Pontificia Universidad Católica de Chile
Departamento de Ingenieria Industrial y de Sistemas
ICS1113- Optimización

Tarea 3

Integrantes: Jacinta Ortiz y Vicente Lavagnino

Índice

1. Pregunta 1:

1). Item 1

Problema de Maximización

Este ejercicio se trata de un caso de maximización debido a que si nos fijamos en las branches del modelo, vemos que a medida que avanzamos en la iteraciones, llegamos a valores de v^* cada vez más pequeños, es decir peores resultados para el problema.

Tabla

Subproblema	Incumbente	Mejor Cota	Gap
P0	-∞	170	$\emptyset$
P1	-∞	170	$\emptyset$
P2	-∞	165	$\emptyset$
P3	-∞	165	$\emptyset$
P4	152	165	0.0855
P5	152	165	0.0855
P6	152	160	0.0526
P7	152	160	0.0526
P8	152	160	0.0526
P9	152	160	0.0526
P10	157	159	0.0127
P11	158	159	0.0063
P12	158	159	0.0063
P13	158.5	159	0.0031
P14	158.5	159	0.0031

Nodos

Si el algoritmo se hubiera ejecutado de manera correcta, los siguientes nodos no se debieron resolver: P7, P8, P12, P14.Esto se debe a que con el valor

Estos nodos podrían haberse podado porque sus valores eran mayores que el valor incumbente de 150 encontrado en P5.

Gap <2.5 %

El nodo P10 es el primer nodo en el que se podría detener el algoritmo y obtener un GAP menor al 2.5 %.

2). Item 2

3). Item 3

4). Item 4

Del problema de maximización, se pueden deducir los siquientes covers distintos:

- C= {1,2,3}
- C= {1,2,4}

- C= {2,3,6}
- C= {4,5}

El nuevo problema, con las 4 restricciones añadidas sería el siguiente:

$$\mathrm{x}_1, x_2, x_3, x_4, x_5, x_6 \in \{0, 1\}$$

2. Pregunta 2:

1). Item 1

Verdadera

No siempre ocurre que para un Q idéntico, este tenga la misma solución óptima, sin embargo existe el caso. Para demostrar la veracidad de la afirmación entonces demostraremos con un ejemplo.

Sea Q idéntico a P tal que

$$\min_{x \in \mathbb{R}^2} \quad f(x) = (x_1 - 1)^2 + (x_2 - 1)^2 \tag{1}$$

Para el caso de P cons sus retricciones

$$x_1 - 1 \leq 0 \tag{2}$$

$$x_2 - 1 \leq 0 \tag{3}$$

La solución óptima para P y Q también es $x^* = (1, 1)$, ya que es el mínimo global de $f(x)$ sin ninguna restricción.

Verdadera

El caso base de esta afirmación la podemos ver con cualquier función lineal, ya que estas funciones son cóncavas y convexas al mismo tiempo.

Esto se debe a que se cumplen ambos principios, de modo tal que para $f(x) = ax = b$ con a y b constantes, se cumple que

$$f(\lambda x_1 + (1 - \lambda)x_2) = \lambda f(x_1) + (1 - \lambda) f(x_2) \tag{4}$$

Falso

Esta afirmación es falsa ya que no necesariamente es así, para esto buscaremos un contraejemplo.

$$f(x) = \sin(x) \tag{5}$$

$$f'(x) = \cos(x) \tag{6}$$

Evaluando, vemos que la función $\sin(x)$ tiene infinitos mínimos locales y máximos locales y a la vez admite solución óptima (más de una).

Este contraejemplo muestra que no siempre hay más mínimos locales que máximos locales en un problema de minimización no lineal irrestricto. Por lo tanto, la afirmación es falsa.

Falsa

Es falso ya que, por ejemplo, para el caso de μ_j, en un caso de restricción de igualdad, no tiene condicionalidad en el signo. Es decir, puede tomar valores incluyendo el 0.

Falsa

Verdadero

Esto se debe a que cualquier punto singular pertenece a el conjunto de soluciones factibles del modelo, esto significa que debe satisfacer las restricciones del mismo. Es decir, se puede entender como la unión del espacio muestral de las restricciones del modelo, por lo tanto se puede definir como una expresión de las mismas n restricciónes.

Esto permite que un punto puede satisfacer a más de una restricción, haciendolas linealmente dependtientes.

