Programación Lineal Primal Dual Clase 21

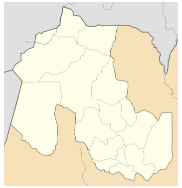
Investigación Operativa UTN FRBA
Curso: I4051
Docente: Martín Palazzo

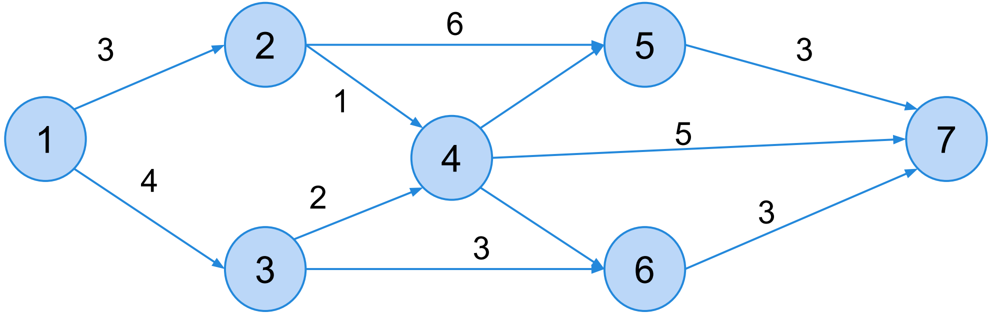
Programacion Lineal en grafos
Camino mas corto

Primal-Dual

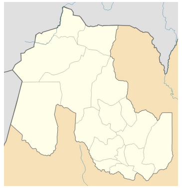
Programación Lineal: camino mas corto

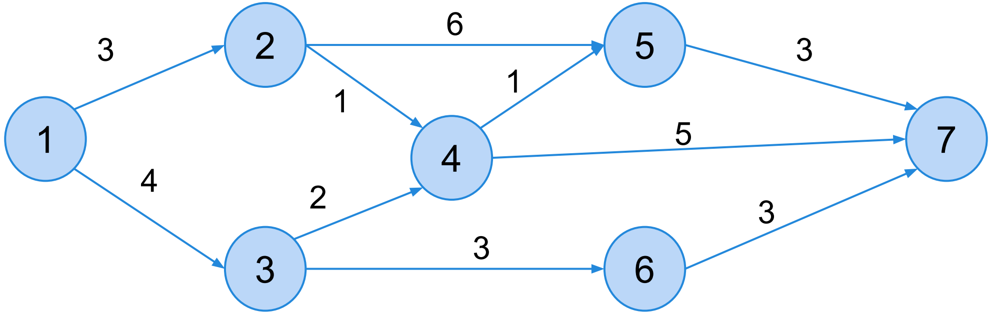
Queremos enviar paquetes por tierra aprovechando la red de centros de distribución del correo Argentino. Los paquetes se envía desde el centro de distribución en San Salvador de Jujuy y tienen que entregarse en el centro de distribución en Tierra del Fuego. Para ellos los paquetes tendrán que atravesar la red de centros de distribución de todo el país. Sabemos que el costo de transporte entre centros i-j es “d”. Queremos encontrar la ruta más corta entre ambos centros de distribución basándose en las distancias entre ellos.





Programación Lineal: camino mas corto





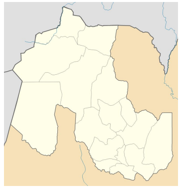
Para modelar el problema como programación lineal la variable de decisión será si un arco es o no parte de la ruta mas corta. Como hay 7 arcos entonces tenemos 7 variables de decisión.

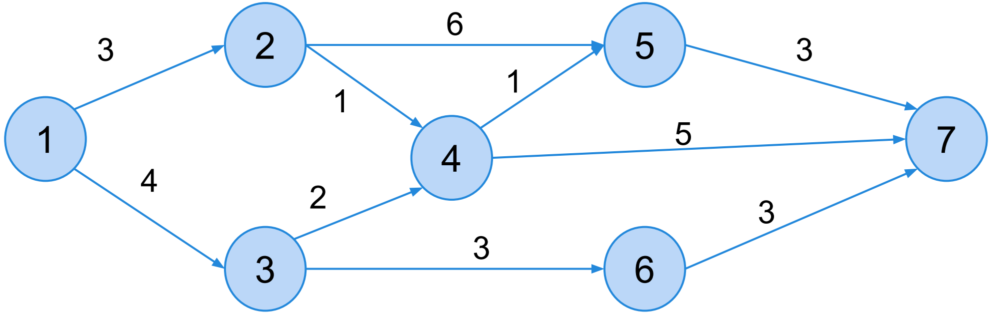
$$y = [y_{12}, y_{12}, y_{24}, y_{25}, y_{34}, y_{36}, y_{45}, y_{47}, y_{57}, y_{67}]$$

