Tarea 1 parte 2

P1) 1)

a) $X_{ij} = \begin{cases} 1 & \text{paciente } i \text{ se atiende sala } j \\ 0 & \sim \end{cases}$

Como se trabajará con el problema relajado $\Rightarrow X_{ij} \in [0,1]$ para así obtener solo variables continuas. También se puede escribir como $0 \leq X_{ij} \leq 1$

La función objetivo del problema será:

$$\max \sum_{i=1}^{m} \sum_{j=1}^{n} t_{ij} \cdot X_{ij}$$

ya que queremos maximizar el tiempo de utilización de las salas de operación, así, debemos sumar los tiempos en donde el paciente i se operó en la sala j

2) restricciones del problema:

2.i) todo paciente $i \in \{1, \ldots, m\}$ es atendido en a lo más una sala de operación

Esta restricción se puede escribir como:

$$\sum_{j=1}^{n} X_{ij} \leq 1$$

tomando un paciente fijo (i está fijo) y sumando en el índice j, X_{ij} da cuenta de que a lo más uno de los valores que puede tomar sea uno.

2.ii) En una sala $j \in \{1, \ldots, n\}$ el tiempo total de operación de los pacientes que son atendidos es a lo más el periodo de tiempo T. Esta restricción se puede escribir como:

$$\sum_{i=1}^{m} X_{ij} \cdot t_{ij} \leq T$$

donde el índice j está fijo y $j \in \{1, \ldots, n\}$, se suma en el índice i, puesto que si el paciente i está en la sala $j \Rightarrow$ sumo el tiempo correspondiente donde la cantidad total de tiempo no debe ser mayor a T

3) función objetivo: $\sum_{i=1}^{m} \sum_{j=1}^{n} t_{ij} \cdot X_{ij}$, identificando a

$$X = X_{ij} = \begin{pmatrix} X_{11} \\ X_{12} \\ \vdots \\ X_{1n} \\ X_{21} \\ X_{22} \\ \vdots \\ X_{m1} \\ X_{(m-1)n} \\ X_{mn} \end{pmatrix}$$

y $t_{ij} = C^t$, donde C^t debe ser un vector de $1 \times mn$, para que las dimensiones sean consistentes y además contiene los tiempos asociados:

$$C^t = (t_{11} \; t_{12} \dots t_{1n} \; t_{21} \; t_{22} \dots t_{m1} \; t_{(m-1)n} \; t_{mn})$$

La restricción a modo general es: $\begin{bmatrix} 1 \\ t_{ij} \end{bmatrix} X_{ij} \leq \begin{bmatrix} 1 \\ T \end{bmatrix}$

Por indicación: $A = \begin{bmatrix} A_1 \\ A_2 \end{bmatrix}$ y $b = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}$

Analizando cada sub-matriz por separado:

A_1] $\sum_{j=1}^{n} X_{ij} \leq 1$, para i fijo $i \in \{1, \dots, m\}$

$i=1$: $X_{11} + \dots + X_{1m} \leq 1$

$\vdots$

$i=m$: $X_{m1} + \dots + X_{mn} \leq 1$

$$\Rightarrow A_1 = \begin{bmatrix} 1 \text{ n veces} & \dots & 0 \text{. nm}-\text{n veces} & & \\ 0 \text{ n veces} & \dots & 1 \text{ n veces} & \dots & 0 \text{ mn}-2\text{n veces} \\ 0 \text{ 2n veces} & \dots & 1 \text{ n veces} & \dots & 0 \text{ mn}-3\text{n veces} \\ & & \vdots & & \\ 0 \text{ (m-1)n veces} & \dots & 1 \text{ n veces} & \dots & \end{bmatrix}$$

A_1 de dimensiones $m \times (mn)$.

