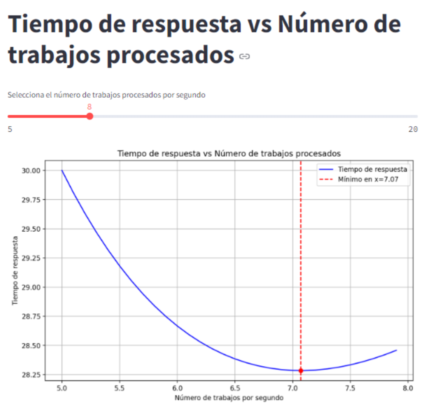
$$x = \sqrt{50} = 5\sqrt{2} = 7.07$$

El valor que minimiza el tiempo de respuesta es $x = 7.07$. Verificamos que este valor cumple con la restricción $x \geq 5$.

Rpta

El valor óptimo de x que minimiza el tiempo de respuesta del sistema es $x = 7.07$ trabajos por segundo.

[h]

[h]

Problema 8

El entrenamiento de un modelo de deep learning en una GPU consume x unidades de energía por lote. El objetivo es maximizar el tamaño del lote x, pero el consumo de energía total no puede exceder las 200 unidades por segundo, y cada lote adicional más allá del 10 reduce el rendimiento en un 10 %.

Restricciones

1. Consumo de energía total:

$$x \cdot E(x) \leq 200$$

2. Reducción de rendimiento: Si $x > 10$,

Entonces..

El consumo de energía por lote $E(x)$ se modela como sigue:

$$E(x) = \begin{cases} x & \text{si } x \leq 10 \\ x \cdot (1 + 0.1 \cdot (x - 10)) & \text{si } x > 10 \end{cases}$$

Función

$$x \cdot E(x) \leq 200$$

Solución

Caso 1: $x \leq 10$

$$x^2 \leq 200 \quad \Rightarrow \quad x \leq \sqrt{200} \approx 14.14$$

Caso 2: $x > 10$

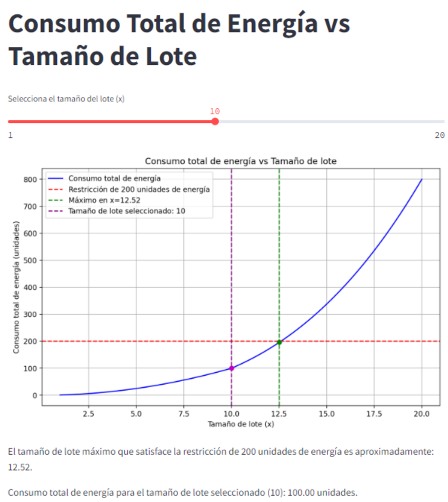
$$x \cdot x \cdot (1 + 0.1(x - 10)) \leq 200$$

Rpta

El tamaño óptimo de lote x depende de si $n \leq 10$, o $n > 10$. En el primer caso, se maximiza el tamaño del lote manteniendo el rendimiento completo; en el segundo caso, el rendimiento disminuye y la energía debe ajustarse para no exceder el límite.

[h] 

```python
problema8.py > ...
import numpy as np
import matplotlib.pyplot as plt
import streamlit as st

st.title('Consumo Total de Energía vs Tamaño de Lote')
st.write('')
tamaño_lote = st.slider('Selecciona el tamaño del lote (x)', min_value=1, max_value=20, value=10)

def energia_consumida(x):
    if x <= 10:
        return x
    else:
        return x * (1 + 0.1 * (x - 10))

def consumo_total(x):
    return x * energia_consumida(x)

x_values = np.linspace(1, 20, 100)
consumo = np.array([consumo_total(x) for x in x_values])
x_max = np.max(x_values[consumo <= 200])

fig, ax = plt.subplots(figsize=(10, 6))
ax.plot(x_values, consumo, label='Consumo total de energía', color='b')
ax.axhline(y=200, color='r', linestyle='--', label='Restricción de 200 unidades de energía')
ax.axvline(x=x_max, color='g', linestyle='--', label=f'Máximo en x={x_max:.2f}')
ax.plot(x_max, consumo_total(x_max), 'go')
ax.axvline(x=tamaño_lote, color='purple', linestyle='--', label=f'Tamaño de lote seleccionado: {tamaño_lote}')
ax.plot(tamaño_lote, consumo_total(tamaño_lote), 'mo')
ax.set_title('Consumo total de energía vs Tamaño de lote')
ax.set_xlabel('Tamaño de lote (x)')
ax.set_ylabel('Consumo total de energía (unidades)')
ax.grid(True)
ax.legend()
st.pyplot(fig)
st.write(f"El tamaño de lote máximo que satisface la restricción de 200 unidades de energía es aproximadamente: {x_max:.2f}.")
st.write(f"Consumo total de energía para el tamaño de lote seleccionado ({tamaño_lote}): {consumo_total(tamaño_lote):.2f} unidades.")
```



[h]

Problema 9

Una empresa almacena datos en la nube. El costo de almacenamiento por TB es de $50 + 5x$ dólares, donde x es la cantidad de TB de almacenamiento utilizado. La empresa tiene un presupuesto de 500 dólares. Maximiza la cantidad de datos almacenados sin exceder el presupuesto.

Restricciones

1. Presupuesto total:

$$50 + 5x \leq 500$$

Función

Maximizar la cantidad de almacenamiento x sujeto a la restricción del presupuesto.