Además la función objetivo queda determinada como

$$\min_{y} \; w = \sum d_{ij} y_{ij}$$

Programación Lineal: camino mas corto

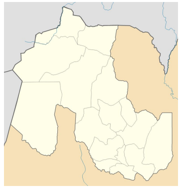


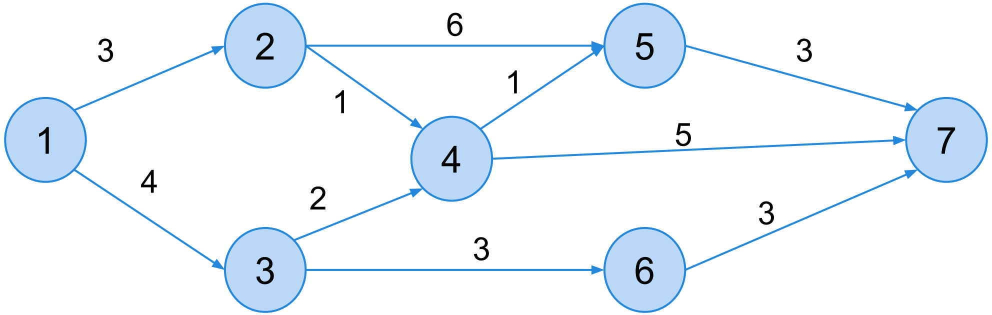


Además vamos a asumir que se envia una unidad de **flujo** desde el nodo origen 1 (Jujuy) al nodo destino 7 (TdF). Para cualquiera de los nodos intermedios el flujo de entrada debe ser igual al flujo de salida, para el nodo de origen el flujo saliente sera = 1 y el nodo destino el flujo entrante sera = 1. Se plantean restricciones de flujo para los nodos.

$$
\begin{aligned}
y_{12} + y_{13} &= 1 && \text{Nodo 1} \\
y_{12} - y_{24} - y_{25} &= 0 && \text{Nodo 2} \\
y_{13} - y_{34} - y_{36} &= 0 && \text{Nodo 3} \\
y_{24} + y_{34} - y_{45} - y_{47} &= 0 && \text{Nodo 4} \\
y_{25} + y_{45} - y_{57} &= 0 && \text{Nodo 5} \\
y_{36} - y_{67} &= 0 && \text{Nodo 6} \\
y_{57} + y_{67} &= 1 && \text{Nodo 7}
\end{aligned}
$$

Programación Lineal: camino mas corto





Tarea: resolver el problema propuesto en Python. Plantear el problema Dual y resolverlo también para verificar que la función objetivo es la misma en el argumento máximo de cada problema. Qué interpretación podemos darle al problema dual? Si en el primal estamos minimizando una distancia a recorrer, que maximizamos en el dual?

Programación Lineal: camino mas corto

```python
import pulp
# definimos si es un problema de minimizacion o maximizacion
linprog_primal = LpProblem("Primal", LpMinimize)

# definimos las variables de decision, el tipo de variable y la cota inferior
y12 = LpVariable('y12', lowBound=0, cat='Continuous')
y13 = LpVariable('y13', lowBound=0, cat='Continuous')
y24 = LpVariable('y24', lowBound=0, cat='Continuous')
y25 = LpVariable('y25', lowBound=0, cat='Continuous')
y34 = LpVariable('y34', lowBound=0, cat='Continuous')
y36 = LpVariable('y36', lowBound=0, cat='Continuous')
y45 = LpVariable('y45', lowBound=0, cat='Continuous')
y47 = LpVariable('y47', lowBound=0, cat='Continuous')
y57 = LpVariable('y57', lowBound=0, cat='Continuous')
y67 = LpVariable('y67', lowBound=0, cat='Continuous')

# primero agregamos la funcion objetivo
linprog_primal += 3*y12 + 4*y13 + 1*y24 + 6*y25 + 2*y34 + 3*y36 + 1*y45 + 5*y47 + 3*y57 + 3*y67 ,
"Funcion objetivo"

# luego agregamos restricciones
linprog_primal += y12 + y13 == 1 , "balance flujo nodo 1"
linprog_primal += y12 - y24 - y25 == 0, "balance flujo nodo 2"
linprog_primal += y13 - y34 - y36 == 0, "balance de flujo nodo 3"
linprog_primal += y24 + y34 - y45 - y47 == 0, "balance flujo nodo 4"
linprog_primal += y25 + y45 - y57 == 0, "balance flujo nodo 5"
linprog_primal += y36 - y67 == 0, "balance flujo nodo 6"
linprog_primal += y57 + y67 == 0, "balance flujo nodo 7"

# Resolver el problema con el solver de PULP
linprog_primal.solve()

# valor de la funcion objetivo
value(linprog_primal.objective)
```