A_2] $\sum_{i=1}^{m} X_{ij} \cdot t_{ij} \leq T$ para j fijo $j \in \{1, \dots, n\}$

$j=1$: $X_{11} t_{11} + X_{21} t_{21} + \dots + X_{m1} t_{m1} \leq T$

$\vdots$

$j=n$: $X_{1n} t_{1n} + X_{2n} t_{2n} + \dots + X_{mn} t_{mn} \leq T$

A_2 tiene submatrices de $n \times n$ con tiempos indicados en el índice t_{ij} ubicados en la diagonal de la submatriz, por lo tanto la dimensión de A_2 es de $n \times (mn)$.

$$A_2 = \begin{bmatrix} t_{11} & 0 & \cdots & t_{21} & & \cdots & t_{n1} & \cdots & 0 \\ 0 & t_{12} & & \vdots & t_{22} & & \vdots & \ddots & \vdots \\ \vdots & & \ddots & & & \ddots & & & \\ \substack{(n-1) \\ \text{veces } 0} & & t_{1n} & 0 & \cdots & t_{2n} & 0 & & t_{mn} \end{bmatrix}$$

$$\Rightarrow A = \begin{bmatrix} A_1 \\ A_2 \end{bmatrix} = \begin{bmatrix} 1 \text{ n veces } 0 \text{ mn-n veces} \cdots \\ 0 \text{ n veces} \cdots 1 \text{ n veces} \cdots 0 \text{ mn-2n veces} \cdots \\ \vdots \\ 0 \text{ (m-1)n veces} \cdots 1 \text{ n veces} \cdots \\ t_{11} \; 0 \cdots t_{21} \cdots 0 \; \cdots t_{n1} \; 0 \\ \ddots \quad \ddots \quad \ddots \\ 0 \quad t_{1n} \quad 0 \quad t_{2n} \quad t_{mn} \end{bmatrix}$$

Así A es una matriz de dimensiones $(m+n) \times (mn)$.

b1| Matriz columna de unos de dimensiones $(m \times 1)$,

$$b_1 = \begin{pmatrix} 1 \\ \vdots \\ \text{m veces} \end{pmatrix}$$

b2| Matriz columna de T de dimensiones $(n \times 1) \Rightarrow b_2 = \begin{pmatrix} T \\ \vdots \\ \text{n veces} \end{pmatrix}$

$$\Rightarrow b = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} = \begin{pmatrix} 1 \\ \vdots \\ \text{m veces} \\ T \\ \vdots \\ \text{n veces} \end{pmatrix}$$

b es un vector columna de dimensiones $(m+n) \times 1$

Así las restricciones quedan

$$\begin{pmatrix} X_{11} + X_{12} + \cdots + X_{1n} \\ \vdots \\ X_{m1} + X_{m2} + \cdots + X_{mn} \end{pmatrix} \leq \begin{pmatrix} 1 \\ \vdots \\ \text{m veces} \end{pmatrix} \text{ y } \begin{pmatrix} X_{11}t_{11} + \cdots + X_{m1}t_{m1} \\ \vdots \\ X_{1n}t_{1n} + \cdots + X_{mn}t_{mn} \end{pmatrix} \leq \begin{pmatrix} T \\ \vdots \\ \text{n veces} \end{pmatrix}$$

4) pasando a problema de minimización:

$$\Rightarrow \min \; -c^t x$$
$$\text{s.a} \quad Ax \leq b$$
$$0 \leq x \leq 1$$

donde se reemplazan los valores obtenidos anteriormente

b) f) primal: $\max \; c^t x$ s.a $Ax \leq b$, $0 \leq x \leq 1$ dual: $\min \; b^t y$ s.a $A^T y \geq c$, y

Dado los valores anteriores:

$b^t = [1 \dots m \text{ veces } T \dots n \text{ veces}]$ vector fila de $1 \times (m+n)$

$$C = \begin{bmatrix} t_{11} \\ t_{12} \\ \vdots \\ t_{(m-1)n} \\ t_{mn} \end{bmatrix}$$ vector columna de $(mn) \times 1$

$$A^t = \left[\begin{array}{ccc|ccc} 1 & 0 & & t_{11} & 0 & 0 \\ \vdots & n \text{ veces} & & \vdots & t_{12} & \ddots \; t_{1n} \\ n\text{ veces } 0 & 1 \; n \text{ veces} & \dots & \vdots & & \\ & \vdots & & & & \\ & 0 \; mn-n \text{ veces} & & t_{m1} & & \ddots \; t_{mn} \end{array}\right]$$ matriz de $(mn) \times (m+n)$

Reemplazando los valores en el dual se tiene

Tarea 1 parte 2

Pía Contreras

19.840.187-0

Un poco de explicación del código y matrices que se obtuvieron

Lo primero que se hizo fue cambiar la semilla y fijarla, con el rut correspondiente, al igual que en parte 1 de la tarea.