Falso

Verdadero

2). Item 2

Suponiendo que si $\nabla g_i(\bar{x})^T d \leq 0$ para todo $i \in I$, entonces d es factible el sentido de las restricciones activas.

Sin embargo al hacer esto, vemos un problema ya que $\nabla f(\bar{x})^T d < 0$, esto significa que existe un problemacon las condiciones de optimalidad KKT, ya que$f$ se reduciría sin violar las restricciones activas.

Por lo tanto, podemos demostrar que para que $\nabla f(\bar{x})^T d < 0$ debe existir al menos un $i \in I$ tal que $\nabla g_i(\bar{x})^T d > 0$.

3). Item 3

Gráfico y Solución

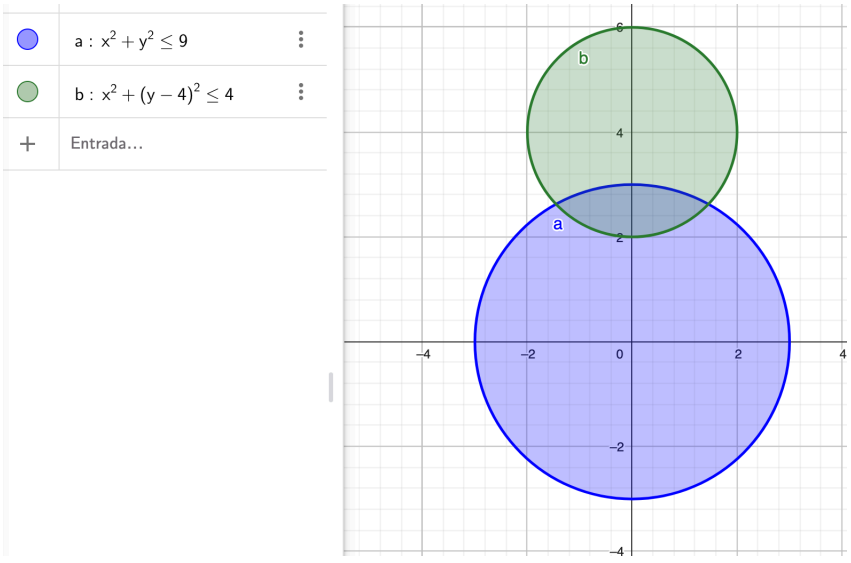

Gráfico de las restricciones en 2D

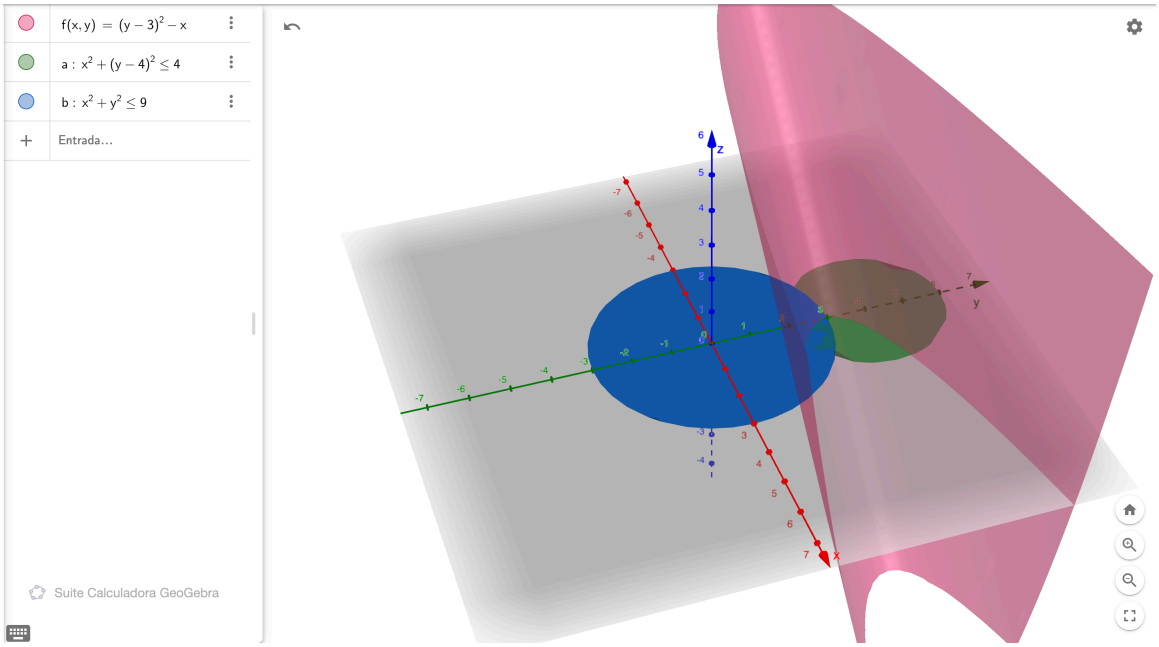

Gráfico 3D de las restricciones y si intersección con la función a optimizar

Mediante este gráfico, podemos ver el punto de intersección de las restricciones, donde $x \approx -1{,}5$ y $y \approx 3$, resolviendo este modelo en detalle podemos ver que los voles definitivos son:

