Curso de Optimización (DEMAT)

Tarea 10

Descripción:	Fechas
Fecha de publicación del documento:	**Mayo 12, 2022**
Fecha límite de entrega de la tarea:	**Mayo 17, 2022**

Indicaciones

- **Envie el notebook que contenga los códigos y las pruebas realizadas de cada ejercicio.**
- **Si se requiren algunos scripts adicionales para poder reproducir las pruebas, agreguelos en un ZIP junto con el notebook.**
- **Genere un PDF del notebook y envielo por separado.**

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.**

Descripción del proyecto

Pryecto: Implementación de un algoritmo genético para resolver el problema $3-SAT$ **(Boolean satisfiability problem)**

Introducción:

La evolución a través de la selección natural de una población de individuos elegidos al azar se puede considerar como una búsqueda a través del espacio de posibles valores cromosómicos. En ese sentido, un algoritmo genético (AG) es una búsqueda estocástica de una solución óptima a un problema dado.

El problema SAT es el problema de saber si, dada una expresión booleana con variables y sin cuantificadores, hay alguna asignación de valores para sus variables que hace la expresión verdadera.

En el proyecto se implementará un algoritmo genético (AG).

Características que simula un AG acerca de la selección natural:

– Un sistema biológico consta de una población de individuos, muchos de los cuales tienen la habilidad de reproducirse.

– Los individuos tienen una vida útil finita.

– Hay variación en la población.

– La habilidad para sobrevivir está correlacionada positivamente con la habilidad de reproducirse.

Características de un AG:

– Representación: cadenas de bits (1s y 0s)

– Selección: Ruleta (Proporcional, esto se explícara más detalladamente en la presentación)

– Variaciones: Cruza.

Breve Pseudocodigo: Inicializar $t = 0$

– Inicializar $P(t)$

– Evaluar estructuras en $P(t)$

– Mientras no se llegue a la condición de paro, hacer $t = t + 1$

• Seleccionar $R(t)$ **de** $P(t-1)$

• Cruzar R(t) para obtener $C(t)$

• Mutar C(t) para obtener $C'(t)$

• Evaluar estructuras en $C'(t)$

• Reemplazar $P(t)$ **con** $C'(t)$ **y** $P(t-1)$

Mencionare las aplicaciones de estos códigos y su relevancia en la ciencia.

Bibliografía: Computational Intelligence, Second Edition. ***Andries P. Engelbrecht. University of Pretoria, South Africa.***

Ejercicio 2. (3 puntos)

Considere el ejemplo visto en clase:

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

Vimos que se puede escribir en forma estándar como:

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

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

Las condiciones KKT son:

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

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 $\mathbf{s}$

y por la condición de complementaridad (5), sabemos cuáles son las componentes de s que son cero y cuáles deben ser calculadas. Use eso y la condición (1) para calcular λ y las componentes de s desconocidas. Imprima los vectores λ y 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 $\mathbf{c}^\top \mathbf{x}_*$.

Solución:

In [1]:

```
# Pongo el codigo para obtener la solución x que calculamos en clase

import numpy as np
from itertools import combinations

#Coficientes de la función a minimizar
c = np.array([-1, -1, 0, 0, 0, 0 ])

#Restricciones
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 toler
ancia
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
```

In [2]:

```
# Solucion del ejercicio
lam=np.linalg.solve(A[:,[0,1,3,5]].T,c[[0,1,3,5]])
print('lambda=', lam)
s = c - A.T@lam
print('s=', s)
```

```
lambda= [-0.04166667  0.          1.08333333 -0.        ]
s= [0.         0.         0.04166667 0.         1.08333333 0.        ]
```

In [3]:

```
print(b.T@lam)
print(c.T@xsol)
```

```
-51.25000000000001
-51.25
```

Ejercicio 3. (3 puntos)

Considere el problema

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

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 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 $\mathbf{c}^\top \mathbf{x}_*$
.