Luego se definió el número de salas de cirugías (n=2 o n=3, el n en este caso sólo puede ser uno de esos dos números) y luego el número de pacientes (m=4 o m=5, el m en este caso sólo puede ser uno de estos dos números), luego se definió T en el cual solo puede haber m operaciones, en este caso se eligió T=10.

Los números anteriores quieren decir que el hospital cuenta con 2 pabellones, y 4 operaciones diarias, otro hospital cuenta con 3 pabellones y 5 operaciones diarias.

Se creó la matriz de los tiempos t_ij en intervalos aleatorios entre a y b, donde a=tmin=3, es decir, 3 es el tiempo mínimo del paciente i en la sala j y b=tmax=5 es el tiempo máximo del paciente i en la sala j, se siguió la instrucción de que t_ij=a+(b-a)*numeroaleatorio.

Con este arreglo, se creó una matriz de valores aleatorios en cada componente, de dimensiones nxm.

La matriz que se obtuvo para n=2 y m=4 para t_ij fue:

```
[[3.34755953 4.9724619 ]
 [4.57083893 4.8684385 ]
 [4.82529513 4.17403817]
 [4.93673813 4.5137577 ]]
```

La matriz t_ij que se obtuvo para n=3 y m=5 fue :

```
[[3.34755953 4.9724619  4.57083893]
 [4.8684385  4.82529513 4.17403817]
 [4.93673813 4.5137577  3.91561052]
 [3.30035041 4.34136338 4.14828654]
 [4.31344414 4.70909587 4.91792262]]
```

Se definió c, el cual resulta ser un arreglo de la matriz t_ij, el cual las componentes de la matriz las traspasa al vector de dimensiones 1xm*n.

Para n=2 y m=4, c resulta ser un vector de dimensiones 1x8, y es como sigue:

```
[3.34755953 4.9724619  4.57083893 4.8684385  4.82529513 4.17403817
 4.93673813 4.5137577 ]
```

Por otra parte, para n=3 y m=5, c resulta ser un vector de dimensiones 1x15 y es:

```
[3.34755953 4.9724619  4.57083893 4.8684385  4.82529513 4.17403817
 4.93673813 4.5137577  3.91561052 3.30035041 4.34136338 4.14828654
 4.31344414 4.70909587 4.91792262]
```

De la función objetivo $\sum_{i=1}^{m}\sum_{j=1}^{n} t_{ij}x_{ij}.$ y de las restricciones (las cuales se encuentran mejor escritas en la parte anterior), se puede desprender que A=[A1,A2] y b=[b1,b2]

Por lo tanto, se debe desprender las submatrices tanto de A como de b.

Partiendo por el **vector b**:

-b1: es un vector de uno de dimensiones mx1, para n=2 y m=4 de 4x1

```
[[1.]
 [1.]
 [1.]
 [1.]]
```

Para n=3 y m=5, es un vector de dimensiones 5x1

```
[[1.]
 [1.]
 [1.]
 [1.]
 [1.]]
```

-b2: vector de T de dimensiones nx1, para n=2 y m=4 de 2x1

```
[[10.]
 [10.]]
```

Para n=3 y m=5, es un vector de dimensiones 3x1

```
[[10.]
 [10.]
 [10.]]
```

Matriz A:

El vector x está definido de la siguiente manera y de dimensiones (m*n) x1:

$$x = \begin{pmatrix} x_{11} \\ x_{12} \\ \vdots \\ x_{1n} \\ x_{21} \\ x_{22} \\ \vdots \\ x_{m1} \\ \vdots \\ x_{(m-1)n} \\ x_{mn} \end{pmatrix}$$

-A1: Para que la multiplicación con x tenga sentido, el ancho de esta matriz debe ser (m*n). Por la forma de x vemos que habrá m resultados de la sumatoria de x_ij distintos uno para cada i. Así se define A1 tal que multiplique por unos y sume para cada i distintos y así generar m resultados.

Para esto, las primeras n componentes del vector x se multipliquen por uno y el resto de las componentes se multipliquen por cero. Así como se quiere hacer para cada valor de i A1 debe tener m filas y coincidir la cantidad de columnas con el largo del vector x, es decir, si x es de largo (m*n), entonces A1 tendrá m*n columnas.

La matriz se creó usando ciclos for.