$$x_1 = -1{,}45237 \; x_2 = 2{,}625$$
$$\text{máx}(f(x_1, x_2)) = 1{,}59299 \; [1]$$

Condiciones de KKT

Para verificar las condiciones, primero escribimos la función Lagrangeana:

$$L(x_1, x_2, \mu_1, \mu_2) = -(x_2 - 3)^2 + x_1 + \mu_1(x_1^2 + x_2^2 - 9) + \mu_2(x_1^2 + (x_2 - 4)^2 - 4)$$

Luego resolvemos las derivadas parciales igualadas a cero:

$$\frac{\partial L}{\partial x_1} = 1 + 2\mu_1 x_1 + 2\mu_2 x_1 = 0$$
$$\frac{\partial L}{\partial x_2} = -2(x_2 - 3) + 2\mu_1 x_2 + 2\mu_2(x_2 - 4) = 0$$
$$x_1 = -\frac{1}{2(\mu_1 + \mu_2)}$$
$$x_2 = \frac{3 - 4\mu_2}{1 - \mu_1 - \mu_2}$$

Después, como $\mu_1, \mu_2 \geq 0$, evualamos con $(x_1 = -1,45, x_2 = 2,62)$:

$$-1,45 = -\frac{1}{2(\mu_1 + \mu_2)}$$

$2,62 = \frac{3-4\mu_2}{1-\mu_1-\mu_2}$
Resolviendo el sistema se obtiene $\mu_1 = 0,02$ y $\mu_2 = 0,32$ ambos valores mayores a cero.

[1] se adjunta desarrollo en anexo

3. Pregunta 3:

1). main1.py

```
# TAREA 3
# Jacinta Ortiz y Vicente Lavagnino

# Pregunta 3.1

## _____ CSV _____ ##
import pandas as pd
from gurobipy import GRB, Model, quicksum

# PARAMETROS

peso = []
estatura = []
edad = []
contorno_cabeza = []
distancia_hombro_cuello = []
altura_cuello = []
contorno_cuello = []
altura_cabeza = []
peso_cabeza = []
distancia_hombro_cabeza = []
contorno_caja_toraxica = []
contorno_brazo  = []
largo_del_brazo = []
distancia_menton_oreja = []
ancho_cabeza = []
gramos_relleno = []

# MUESTRA CSV
muestra = pd.read_csv("muestras.csv", header=0).values.tolist()

for i in range(len(muestra)):
    peso.append(muestra[i][0])
    estatura.append(muestra[i][1])
    edad.append(muestra[i][2])
    contorno_cabeza.append(muestra[i][3])
    distancia_hombro_cuello.append(muestra[i][4])
    altura_cuello.append(muestra[i][5])
    contorno_cuello.append(muestra[i][6])
    altura_cabeza.append(muestra[i][7])
    peso_cabeza.append(muestra[i][8])
    distancia_hombro_cabeza.append(muestra[i][9])
    contorno_caja_toraxica.append(muestra[i][10])
    contorno_brazo.append(muestra[i][11])
    largo_del_brazo.append(muestra[i][12])
    distancia_menton_oreja.append(muestra[i][13])
    ancho_cabeza.append(muestra[i][14])
    gramos_relleno.append(muestra[i][15])

encuestados = len(peso)
```

```python
## _____ GUROBI _____ ##

# Generacion del modelo
model = Model("Ejercicio 3.1")
model.setParam("TimeLimit", 60)  # Tiempo maximo en segundos

# Se instancian variables de decision
b = model.addVar(vtype=GRB.CONTINUOUS, name="b")
w_i = model.addVars(15, vtype=GRB.CONTINUOUS, name="w_i")

#------------------ Agregar las variables al modelo -------------------
model.update()

#------------------------- Funcion Objetivo ---------------------------
objetivo = quicksum((b + quicksum(w_i[j] * muestra[i][j] for j in range(15)) - gramos_relleno[i])
↪  ** 2 for i in range(encuestados))
model.setObjective(objetivo, GRB.MINIMIZE)

# Optimizar el modelo
model.optimize()

if model.status == GRB.OPTIMAL:
    b_opt = b.X
    w_opt = [w_i[i].X for i in range(15)]

    columnas = [
        "peso", "estatura", "edad", "contorno_cabeza", "distancia_hombro_cuello",
        "altura_cuello", "contorno_cuello", "altura_cabeza", "peso_cabeza",
        "distancia_hombro_cabeza", "contorno_caja_toraxica", "contorno_brazo",
        "largo_del_brazo", "distancia_menton_oreja", "ancho_cabeza"
    ]

    for j in range(15):
        print(f'{columnas[j]}: {w_opt[j]}')

    print(f'b: {b_opt}')
    print(f'Función objetivo: {model.objVal}')
else:
    print("No se encontró una solución óptima.")
```

