Tarea 10 Optimización

Roberto Vásquez Martínez
Profesor: Joaquín Peña Acevedo

17/Mayo/2022

1 Ejercicio 1 (4 puntos)

1. Escriba la descripción del tema para el proyecto final del curso.
2. La descripción no tiene que ser detallada. Sólo debe quedar claro cual el problema que quiere resolver, si ya cuentan con la información para resolver el problema (datos, referencia bibliográfica, etc.)
3. Mencione el tipo de pruebas que va a realizar y la manera en que va a validar los resultados.
4. En la semana de 16 de mayo recibirá un mensaje que indica si el tema fue aceptado o necesita precisar algo o cambiarlo.

1.1 Descripción del proyecto

A continuación, describiré los componentes principales de mi propuesta de Proyecto Final para el curso de Optimización.

Título: Optimización en el caso no diferenciable: Subdiferenciales.

El propósito del proyecto será dar una introducción al caso de optimización cuando la función objetivo f es continua pero no diferenciable. El caso en el cual $f \in C^1$ o $f \in C^2$ se discutió en el curso con los métodos Newton/Quasi-Newton y de búsqueda en línea.

En primer lugar, generalizaremos el concepto de gradiente al de *subgradiente* o subdiferencial en el caso cuando f es una función convexa. Se pretende desarrollar un poco de teoría de esta generalización y probar la condición de optimalidad en términos de subdiferenciales.

Para ilustrar el problema de optimización en el caso no diferenciable proponemos resolver el problema de regresión LASSO. Este problema se puede resolver utilizando *descenso por coordenadas* y se pueden verificar las condiciones de optimalidad usando el subdiferencial de la función objetivo LASSO. Además, presentaremos el caso particular con un solo predictor para determinar la solución LASSO a través de subdiferenciales.

Para el caso multipredictores utilizaremos el conjunto de datos *crime data: crime rate and five predictors* tomado del libro *Statistical Learning with Sparsity: The LASSO and Generalizations*.

La teoría sobre cálculo subdiferencial será tomada del libro *Convex Analysis and Minimization Algorithms I: Fundamentals*.

2 Ejercicio 2 (3 puntos)

Considere el ejemplo visto en clase:

$$\begin{array}{ll} \max & x_1 + x_2 \\ & 50x_1 + 24x_2 \leq 2400 \\ & 30x_1 + 33x_2 \leq 2100 \\ & x_1 \geq 45 \\ & x_2 \geq 5 \end{array}$$

Vimos que se puede escribir en forma estándar como:

$$\begin{array}{ll} \min & -x_1 - x_2 \\ & 50x_1 + 24x_2 + x_3 = 2400 \\ & 30x_1 + 33x_2 + x_4 = 2100 \\ & x_1 - x_5 = 45 \\ & x_2 - x_6 = 5 \\ & x_1, x_2, x_3, x_4, x_5, x_6 \geq 0 \end{array}$$

Puede usar el código del Notebook **ejemploPuntosBasicosFactibles.ipynb** para obtener la solución $\mathbf{x}_*$ del problema en forma estándar.

Las condiciones KKT son:

$$\begin{aligned} \mathbf{A}^\top \lambda + \mathbf{s} &= \mathbf{c}, && (1) \\ \mathbf{A}\mathbf{x} &= \mathbf{b}, && (2) \\ \mathbf{x} &\geq \mathbf{0}, && (3) \\ \mathbf{s} &\geq \mathbf{0}, && (4) \\ x_i s_i &= 0, \quad i = 1, 2, ..., n. && (5) \end{aligned}$$

Debe ser claro que por la manera en que se calculó $\mathbf{x}_*$ en el ejemplo de la clase, se cumplen las condiciones (2) y (3).