Lo que entrega para n=2 y m=4 es una matriz de dimensiones 4x8 :

```
[[1. 1. 0. 0. 0. 0. 0. 0.]
 [0. 0. 1. 1. 0. 0. 0. 0.]
 [0. 0. 0. 0. 1. 1. 0. 0.]
 [0. 0. 0. 0. 0. 0. 1. 1.]]
```

La matriz A1 para n=3 y m=5 es una matriz de dimensiones 5x15, y queda como sigue:

```
[[1. 1. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 1. 1. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 1. 1. 1. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 1. 1. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 1. 1.]]
```

-A2:es la multiplicación de x con el tiempo asociado. En este caso, se hizo la matriz a mano, es decir, solo funciona para un (m=4 y n=2) y (m=5 y n=3). En este caso para un j fijo se cumple que la sumatoria de la multiplicación en el índice i del x con el tiempo asociado debe ser menor o igual a T. Por el análisis dimensional hecho anteriormente (A1), se concluye que A2 es de dimensiones nx(m*n).

Para n=2 y m=4 la matriz A2 es de dimensiones 2x8 y queda como:

```
[[3.34755953 0.         4.57083893 0.         4.82529513 0.
  4.93673813 0.        ]
 [0.         4.9724619  0.         4.8684385  0.         4.17403817
  0.         4.5137577 ]]
```

Para n=3 y m=5 la matriz A2 es de dimensiones 3x15 y queda como:

```
[[3.34755953 0.         0.         4.9724619  0.         0.
  4.93673813 0.         0.         3.30035041 0.         0.
  4.31344414 0.         0.        ]
 [0.         4.9724619  0.         0.         4.82529513 0.
  0.         4.5137577  0.         0.         4.34136338 0.
  0.         4.70909587 0.        ]
 [0.         0.         4.57083893 0.         0.         4.17403817
  0.         0.         3.91561052 0.         0.         4.14828654
  0.         0.         4.91792262]]
```

Finalmente se concatenan las matrices siguiendo las instrucciones del enunciado.

A tendrá dimensiones (m+n)x(m*n) .

Si n=2 y m=4, A será de dimensiones (6)x(8) y quedará como :

```
[[1.         1.         0.         0.         0.         0.
  0.         0.        ]
 [0.         0.         1.         1.         0.         0.
  0.         0.        ]
 [0.         0.         0.         0.         1.         1.
  0.         0.        ]
 [0.         0.         0.         0.         0.         0.
  1.         1.        ]
 [3.34755953 0.         4.57083893 0.         4.82529513 0.
  4.93673813 0.        ]
 [0.         4.9724619  0.         4.8684385  0.         4.17403817
  0.         4.5137577 ]]
```

Si n=3 y m=5, A será de dimensiones (7)x(15) y quedará como:

```
[[1.         1.         1.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.        ]
 [0.         0.         0.         1.         1.         1.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.        ]
 [0.         0.         0.         0.         0.         0.
  1.         1.         1.         0.         0.         0.
  0.         0.         0.        ]
 [0.         0.         0.         0.         0.         0.
  0.         0.         0.         1.         1.         1.
  0.         0.         0.        ]
 [0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  1.         1.         1.        ]
 [3.34755953 0.         0.         4.9724619  0.         0.
  4.93673813 0.         0.         3.30035041 0.         0.
  4.31344414 0.         0.        ]
 [0.         4.9724619  0.         0.         4.82529513 0.
  0.         4.5137577  0.         0.         4.34136338 0.
  0.         4.70909587 0.        ]
 [0.         0.         4.57083893 0.         0.         4.17403817
  0.         0.         3.91561052 0.         0.         4.14828654
  0.         0.         4.91792262]]
```

Mientras que el vector b, concatenando b1 y b2 queda de dimensiones (m+n)x1

Si n=2 y m=4, b será de dimensiones (6)x1 y queda como sigue:

```
[[ 1.]
 [ 1.]
 [ 1.]
 [ 1.]
 [10.]
 [10.]]
```

Mientras que para n=3 y m=5, el vector b será de dimensiones 8x1 y queda como sigue:

```
[[ 1.]
 [ 1.]
 [ 1.]
 [ 1.]
 [ 1.]
 [10.]
 [10.]
 [10.]]
```

c) Así se tiene todo para utilizar en la función linprog la cual, encontrará un x tal que se minimizará el problema, por lo tanto, debemos usar -c.