Solución:

Forma estándar

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

In [4]:

```
#Coficientes de la función a minimizar
c = np.array([-1, -2, -1, -1, 0, 0, 0])

#Restricciones
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| %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 toler
ancia
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| No factible
  33.8   -23.3|     0.67    16.00     0.00    -9.33     0.00     0.00| No factible
   6.0    -6.0|     6.00     0.00     0.00     0.00    -4.00     0.00| No factible
  45.7    14.0|    10.00   -12.00     0.00     0.00     0.00    28.00| No factible
  11.2    -7.0|     3.00     2.00     0.00     0.00     0.00     0.00| Factible
   4.9    -4.7|     7.33     0.00    -2.00    -0.67     0.00     0.00| No factible
  12.4    -6.0|     6.00     0.00     0.00     0.00    -4.00     0.00| No factible
  10.2    -5.2|     7.60     0.00    -2.40     0.00     0.00    -3.20| No factible
   7.9    -4.7|     6.00     0.00    -1.33     0.00     0.00     0.00| No factible
   6.0    -6.0|     6.00     0.00     0.00     0.00    -4.00     0.00| No factible
  24.9    -2.0|     6.00     0.00     0.00    -4.00     0.00    16.00| No factible
  12.5    -4.7|     3.33     0.00     0.00     1.33     0.00     0.00| Factible
   5.8    -6.0|     6.00     0.00     0.00     0.00    -4.00     0.00| No factible
   8.9    -6.0|     6.00     0.00     0.00     0.00    -4.00     0.00| No factible
   8.9    -4.0|     4.00     0.00     0.00     0.00     0.00     4.00| Factible
   9.9   -25.2|     0.00    17.60     0.20   -10.20     0.00     0.00| No factible
   7.8   -15.0|     0.00     4.00     7.00     0.00   -17.00     0.00| No factible
  48.0   -66.0|     0.00    38.00   -10.00     0.00     0.00  -102.00| No factible
   5.1    -9.3|     0.00     4.00     1.33     0.00     0.00     0.00| Factible
   6.9   -25.5|     0.00    18.00     0.00   -10.50     0.50     0.00| No factible
  23.5   -26.0|     0.00    18.00     0.00   -10.00     0.00    -2.00| No factible
  34.8   -28.0|     0.00    20.00     0.00   -12.00     0.00     0.00| No factible
  11.9   -36.0|     0.00    18.00     0.00     0.00   -10.00   -42.00| No factible
   3.7    -8.0|     0.00     4.00     0.00     0.00     4.00     0.00| Factible
  11.9   -16.0|     0.00     8.00     0.00     0.00     0.00   -12.00| No factible
   8.1   -12.0|     0.00     0.00     9.00     3.00   -22.00     0.00| No factible
  33.0    10.0|     0.00     0.00     9.00   -19.00     0.00    88.00| No factible
   5.3    -4.7|     0.00     0.00     1.67     3.00     0.00     0.00| Factible
   6.9    -9.0|     0.00     0.00     9.00     0.00   -19.00    12.00| 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| 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| Factible
  17.9    -8.0|     0.00     0.00     0.00     8.00     0.00   -20.00| No factible
   1.0     0.0|     0.00     0.00     0.00     0.00     8.00    12.00| Factible

Solución del problema estándar:
x*= [ 0.          4.          1.33333333  0.          0.          0.
 11.33333333]
Valor de la función objetivo en x*= -9.333333333333334
```

In [5]:

```
lam=np.linalg.solve(A[:,[1,2,6]].T,c[[1,2,6]])
print(lam)
s = c - A.T@lam
print('s=', s)
```

```
[-0.33333333 -0.55555556  0.        ]
s= [0.77777778 0.         0.          1.55555556 0.33333333 0.55555556
 0.        ]
```

In [6]:

```
print(b.T@lam)
print(c.T@xsol)
```

```
-9.333333333333334
-9.333333333333334
```

Notemos que los valores (en ambos ejercicios) de $\mathbf{c}^\top \mathbf{x}_* = \mathbf{b}^\top \lambda$
son iguales como se deseaba para comparar el método simple con el dual.