1. Dado $\mathbf{x}_*$ y por la condición de complementaridad (5), sabemos cuáles son las componentes de $\mathbf{s}$ que son cero y cuáles deben ser calculadas. Use eso y la condición (1) para calcular λ y las componentes de $\mathbf{s}$ desconocidas. Imprima los vectores λ y $\mathbf{s}$.
2. Verique que se cumplen la condiciones (4) y (5), y con esto se comprueba que $\mathbf{x}_*$ es solución del problema estándar.
3. Calcule el valor

$$\mathbf{b}^\top \lambda$$

y compare este valor con el valor de la función objetivo $\mathbf{c}^\top \mathbf{x}_*$.

2.0.1 Solución:

[14]:
```
# Pongo el codigo para obtener la solución x que calculamos en clase

import numpy as np
from itertools import combinations

c = np.array([-1, -1, 0, 0, 0, 0 ])
b = np.array([2400, 2100, 45, 5])
A = np.array([[50, 24, 1, 0, 0, 0],
              [30, 33, 0, 1, 0, 0],
              [ 1,  0, 0, 0,-1, 0],
              [ 0,  1, 0, 0, 0,-1] ])

m,n = A.shape
comb = list(combinations(list(range(n)), m))
print('Número de combinaciones:', len(comb))

dmin = None
for icols in comb:
    # Indices de las columnas seleccionadas
    jj = list(icols)
    # Matriz básica
    B     = A[:, jj]
    condB = np.linalg.cond(B)
    if condB>1.0e14:
        print('Es casi singular la matriz B con columnas', jj)
    else:
        # Solucion del sistema B*x=b
        xb = np.linalg.solve(B, b)
        # Solucion del problema en forma estándar
        x     = np.zeros(n)
        x[jj] = xb
        # Evaluación de la función objetivo
        f = np.vdot(c, x)
        # Se revisa si es vector x es factible. Claramente se cumple que A*x=b,
        # pero hay que verificar que x>=0.
        smsg  = 'No factible'
        bfact = False
        if sum(x>=0)==len(x):
            bfact = True
            smsg  = 'Factible'
        if bfact:
            # Si x es factible, almacenamos en xsol el punto x donde f es mínima
            if dmin==None:
                dmin = f
                xsol = x.copy()
            elif dmin>f:
```

```
                dmin = f
                xsol = x.copy()
        print("%6.1f %7.1f| % 8.2f % 8.2f % 8.2f % 8.2f % 8.2f % 8.2f| %s" %␣
 ↪(condB,
                          f,  x[0], x[1], x[2], x[3], x[4], x[5], smsg))

# Fijamos una tolerancia y hacemos cero las componentes de x que son menores que␣
 ↪la tolerancia
tol = (np.finfo(float).eps)**(3.0/4)
ii  = np.where(xsol<tol)[0]
xsol[ii] = 0.0

print('\nSolución del problema estándar:')
print('x*=', xsol)
print('Valor de la función objetivo en x*=', dmin)
```

```
Número de combinaciones: 15
4890.1   -50.0|    45.00     5.00    30.00   585.00     0.00     0.00| Factible
2175.8   -69.5|    64.50     5.00  -945.00     0.00    19.50     0.00| No
factible
1975.8   -67.7|    45.00    22.73  -395.45     0.00     0.00    17.73| No
factible
1305.9   -50.6|    45.60     5.00     0.00   567.00     0.60     0.00| Factible
2720.9   -51.2|    45.00     6.25     0.00   543.75     0.00     1.25| Factible
  70.1   -66.5|    30.97    35.48     0.00     0.00   -14.03    30.48| No
factible
Es casi singular la matriz B con columnas [0, 2, 3, 4]
3402.0   -45.0|    45.00     0.00   150.00   750.00     0.00    -5.00| No
factible
 113.4   -70.0|    70.00     0.00 -1100.00     0.00    25.00    -5.00| No
factible
  68.0   -48.0|    48.00     0.00     0.00   660.00     3.00    -5.00| No
factible
1667.0    -5.0|     0.00     5.00  2280.00  1935.00   -45.00     0.00| No
factible
Es casi singular la matriz B con columnas [1, 2, 3, 5]
  50.5   -63.6|     0.00    63.64   872.73     0.00   -45.00    58.64| No
factible
  69.4  -100.0|     0.00   100.00     0.00 -1200.00   -45.00    95.00| No
factible
   1.0     0.0|     0.00     0.00  2400.00  2100.00   -45.00    -5.00| No
factible

Solución del problema estándar:
x*= [ 45.     6.25   0.   543.75   0.     1.25]
Valor de la función objetivo en x*= -51.25
```