Solucion

El costo de almacenamiento por x TB se expresa como:

$$C(x) = 50 + 5x$$

La restricción se puede reescribir como:

$$50 + 5x \leq 500$$

Resolviendo esta ecuación:

$$5x \leq 500 - 50$$

$$5x \leq 450 \quad \Rightarrow \quad x \leq 90$$

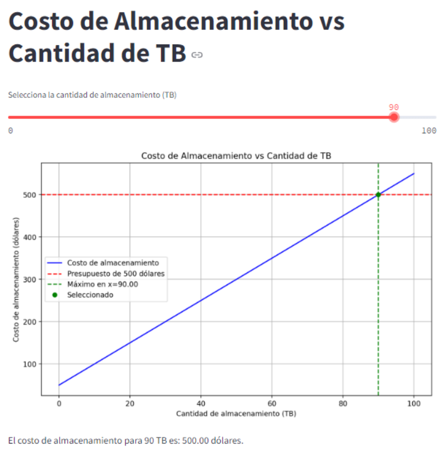
Rpta

La empresa puede almacenar un máximo de 90 TB sin exceder su presupuesto de 500 dólares.

[h] 

problema9.py > ...

```python
import numpy as np
import matplotlib.pyplot as plt
import streamlit as st

st.title('Costo de Almacenamiento vs Cantidad de TB')
st.write('')

def costo_almacenamiento(x):
    return 50 + 5 * x

presupuesto = 500
x_val = np.linspace(0, 100, 100)
costos = costo_almacenamiento(x_val)
x_max = (presupuesto - 50) / 5
cantidad_almacenamiento = st.slider(
    'Selecciona la cantidad de almacenamiento (TB)',
    min_value=0,
    max_value=100,
    value=0
)

costo_seleccionado = costo_almacenamiento(cantidad_almacenamiento)

fig, ax = plt.subplots(figsize=(10, 6))
ax.plot(x_val, costos, label='Costo de almacenamiento', color='b')
ax.axhline(y=presupuesto, color='r', linestyle='--', label='Presupuesto de 500 dólares')
ax.axvline(x=x_max, color='g', linestyle='--', label=f'Máximo en x={x_max:.2f}')
ax.plot(cantidad_almacenamiento, costo_seleccionado, 'go', label='Seleccionado')
ax.set_title('Costo de Almacenamiento vs Cantidad de TB')
ax.set_xlabel('Cantidad de almacenamiento (TB)')
ax.set_ylabel('Costo de almacenamiento (dólares)')
ax.grid(True)
ax.legend()
st.pyplot(fig)
st.write(f"El costo de almacenamiento para {cantidad_almacenamiento} TB es: {costo_seleccionado:.2f} dólares.")
```



[h]

Problema 10

Un sistema de mensajería tiene una latencia L(x) $= 100 - 2x$, donde x es el número de mensajes por segundo. La latencia no puede ser inferior a 20 ms debido a restricciones del protocolo. Maximiza el número de mensajes enviados sin que la latencia caiga por debajo de este límite

Restricciones

1. Latencia mínima:

$$L(x) \geq 20$$

Función

Maximizar la cantidad de mensajes x sujeto a la restricción de latencia.

Solucion

La latencia se expresa como:

$$L(x) = 100 - 2x$$

La restricción de latencia mínima se puede reescribir como:

$$100 - 2x \geq 20$$

Resolviendo esta desigualdad:

$$-2x \geq 20 - 100$$

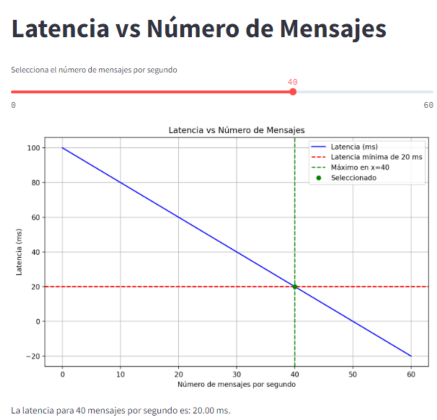
$$-2x \geq -80 \quad \Rightarrow \quad x \leq 40$$

Rpta

El número máximo de mensajes que se pueden enviar por segundo, sin que la latencia caiga por debajo de 20 ms, es 40.

```python
import numpy as np
import matplotlib.pyplot as plt
import streamlit as st

st.title('Latencia vs Número de Mensajes')
st.write('')

def latencia(x):
    return 100 - 2 * x

x_val = np.linspace(0, 60, 100)
latencias = latencia(x_val)
x_max = 40

numero_mensajes = st.slider(
    'Selecciona el número de mensajes por segundo',
    min_value=0,
    max_value=60,
    value=0
)

latencia_seleccionada = latencia(numero_mensajes)

fig, ax = plt.subplots(figsize=(10, 6))
ax.plot(x_val, latencias, label='Latencia (ms)', color='b')
ax.axhline(y=20, color='r', linestyle='--', label='Latencia mínima de 20 ms')
ax.axvline(x=x_max, color='g', linestyle='--', label=f'Máximo en x={x_max}')
ax.plot(numero_mensajes, latencia_seleccionada, 'go', label='Seleccionado')
ax.set_title('Latencia vs Número de Mensajes')
ax.set_xlabel('Número de mensajes por segundo')
ax.set_ylabel('Latencia (ms)')
ax.grid(True)
ax.legend()
st.pyplot(fig)
st.write(f"La latencia para {numero_mensajes} mensajes por segundo es: {latencia_seleccionada:.2f} ms.")
```

[h]



[h]