Resultado

```
Barrier solved model in 14 iterations and 0.00 seconds (0.00 work units)
Optimal objective 3.96190545e+04

peso: 0.1280667786330526
estatura: 0.591931249386685
edad: 3.8035343577380325e-11
contorno_cabeza: 2.467538032512749e-10
distancia_hombro_cuello: 0.01041173367754975
altura_cuello: 0.5043019625965052
contorno_cuello: 0.2551237854177278
altura_cabeza: 4.0386171881273185e-10
peso_cabeza: 2.4725921528278244e-09
distancia_hombro_cabeza: 1.4799481779248378e-10
contorno_caja_toraxica: 0.04640411765537249
contorno_brazo: 1.041650012178981e-10
largo_del_brazo: 0.157046900424657
distancia_menton_oreja: 0.7025815960112788
ancho_cabeza: 2.8799868122953445e-10
b: 387.50675285109253
Función objetivo: 39619.054500412196
vicentelavagnino@Vicentes-MacBook-Pro ~/L/C/O/S/2/I/T/T/code>
```

2). main2.py

NOTA: Al ejecutar el código, se solicitará ingresar manualmente el valor de λ en la consola.

```
# TAREA 3
# Jacinta Ortiz y Vicente Lavagnino

# Pregunta 3.2

## _____ CSV _____ ##
import pandas as pd
from gurobipy import GRB, Model, quicksum

# PARAMETROS

peso = []
estatura = []
edad = []
contorno_cabeza = []
distancia_hombro_cuello = []
altura_cuello = []
contorno_cuello = []
altura_cabeza = []
peso_cabeza = []
distancia_hombro_cabeza = []
contorno_caja_toraxica = []
contorno_brazo  = []
largo_del_brazo = []
distancia_menton_oreja = []
ancho_cabeza = []
gramos_relleno = []

# MUESTRA CSV
```

```
muestra = pd.read_csv("muestras.csv", header=0).values.tolist()

for i in range(len(muestra)):
    peso.append(muestra[i][0])
    estatura.append(muestra[i][1])
    edad.append(muestra[i][2])
    contorno_cabeza.append(muestra[i][3])
    distancia_hombro_cuello.append(muestra[i][4])
    altura_cuello.append(muestra[i][5])
    contorno_cuello.append(muestra[i][6])
    altura_cabeza.append(muestra[i][7])
    peso_cabeza.append(muestra[i][8])
    distancia_hombro_cabeza.append(muestra[i][9])
    contorno_caja_toraxica.append(muestra[i][10])
    contorno_brazo.append(muestra[i][11])
    largo_del_brazo.append(muestra[i][12])
    distancia_menton_oreja.append(muestra[i][13])
    ancho_cabeza.append(muestra[i][14])
    gramos_relleno.append(muestra[i][15])

encuestados = len(peso)

## _____________________ GUROBI _____________________ ##

# Generacion del modelo
model = Model("Ejercicio 3.2")
model.setParam("TimeLimit", 60)  # Tiempo maximo en segundos

  = int(input("Ingrese el valor de  : "))

# Se instancian variables de decision
b = model.addVar(vtype=GRB.CONTINUOUS, name="b")
w_i = model.addVars(15, vtype=GRB.CONTINUOUS, name="w_i")
z_i = model.addVars(15, vtype=GRB.CONTINUOUS, name="z_i")

#------------------ Agregar las variables al modelo -------------------
model.update()

#------------------------- Funcion Objetivo ---------------------------
objetivo = quicksum((b + quicksum(w_i[j] * muestra[i][j] for j in range(15)) - gramos_relleno[i])
↪  ** 2 for i in range(encuestados)) +   * quicksum(z_i[j] for j in range(15))
model.setObjective(objetivo, GRB.MINIMIZE)


#------------------------ Restricciones ---------------------------
for j in range(15):
    model.addConstr(-z_i[j] <= w_i[j], name=f"R1")
    model.addConstr(w_i[j] <= z_i[j], name=f"R2")

# Optimizar el modelo
model.optimize()

if model.status == GRB.OPTIMAL:
    b_sol = b.X
    w_sol = [w_i[i].X for i in range(15)]
    z_sol = [z_i[j].X for j in range(15)]
```

```
    columnas = [
        "peso", "estatura", "edad", "contorno_cabeza", "distancia_hombro_cuello",
        "altura_cuello", "contorno_cuello", "altura_cabeza", "peso_cabeza",
        "distancia_hombro_cabeza", "contorno_caja_toraxica", "contorno_brazo",
        "largo_del_brazo", "distancia_menton_oreja", "ancho_cabeza"
    ]

    for j in range(15):
        print(f'{columnas[j]}: {w_sol[j]}')
        # print(f'Valor Absoluto: {z_sol[j]}')

    print(f'b: {b_sol}')
    print(f"Valor  : { }")

    print("\n")
    print("Los 5 ponderadores más importantes son:")

    mejores = []

    for j in range(15):
        mejores.append((columnas[j], w_sol[j]))

    mejores.sort(key=lambda x: x[1], reverse=False)

    for i in range(5):
        print(f"{mejores[i][0]}: {mejores[i][1]}")

    print(f'Función objetivo: {model.objVal}')

else:
    print("No se encontró una solución óptima.")
```