Para hallar $\lambda \in \mathbb{R}^m$ y $\mathbf{s} \in \lambda\mathbb{R}^n$ donde $\mathbf{A} = [A_1, \ldots, A_n] \in \mathbb{R}^{n\times m}$ debemos recurrir a las condiciones

KKT. Hemos visto en clase que estas son necesarias y suficientes para que $\mathbf{x}_*$ sea óptimo local.

Como se deben satisfacer las condiciones de (1) a (5) para la solución óptima $\mathbf{x}_*$ encontrada en la celda de código anterior, si $\mathbf{B}$ es la matriz básica asociada a la base $\mathcal{B}$ de la solución encontrada entonces

$$\mathbf{B}^T\lambda = \widetilde{\mathbf{c}},$$

donde $\widetilde{\mathbf{c}} = [c_i]_{i\in\mathcal{B}}$, pues por la condición de complementariedad $s_i = 0$ para $i \in \mathcal{B}$.

Por lo tanto, para calcular λ basta resolver el sistema de ecuaciones $\mathbf{B}^T\lambda = \widetilde{\mathbf{c}}$.

Por otro lado, para hallar $\mathbf{s}$, otra vez por la condición de complementariedad basta con encontrar $\widehat{\mathbf{s}} = [s_i]_{i\notin\mathcal{B}}$. Si $\mathbf{N} = [A_i]_{i\notin\mathcal{B}}$ entonces por (1) se debe cumplir que

$$\widehat{\mathbf{s}} = \widehat{\mathbf{c}} - \mathbf{N}^T\lambda,$$

donde $\widehat{\mathbf{c}} = [c_i]_{i\notin\mathcal{B}}$.

A continuación, calculamos λ y $\mathbf{s}$ con lo antes descrito.

[15]:
```
# Solucion del ejercicio

np.set_printoptions(precision=4)


jj=list(np.squeeze(np.where(xsol>0)))
B=A[:,jj]
tilde_c=c[jj]
# B.T*lamb=tilde_c
lamb=np.linalg.solve(B.T,tilde_c)
print('El vecto lambda es: ',lamb)
```

```
El vecto lambda es:  [-0.0417  0.      1.0833 -0.    ]
```

Como $\mathbf{s}$ corresponde a los multiplicadores de Lagrange de las reestricciones de desigualdad ($\mathbf{x} \geq 0$) entonces esperamos que $\mathbf{s} \geq 0$. Sabemos $s_i = 0$ para $i \in \mathcal{B}$, en la siguiente celda de código calculamos $\hat{\mathbf{s}}$ que corresponde a las entradas del vector $\mathbf{s}$ no básicas.

[16]:
```
s=np.zeros(A.shape[1])
jj_complement=list(np.squeeze(np.where(xsol==0)))
N=A[:,jj_complement]
hat_c=c[jj_complement]
# hat_s=c_hat-N.T*lamb
hat_s=hat_c-N.T@lamb
s[jj_complement]=hat_s
print('El valor del vector s es: ',s)
```

```
El valor del vector s es:  [0.     0.     0.0417 0.     1.0833 0.    ]
```

A partir de estos resultados es claro que se cumplen las condiciones KKT.

Finalmente, para el númeral 3 ejecutamos la siguiente celda de código.

[17]:
```
print('Valor de b.T lambda: ',np.vdot(b,lamb))
print('Valor de c.T x_ast: ',np.vdot(c,xsol))
```

```
Valor de b.T lambda:  -51.25000000000001
Valor de c.T x_ast:  -51.25
```

Vemos que el valor es el mismo, debido a que el valor $\mathbf{b}^T\lambda$ corresponde a la función objetivo del problema dual.