$$y = [y_{12}, y_{12}, y_{24}, y_{25}, y_{34}, y_{36}, y_{45}, y_{47}, y_{57}, y_{67}]$$

$$\min_{y} \; w = \sum d_{ij} y_{ij}$$

$$\begin{aligned}
y_{12} + y_{13} &= 1 \\
y_{12} - y_{24} - y_{25} &= 0 \\
y_{13} - y_{34} - y_{36} &= 0 \\
y_{24} + y_{34} - y_{45} - y_{47} &= 0 \\
y_{25} + y_{45} - y_{57} &= 0 \\
y_{36} - y_{67} &= 0 \\
y_{57} + y_{67} &= 1
\end{aligned}$$

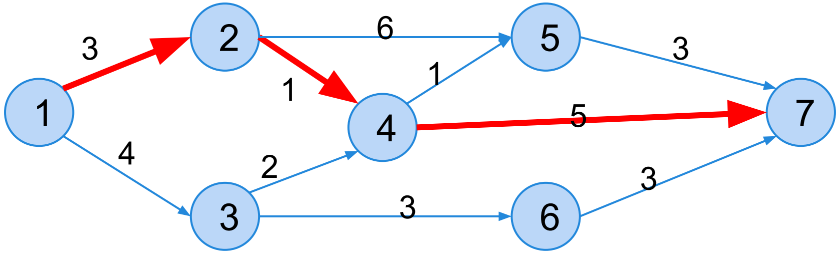
Ioperativ_clase21_shortest_path.ipynb

Programación Lineal: camino mas corto

```
# valor de la funcion objetivo
print(value(linprog_primal.objective))
9.0

# obtenemos el valor de la variable de decision X1....X12 en el punto
solucion_primal = np.array([[linprog_primal.variables()[0].varValue,
  linprog_primal.variables()[1].varValue,
  linprog_primal.variables()[2].varValue,
  linprog_primal.variables()[3].varValue,
  linprog_primal.variables()[4].varValue,
  linprog_primal.variables()[5].varValue,
  linprog_primal.variables()[6].varValue,
  linprog_primal.variables()[7].varValue,
  linprog_primal.variables()[8].varValue,
  linprog_primal.variables()[9].varValue,
  ]])

# imprimimos en pantalla las variables de decision en el optimo
print(solucion_primal)
[[1. 0. 1. 0. 0. 0. 0. 1. 0. 0.]]
```