Resultado

```
Barrier solved model in 15 iterations and 0.00 seconds (0.00 work units)
Optimal objective 3.96214481e+04

peso: 0.12836786879425946
estatura: 0.5922254294221589
edad: 1.1646215700619181e-13
contorno_cabeza: 7.541225003173428e-13
distancia_hombro_cuello: 0.008567955517354646
altura_cuello: 0.5024504642254617
contorno_cuello: 0.25516520965286327
altura_cabeza: 1.234267416492333e-12
peso_cabeza: 7.466390139463014e-12
distancia_hombro_cabeza: 4.5253626520319754e-13
contorno_caja_toraxica: 0.046504870083155764
contorno_brazo: 3.1890012964369836e-13
largo_del_brazo: 0.15714717238918646
distancia_menton_oreja: 0.7008966171351756
ancho_cabeza: 8.806127828958442e-13
b: 387.47686586913454
Valor λ: 1


Los 5 ponderadores más importantes son:
edad: 1.1646215700619181e-13
contorno_brazo: 3.1890012964369836e-13
distancia_hombro_cabeza: 4.5253626520319754e-13
contorno_cabeza: 7.541225003173428e-13
ancho_cabeza: 8.806127828958442e-13
Función objetivo: 39621.448097892106
vicentelavagnino@Vicentes-MacBook-Pro ~/L/C/O/S/2/I/T/T/code>
```

Respecto a los datos más importante y el valor de λ, tras varias iteraciones se logra ver que $\lambda = 1000000$ (ese orden de magnitud), logra hacer 0 a alguno de los ponderadores, además se nota que la relación de cual pondera más siempre se mantiene en el mismo orden de cual pondera más y cual pondera menos, independiente del valor λ, es por esto que el resultado de esta consola se usa para un caso génerico de $\lambda = 1$.