Por otro lado, probaremos la librería `pulp` para resolver el mismo de problema de programación lineal. El modelo descrito matemáticamente los describimos es el siguiente

[18]:
```
from pulp import LpMaximize, LpProblem, LpStatus, lpSum, LpVariable

# Creacción de una instancia de la  clase
model = LpProblem(name="small-problem", sense=LpMaximize)

# Variables de decisión
x1 = LpVariable(name="x1", lowBound=45)
x2 = LpVariable(name="x2", lowBound=5)

# Se agregan las restricciones del modelo
model += (50*x1 + 24*x2  <= 2400, "R1")
model += (30*x1 + 33*x2  <= 2100, "R2")

# Función objetivo
model += x1+x2

print(model)
```

```
small-problem:
MAXIMIZE
1*x1 + 1*x2 + 0
SUBJECT TO
R1: 50 x1 + 24 x2 <= 2400

R2: 30 x1 + 33 x2 <= 2100

VARIABLES
45 <= x1 Continuous
5 <= x2 Continuous
```

Y la solución es

[19]:
```
# Cálculo de la solución
from pulp import PULP_CBC_CMD

status = model.solve(PULP_CBC_CMD(msg=False))

print("Resultado: ", model.status, " | ", LpStatus[model.status])

print("Valor de la funciónn objetivo: " , model.objective.value())


print('Solución:')
for var in model.variables():
    print("%10s: %f"  % (var.name, var.value()) )

print('\nVariables de holgura:')
for name, constraint in model.constraints.items():
    print("%10s: %f" % (name, constraint.value()) )
```

```
Resultado:  1  |  Optimal
Valor de la funciónn objetivo:  51.25
Solución:
        x1: 45.000000
        x2: 6.250000

Variables de holgura:
        R1: 0.000000
        R2: -543.750000
```

Que coincide con el resultado obtenido con el código visto en clase

3 Ejercicio 3 (3 puntos)

Considere el problema

$$\begin{array}{rl} \min & x_1 + 2x_2 + x_3 + x_4 \\ \text{sujeto a} & 2x_1 + x_2 + 3x_3 + x_4 \leq 8 \\ & 2x_1 + 3x_2 + 4x_4 \leq 12 \\ & 3x_1 + x_2 + 2x_3 \leq 18 \\ & x_1, x_2, x_3, x_4 \geq 0 \end{array}$$

1. Escriba el problema en su forma estándar.
2. Construya los vectores $\mathbf{c}, \mathbf{b}$ y la matriz $\mathbf{A}$ del problema estándar y calcule la solución $\mathbf{x}_*$ del problema. Puede usar el código anterior.
3. Calcule los vectores λ y $\mathbf{s}$ y verique que se cumplen la condiciones (4) y (5), y con esto se comprueba que $\mathbf{x}_*$ es solución del problema estándar.
4. Calcule el valor

$$\mathbf{b}^\top \lambda$$

y compare este valor con el valor de la función objetivo $\mathbf{c}^\top \mathbf{x}_*$.

3.0.1 Solución:

En este caso, lo que haremos será resolver el problema de optimización

$$\max\{x_1 + 2x_2 + x_3 + x_4\},$$

pues el problema de minimización con las restricciones proporcionadas tiene como solución la solución trivial trivial $x_1 = x_2 = x_3 = x_4 = 0$.

El problema de maximización sujeto a las mismas reestricciones equivale a resolver $\min\{-x_1 - 2x_2 - x_3 - x_4\}$ con las mismas restricciones.

En primer lugar, escribiremos el problema en forma estándar. Como $x_i \geq 0$ para $i = 1, 2, 3, 4$ para obtener la forma estándar podemos ignorar el vector con componentes x_i^- (la parte negativa) pues esta es 0 bajo estas restricciones de positividad.