La función se define de la siguiente manera: x=linprog(-c,A,b,bounds=(0,1),method='simplex'), donde bounds está entre 0 y 1 ya que los x_ij están entre 0 y 1 en el problema relajado.

Los resultados obtenidos para n=2 y m=4 fueron:

```
  status: 0
   slack: array([0.         , 0.         , 0.         , 0.         , 0.23796674,
       0.1590996 , 1.         , 0.         , 1.         , 0.         ,
       0.         , 1.         , 0.         , 1.         ])
 success: True
     fun: -19.602933662105
       x: array([0., 1., 0., 1., 1., 0., 1., 0.])
 message: 'Optimization terminated successfully.'
     nit: 4
```

Donde el vector x que minimiza la función es x=[0,1,0,1,1,0,1,0] en 4 iteraciones y resultado exitoso, además se obtiene un x del problema relajado el cuál toma valores entre 0 y 1, en este caso toma 0 y 1.

Los resultados obtenidos para n=3 y m=5 fueron:

```
 status: 0
  slack: array([0.         , 0.         , 0.         , 0.         , 0.         ,
      0.09079997, 0.68617472, 5.08207738, 1.         , 0.         ,
      1.         , 0.         , 1.         , 1.         , 0.         ,
      1.         , 1.         , 1.         , 0.         , 1.         ,
      1.         , 1.         , 0.         ])
uccess: True
    fun: -24.036924523961098
      x: array([0., 1., 0., 1., 0., 0., 1., 0., 0., 0., 1., 0., 0., 0., 1.])
essage: 'Optimization terminated successfully.'
    nit: 5
```

Donde el vector x que minimiza la función objetivo es x= [0,1,0,1,0,0,1,0,0,0,1,0,0,0,1] en 5 iteraciones con resultado de optimización exitoso, además se obtiene un x del problema relajado el cual toma valores entre 0 y 1, en este caso toma 0 y 1.

Al calcular c transpuesto por x se obtiene:

```
19.602933662105
```

Para los valores obtenidos anteriormente se puede ver que, dentro del conjunto factible, es decir, las restricciones del problema, el x que minimiza la función es el mismo para ambos, así, si vemos el conjunto de restricciones como un poliedro, el óptimo será visto el vértice que permite minimizar la función.

Se vio que este problema (problema primal) tiene óptimo, por lo tanto, por teorema, existe solución básica factible, que es óptima, también el problema primal es factible y acotado, por lo tanto, existe solución óptima, la cual se puede escoger básica factible.

Así x es tal que minimiza la función objetivo, y es el vértice del poliedro (siguiendo la dirección -c) conjunto factible (que lo determinan las restricciones).

d) Variando los valores de t_ij en [a+0.5,b+0.5], en n=2 y m=4, se obtiene:

```
 status: 0
  slack: array([0.        , 0.1566378 , 0.14309691, 0.        , 0.        ,
        0.        , 1.        , 0.        , 1.        , 0.1566378 ,
        0.14309691, 1.        , 0.        , 1.        ])
success: True
    fun: -20.0
      x: array([0.        , 1.        , 0.        , 0.8433622 , 0.85690309,
        0.        , 1.        , 0.        ])
message: 'Optimization terminated successfully.'
    nit: 4
```

Donde se ve que el vector x cambia respecto al x obtenido anteriormente.

Si movemos en -0.8 a y b, se obtiene:

```
 status: 0
  slack: array([0.        , 0.        , 0.        , 0.        , 1.83796674,
        1.7590996 , 1.        , 0.        , 1.        , 0.        ,
        0.        , 1.        , 0.        , 1.        ])
success: True
    fun: -16.402933662105
      x: array([0., 1., 0., 1., 1., 0., 1., 0.])
message: 'Optimization terminated successfully.'
    nit: 4
```

Donde se ve que el vector x no cambia respecto al primero.

Si sumo 0.8 en a y b, se vio que el vector x si cambio respecto al primero y si resto 0.5 en a y b se vio que el vector x no cambió respecto al primero.

De esto se puede desprender de que si sumo valores a los valores fijo a [a,b] entonces cambiará el vector x, mientras que si resto valores respecto al [a,b] fijo, entonces el vector x no cambia.