3). main3.py

```python
# TAREA 3
# Jacinta Ortiz y Vicente Lavagnino

# Pregunta 3.3

## _____ CSV _____ ##
import pandas as pd
from gurobipy import GRB, Model, quicksum

# PARAMETROS

peso = []
estatura = []
edad = []
contorno_cabeza = []
distancia_hombro_cuello = []
altura_cuello = []
```

```
contorno_cuello = []
altura_cabeza = []
peso_cabeza = []
distancia_hombro_cabeza = []
contorno_caja_toraxica = []
contorno_brazo  = []
largo_del_brazo = []
distancia_menton_oreja = []
ancho_cabeza = []
gramos_relleno = []

# MUESTRA CSV
muestra = pd.read_csv("muestras.csv", header=0).values.tolist()

for i in range(len(muestra)):
    peso.append(muestra[i][0])
    estatura.append(muestra[i][1])
    edad.append(muestra[i][2])
    contorno_cabeza.append(muestra[i][3])
    distancia_hombro_cuello.append(muestra[i][4])
    altura_cuello.append(muestra[i][5])
    contorno_cuello.append(muestra[i][6])
    altura_cabeza.append(muestra[i][7])
    peso_cabeza.append(muestra[i][8])
    distancia_hombro_cabeza.append(muestra[i][9])
    contorno_caja_toraxica.append(muestra[i][10])
    contorno_brazo.append(muestra[i][11])
    largo_del_brazo.append(muestra[i][12])
    distancia_menton_oreja.append(muestra[i][13])
    ancho_cabeza.append(muestra[i][14])
    gramos_relleno.append(muestra[i][15])

encuestados = len(peso)


edad = []
contorno_brazo = []
distancia_hombro_cabeza = []
contorno_cabeza = []
ancho_cabeza = []
gramos_relleno = []

for i in range(len(muestra)):
    edad.append(muestra[i][2])
    contorno_cabeza.append(muestra[i][3])
    distancia_hombro_cabeza.append(muestra[i][9])
    contorno_brazo.append(muestra[i][11])
    ancho_cabeza.append(muestra[i][14])
    gramos_relleno.append(muestra[i][15])


nueva_muestra = []

for i in range(encuestados):
    nueva_muestra.append([edad[i], contorno_cabeza[i], distancia_hombro_cabeza[i],
    ↪  contorno_brazo[i], ancho_cabeza[i]])
```

```python
## _____ GUROBI _____ ##

# Generacion del modelo
model = Model("Ejercicio 3.3")
model.setParam("TimeLimit", 60)  # Tiempo maximo en segundos

# Se instancian variables de decision
b = model.addVar(vtype=GRB.CONTINUOUS, name="b")
w_i = model.addVars(15, vtype=GRB.CONTINUOUS, name="w_i")

#------------------ Agregar las variables al modelo -------------------
model.update()

#------------------------- Funcion Objetivo ---------------------------
objetivo = quicksum((b + quicksum(w_i[j] * nueva_muestra[i][j] for j in range(5)) -
↪  gramos_relleno[i]) ** 2 for i in range(encuestados))
model.setObjective(objetivo, GRB.MINIMIZE)

# Optimizar el modelo
model.optimize()

if model.status == GRB.OPTIMAL:
    b_opt = b.X
    w_opt = [w_i[i].X for i in range(5)]

    columnas = ["edad", "contorno_cabeza", "distancia_hombro_cabeza", "contorno_brazo",
    ↪  "ancho_cabeza"]

    for j in range(5):
        print(f'{columnas[j]}: {w_opt[j]}')

    print(f'b: {b_opt}')
    print(f'Función objetivo: {model.objVal}')
else:
    print("No se encontró una solución óptima.")
```

Resultado

```
Barrier solved model in 11 iterations and 0.00 seconds (0.00 work units)
Optimal objective 4.50349943e+04

edad: 0.15809976750476065
contorno_cabeza: 0.16450630597772667
distancia_hombro_cabeza: 0.31582725028290975
contorno_brazo: 0.4659704933667384
ancho_cabeza: 1.3598037081616718
b: 464.19606795481945
Función objetivo: 45034.99426826835
vicentelavagnino@Vicentes-MacBook-Pro ~/L/C/O/S/2/I/T/T/code>
```

4. Anexo

1). Complementos 2.3

Sea agrega cálculo de solución de pregunta 2.3, la cual se utilizó gurobi para calcular el valor más preciso.

```
from gurobipy import Model, GRB

m = Model()


x1 = m.addVar(vtype=GRB.CONTINUOUS, name="x1", lb=-GRB.INFINITY)
x2 = m.addVar(vtype=GRB.CONTINUOUS, name="x2", lb=-GRB.INFINITY)


objective = (x2 - 3)*(x2 - 3) - x1

m.setObjective(objective, GRB.MAXIMIZE)

m.addQConstr(x1*x1 + x2*x2 <= 9, "c1")
m.addQConstr(x1*x1 + (x2 - 4)*(x2 - 4) <= 4, "c2")

m.optimize()

if m.status == GRB.OPTIMAL:
    print(f"Valor Objetivo: {m.objVal}")
    print(f"x1: {x1.X}")
    print(f"x2: {x2.X}")
```