Por lo tanto, solo consideramos las variables de holgura no negativas x_5, x_6, x_7 de forma que se quiere resoler el siguiente problema de optimización

$$\begin{array}{rl} \min & -x_1 - 2x_2 - x_3 - x_4 \\ \text{sujeto a} & 2x_1 + x_2 + 3x_3 + x_4 + x_5 = 8 \\ & 2x_1 + 3x_2 + 4x_4 + x_6 = 12 \\ & 3x_1 + x_2 + 2x_3 + x_7 = 18 \\ & x_1, x_2, x_3, x_4, x_5, x_6, x_7 \geq 0 \end{array}$$

Si

$$A = \begin{pmatrix} 2 & 1 & 3 & 1 & 1 & 0 & 0 \\ 2 & 3 & 0 & 4 & 0 & 1 & 0 \\ 3 & 1 & 2 & 0 & 0 & 0 & 1 \end{pmatrix},$$

$\mathbf{x} = (x_1, x_2, x_3, x_4, x_5, x_6, x_7)^T$, $\mathbf{b} = (8, 12, 18)^T$ y

$$\mathbf{c} = (-1, -2, -1, -1, 0, 0, 0),$$

entonces el problema de programación lineal en forma estándar que debemos resolver es

$$\min \mathbf{c}^T \mathbf{x} \text{ sujeto a } A\mathbf{x} = \mathbf{b} \text{ y } \mathbf{x} \geq 0$$

Para poder obtener los vectores λ y $\mathbf{s}$ de multiplicadores de Lagrange es necesario tener la matriz básica $\mathbf{B}$ y por supuesto, tener identificadas las variables básicas y no básicas del problema de programación lineal.

Utilizamos el mismo código que en el ejercicio 2 para obtener la solución de este problema ya que tenemos la forma estándar.