Para n=3 y m=5, variando los valores t_ij en +0.5, se obtiene:

```
 status: 0
  slack: array([0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 3.43130248, 1.        , 0.        ,
        1.        , 0.16614095, 0.83385905, 1.        , 0.        ,
        1.        , 1.        , 1.        , 0.2475697 , 0.7524303 ,
        1.        , 1.        , 0.        ])
success: True
    fun: -26.481956668764596
      x: array([0.        , 1.        , 0.        , 0.83385905, 0.16614095,
        0.        , 1.        , 0.        , 0.        , 0.        ,
        0.7524303 , 0.2475697 , 0.        , 0.        , 1.        ])
message: 'Optimization terminated successfully.'
    nit: 8
```

Donde se ve que el vector x cambia respecto al x inicial.

Ahora, si se varía [a-0.8,b-0.8], se obtiene:

```
 status: 0
  slack: array([0.        , 0.        , 0.        , 0.        , 0.        ,
       1.69079997, 2.28617472, 5.88207738, 1.        , 0.        ,
       1.        , 0.        , 1.        , 1.        , 0.        ,
       1.        , 1.        , 1.        , 0.        , 1.        ,
       1.        , 1.        , 0.        ])
success: True
    fun: -20.0369245239611
      x: array([0., 1., 0., 1., 0., 0., 1., 0., 0., 0., 1., 0., 0., 0., 1.])
message: 'Optimization terminated successfully.'
    nit: 5
```

Donde se ve que el vector x no cambia.

Si sumo 0.8 a [a,b] se ve que el vector x si cambia, por otro lado si resto 0.5 a [a,b] se ve que el vector x no cambia.

Lo anterior indica, que hay cierto rango de igualdad para el vector óptimo entorno a [a,b], el cual tentativamente podría ser que si sumo al intervalo, entonces cambia el valor óptimo, por otro lado si resto al intervalo, no cambia x.

Como lo anterior no es tan representativito se puede decir que a partir de cierta variación, todas los pabellones estarían ocupados, por lo que no podrían programarse más operaciones.

Las variaciones obtenidas en el vector x se pueden desprender de la fórmula de sensibilidad:

- **Formula de sensibilidad:** Sean (π, s) las variables duales del problema primal asociadas al óptimo x. Entonces un pequeño cambio al vector b, que denotaremos Δb produce una perturbación en el valor óptimo del problema primal de la forma: $c^\top \Delta x = \Delta b^\top \pi$

e) Variando T en +0.5 para n=2 y m=4, se obtuvo:

```
 status: 0
  slack: array([0.        , 0.        , 0.        , 0.        , 0.73796674,
       0.6590996 , 1.        , 0.        , 1.        , 0.        ,
       0.        , 1.        , 0.        , 1.        ])
success: True
    fun: -19.602933662105
      x: array([0., 1., 0., 1., 1., 0., 1., 0.])
message: 'Optimization terminated successfully.'
    nit: 4
```

Donde se ve que el vector x no cambia respecto al primer vector.

Si variamos T en -0.8 se obtiene que:

```
 status: 0
  slack: array([0.        , 0.13164393, 0.11647645, 0.        , 0.        ,
      0.        , 1.        , 0.        , 1.        , 0.13164393,
      0.11647645, 1.        , 0.        , 1.        ])
success: True
    fun: -18.4
      x: array([0.        , 1.        , 0.        , 0.86835607, 0.88352355,
      0.        , 1.        , 0.        ])
message: 'Optimization terminated successfully.'
    nit: 4
```

Donde se puede ver que el vector x si cambio respecto al primero.

Si resto 0.5 a T, entonces el vector x cambia, y si sumo 0.8 el vector x no cambia. Luego de probar con más variaciones a T, se pudo ver que el vector x cambia pequeñas variaciones si T varía entre [-0.1,-1].

Para n=3 y m=5, al variar T en +0.5 se obtuvo:

```
 status: 0
  slack: array([0.        , 0.        , 0.        , 0.        , 0.        ,
      0.59079997, 1.18617472, 5.58207738, 1.        , 0.        ,
      1.        , 0.        , 1.        , 1.        , 0.        ,
      1.        , 1.        , 1.        , 0.        , 1.        ,
      1.        , 1.        , 0.        ])
success: True
    fun: -24.036924523961098
      x: array([0., 1., 0., 1., 0., 0., 1., 0., 0., 0., 1., 0., 0., 0., 1.])
message: 'Optimization terminated successfully.'
    nit: 5
```

Donde se ve que el vector x no cambia.