[21]:
```
c = np.array([-1, -2, -1, -1, 0, 0 ,0])
b = np.array([8, 12, 18])
A = np.array([[2, 1, 3, 1, 1, 0, 0],
              [2, 3, 0, 4, 0, 1, 0],
              [3, 1, 2, 0, 0, 0, 1]])

m,n = A.shape
comb = list(combinations(list(range(n)), m))
print('Número de combinaciones:', len(comb))

dmin = None
for icols in comb:
    # Indices de las columnas seleccionadas
    jj = list(icols)
    # Matriz básica
    B     = A[:, jj]
    condB = np.linalg.cond(B)
    if condB>1.0e14:
        print('Es casi singular la matriz B con columnas', jj)
    else:
        # Solucion del sistema B*x=b
        xb = np.linalg.solve(B, b)
        # Solucion del problema en forma estándar
        x     = np.zeros(n)
        x[jj] = xb
        # Evaluación de la función objetivo
        f = np.vdot(c, x)
        # Se revisa si es vector x es factible. Claramente se cumple que A*x=b,
        # pero hay que verificar que x>=0.
        smsg  = 'No factible'
        bfact = False
        if sum(x>=0)==len(x):
            bfact = True
            smsg  = 'Factible'
        if bfact:
            # Si x es factible, almacenamos en xsol el punto x donde f es mínima
            if dmin==None:
                dmin = f
                xsol = x.copy()
            elif dmin>f:
                dmin = f
                xsol = x.copy()
        print("%6.1f %7.1f| % 8.2f % 8.2f % 8.2f % 8.2f % 8.2f % 8.2f % 8.2f|␣
 ↪%s" % (condB,
                           f,  x[0], x[1], x[2], x[3], x[4], x[5], x[6], smsg))
```

```
# Fijamos una tolerancia y hacemos cero las componentes de x que son menores que␣
 ↪la tolerancia
tol = (np.finfo(float).eps)**(3.0/4)
ii  = np.where(xsol<tol)[0]
xsol[ii] = 0.0

print('\nSolución del problema estándar:')
print('x*=', xsol)
print('Valor de la función objetivo en x*=', dmin)
```

```
Número de combinaciones: 35
   6.8    -3.2|     7.85    -1.23    -2.15     0.00     0.00     0.00     0.00|
No factible
  33.8   -23.3|     0.67    16.00     0.00    -9.33     0.00     0.00     0.00|
No factible
   6.0    -6.0|     6.00     0.00     0.00     0.00    -4.00     0.00     0.00|
No factible
  45.7    14.0|    10.00   -12.00     0.00     0.00     0.00    28.00     0.00|
No factible
  11.2    -7.0|     3.00     2.00     0.00     0.00     0.00     0.00     7.00|
Factible
   4.9    -4.7|     7.33     0.00    -2.00    -0.67     0.00     0.00     0.00|
No factible
  12.4    -6.0|     6.00     0.00     0.00     0.00    -4.00     0.00     0.00|
No factible
  10.2    -5.2|     7.60     0.00    -2.40     0.00     0.00    -3.20     0.00|
No factible
   7.9    -4.7|     6.00     0.00    -1.33     0.00     0.00     0.00     2.67|
No factible
   6.0    -6.0|     6.00     0.00     0.00     0.00    -4.00     0.00     0.00|
No factible
  24.9    -2.0|     6.00     0.00     0.00    -4.00     0.00    16.00     0.00|
No factible
  12.5    -4.7|     3.33     0.00     0.00     1.33     0.00     0.00     8.00|
Factible
   5.8    -6.0|     6.00     0.00     0.00     0.00    -4.00     0.00     0.00|
No factible
   8.9    -6.0|     6.00     0.00     0.00     0.00    -4.00     0.00    -0.00|
No factible
   8.9    -4.0|     4.00     0.00     0.00     0.00     0.00     4.00     6.00|
Factible
   9.9   -25.2|     0.00    17.60     0.20   -10.20     0.00     0.00     0.00|
No factible
   7.8   -15.0|     0.00     4.00     7.00     0.00   -17.00     0.00     0.00|
No factible
  48.0   -66.0|     0.00    38.00   -10.00     0.00     0.00  -102.00     0.00|
No factible
```

```
   5.1    -9.3|     0.00     4.00     1.33     0.00     0.00     0.00    11.33|
Factible
   6.9   -25.5|     0.00    18.00     0.00   -10.50     0.50     0.00     0.00|
No factible
  23.5   -26.0|     0.00    18.00     0.00   -10.00     0.00    -2.00     0.00|
No factible
  34.8   -28.0|     0.00    20.00     0.00   -12.00     0.00     0.00    -2.00|
No factible
  11.9   -36.0|     0.00    18.00     0.00     0.00   -10.00   -42.00     0.00|
No factible
   3.7    -8.0|     0.00     4.00     0.00     0.00     4.00     0.00    14.00|
Factible
  11.9   -16.0|     0.00     8.00     0.00     0.00     0.00   -12.00    10.00|
No factible
   8.1   -12.0|     0.00     0.00     9.00     3.00   -22.00     0.00     0.00|
No factible
  33.0    10.0|     0.00     0.00     9.00   -19.00     0.00    88.00     0.00|
No factible
   5.3    -4.7|     0.00     0.00     1.67     3.00     0.00     0.00    14.67|
Factible
   6.9    -9.0|     0.00     0.00     9.00     0.00   -19.00    12.00     0.00|
No factible
Es casi singular la matriz B con columnas [2, 4, 6]
   4.4    -2.7|     0.00     0.00     2.67     0.00     0.00    12.00    12.67|
Factible
Es casi singular la matriz B con columnas [3, 4, 5]
   4.3    -3.0|     0.00     0.00     0.00     3.00     5.00     0.00    18.00|
Factible
  17.9    -8.0|     0.00     0.00     0.00     8.00     0.00   -20.00    18.00|
No factible
   1.0     0.0|     0.00     0.00     0.00     0.00     8.00    12.00    18.00|
Factible

Solución del problema estándar:
x*= [ 0.      4.      1.3333  0.      0.      0.     11.3333]
Valor de la función objetivo en x*= -9.333333333333334
```

Calculamos el vector λ de las condiciones KKT para programación lineal como en el ejercicio anterior

[22]:
```
jj=list(np.squeeze(np.where(xsol>0)))
B=A[:,jj]
tilde_c=c[jj]
# B.T*lamb=tilde_c
lamb=np.linalg.solve(B.T,tilde_c)
print('El vecto lambda es: ',lamb)
```

```
El vecto lambda es:  [-0.3333 -0.5556  0.    ]
```

Y a partir de este vector λ calculamos el vector $\mathbf{s}$, que son los multiplicadores de Lagrange asociados a la restricción de desigualdad $\mathbf{x} \geq 0$ del problema de programación lineal en forma estándar.

[23]:
```
s=np.zeros(A.shape[1])
jj_complement=list(np.squeeze(np.where(xsol==0)))
N=A[:,jj_complement]
hat_c=c[jj_complement]
# hat_s=c_hat-N.T*lamb
hat_s=hat_c-N.T@lamb
s[jj_complement]=hat_s
print('El valor del vector s es: ',s)
```

```
El valor del vector s es:  [0.7778 0.     0.     1.5556 0.3333 0.5556 0.    ]
```

Finalmente, calculamos $\mathbf{b}^T\lambda$ y lo comparamos con el valor $\mathbf{c}^T\mathbf{x}$

[24]:
```
print('Valor de b.T lambda: ',np.vdot(b,lamb))
print('Valor de c.T x_ast: ',np.vdot(c,xsol))
```

```
Valor de b.T lambda:  -9.333333333333334
Valor de c.T x_ast:  -9.333333333333334
```

Ambos valores coinciden tratándose de las funciones objetivos del problema primal y dual del problema de programación lineal en forma estándar.

Ahora para obtener validar la solución del problema usamos también la librería `pulp`. Primero declaramos el modelo inicial

[11]:
```
# Creacción de una instancia de la  clase
model = LpProblem(name="small-problem", sense=LpMaximize)

# Variables de decisión
x1 = LpVariable(name="x1", lowBound=0)
x2 = LpVariable(name="x2", lowBound=0)
x3 = LpVariable(name="x3", lowBound=0)
x4 = LpVariable(name="x4", lowBound=0)




# Se agregan las restricciones del modelo
model += (2*x1 + x2 + 3*x3 + x4 <= 8, "R1")
model += (2*x1 + 3*x2 + 4*x4  <= 12, "R2")
model += (3*x1 + x2 + 2*x3  <= 18, "R3")

# Función objetivo
model += x1+2*x2+x3+x4

print(model)
```

```
small-problem:
MAXIMIZE
1*x1 + 2*x2 + 1*x3 + 1*x4 + 0
SUBJECT TO
R1: 2 x1 + x2 + 3 x3 + x4 <= 8

R2: 2 x1 + 3 x2 + 4 x4 <= 12

R3: 3 x1 + x2 + 2 x3 <= 18

VARIABLES
x1 Continuous
x2 Continuous
x3 Continuous
x4 Continuous
```

Posteriormente calculamos la solución

[12]:
```
status = model.solve(PULP_CBC_CMD(msg=False))

print("Resultado: ", model.status, " | ", LpStatus[model.status])

print("Valor de la funciónn objetivo: " , model.objective.value())


print('Solución:')
for var in model.variables():
    print("%10s: %f"  % (var.name, var.value()) )

print('\nVariables de holgura:')
for name, constraint in model.constraints.items():
    print("%10s: %f" % (name, constraint.value()) )
```

```
Resultado:  1  |  Optimal
Valor de la funciónn objetivo:  9.3333333
Solución:
        x1: 0.000000
        x2: 4.000000
        x3: 1.333333
        x4: 0.000000

Variables de holgura:
        R1: -0.000000
        R2: 0.000000
        R3: -11.333333
```

Y la solución obtenida por esta librería coincide con la que hemos obtenido hallando las soluciones básicas factibles.