Ahora, si se hace variar T en -0.8 se obtiene:

```
 status: 0
  slack: array([0.        , 0.        , 0.        , 0.        , 0.        ,
      0.        , 0.        , 3.51571137, 1.        , 0.        ,
      1.        , 0.14262553, 0.85737447, 1.        , 0.        ,
      1.        , 1.        , 1.        , 0.18474279, 0.81525721,
      1.        , 1.        , 0.        ])
success: True
    fun: -23.995101624893465
      x: array([0.        , 1.        , 0.        , 0.85737447, 0.14262553,
      0.        , 1.        , 0.        , 0.        , 0.        ,
      0.81525721, 0.18474279, 0.        , 0.        , 1.        ])
message: 'Optimization terminated successfully.'
    nit: 8
```

Donde se ve que el vector x si cambia.

Por lo tanto, se puede desprender, que existen rangos para T, en los cuales el vector x se mantendrá igual que el “original”, mientras que si se escapa de esos rangos (delta), entonces el vector x cambiará su valor de óptimo.

Como lo anterior no es tan representativo pero representa una variación, se puede deducir que al aumentar T, aumenta el numero de pabellones ocupados, es decir, T influye en el tiempo en que se puede usar un pabellón, y como consecuencia podría tener que se podría aumentar el tiempo de utilización de cada pabellón.

Al igual que en el caso anterior, esto se puede demostrar por la fórmula de sensibilidad:

- **Formula de sensibilidad:** Sean (π, s) las variables duales del problema primal asociadas al óptimo x. Entonces un pequeño cambio al vector b, que denotaremos Δb produce una perturbación en el valor óptimo del problema primal de la forma: $c^\top \Delta x = \Delta b^\top \pi$

g) Para poder resolver numéricamente el dual, se utilizó la siguiente regla:

Primal	Dual
$\min_x \; c^\top x$ s.a. $Ax = b$ $x \geq 0$	$\max_y \; b^\top y$ s.a. $A^\top y \leq c$
$\min_x \; c^\top x$ s.a. $Ax \geq b$ $x \geq 0$	$\max_y \; b^\top y$ s.a. $A^\top y \leq c$ $y \geq 0$

El signo ‘-’ es para cambiar las desigualdades, y bounds entre 0 y None, indica que sólo está acotado por 0 por debajo.

En este caso lo que se hizo fue:

```
y=linprog(b,-A3,-c,bounds =(0,None),method='simplex')
print y
```

Donde A3 es la transpuesta de la matriz A.

Lo que se obtiene como valor óptimo (n=2 y m=4) es el x que sigue:

```
  status: 0
   slack: array([1.62490237, 0.        , 0.29759957, 0.        , 0.        ,
       0.65125696, 0.        , 0.42298043])
 success: True
     fun: 19.602933662105002
       x: array([4.9724619 , 4.8684385 , 4.82529513, 4.93673813, 0.        ,
       0.        ])
 message: 'Optimization terminated successfully.'
     nit: 15
```

Al calcular b transpuesto por y, se obtiene:

```
19.602933662105002
```

Por teorema de dualidad en programación lineal se tiene que:

- **Teorema de Dualidad en programación Lineal:**

  I) Si (PL) o (PL_d) tienen solución con valor de la función objetivo finito, entonces el otro problema también y los valores óptimos son iguales.

  II) Si (PL) o (PL_d) es no acotado, entonces el otro problema es infactible.

Tomando la sección i) se ve que los valores óptimos tanto del problema dual como del problema primal debiesen ser iguales, ya que en el problema primal se encontró solución con valor de la función objetivo finito.

Se vio en este caso, que los valores de c transpuesto por x y b transpuesto por y son iguales, esto demuestra el teorema anteriormente mencionado, es decir, calcular el problema dual o primal debería ser equivalente.

Para el dual, el numero de restricciones es m*n y el tamaño del vector solución es m+n, mientras que para el primal las restricciones son m+n y el tamaño del vector solución es m*n.

Finalmente, se concluye que todo problema primal tiene un problema dual asociado, donde se vio en clases que son formas distintas de abordar un problema, lo que se tiene por diferencia es que el método dual al disminuir las restricciones, el vector solución puede simplificar varios cálculos pues se disminuye la complejidad del problema.