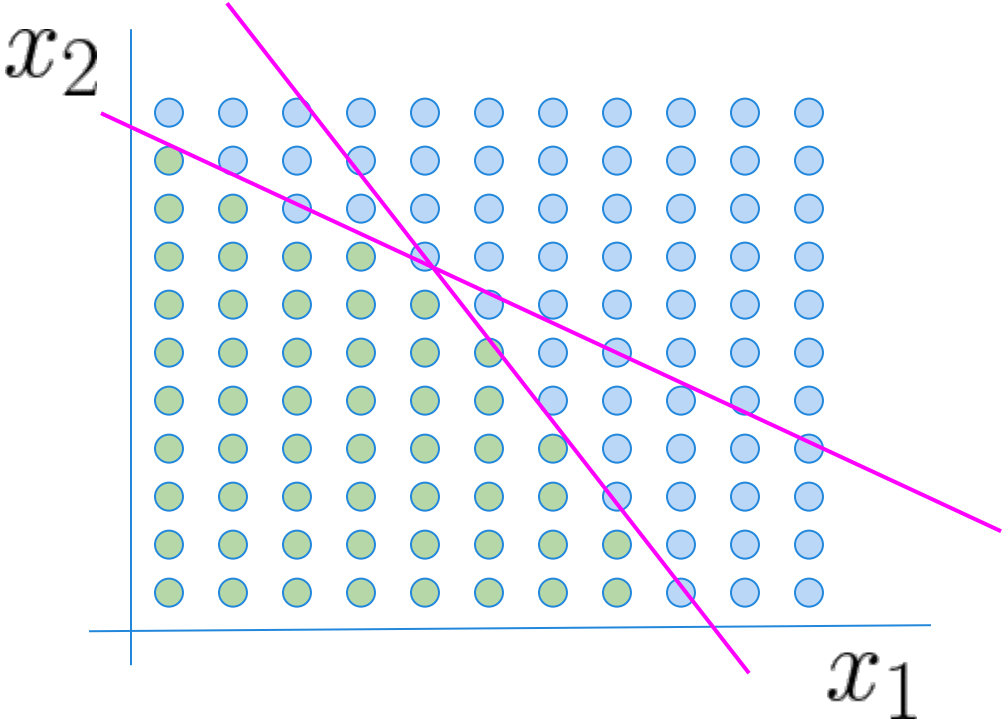


Programacion Entera

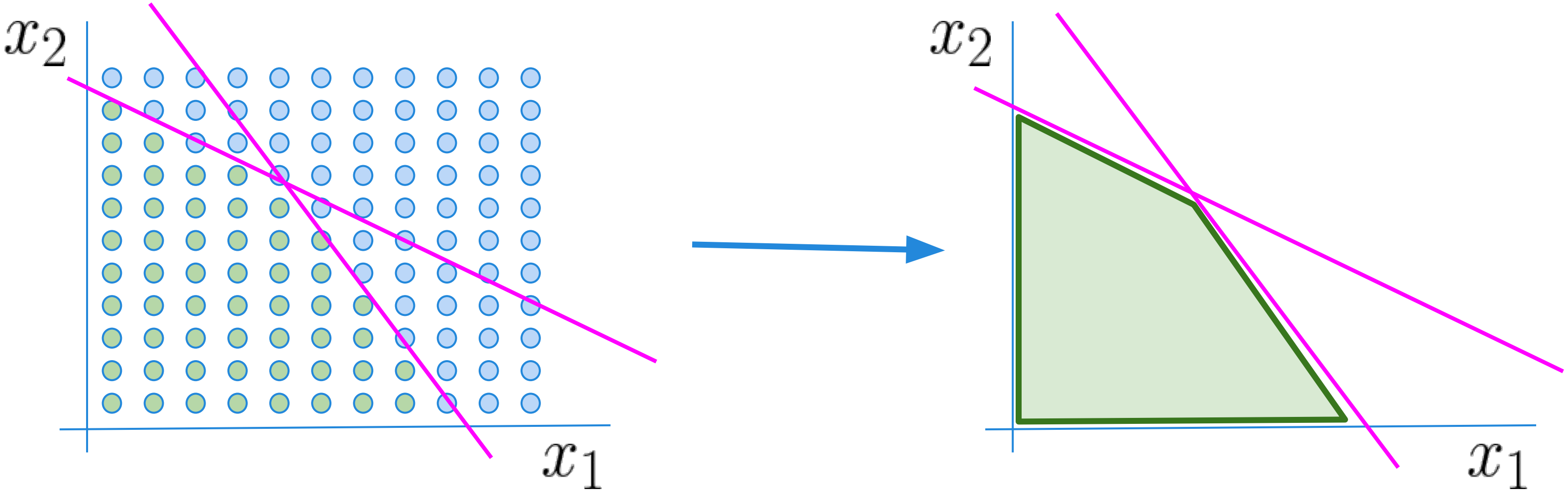
Programacion entera

$$\max_x c^T x$$
$$\text{s.t. } Ax \leq b$$
$$x \geq 0$$
$$x \in \mathbb{Z}^d$$

$$x \in [0, 1, 2, 3, 4, ...,]$$



Programacion entera: relajación via programacion lineal



$$\max_x \; c^T x \quad \text{s.t.} \quad Ax \leq b, \quad x \geq 0, \quad x \in \mathbb{Z}^d$$

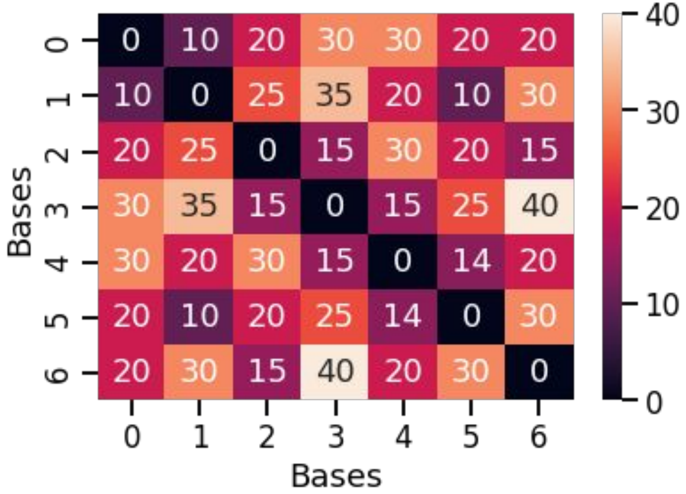
$x \in [0, 1, 2, 3, 4, ...,]$

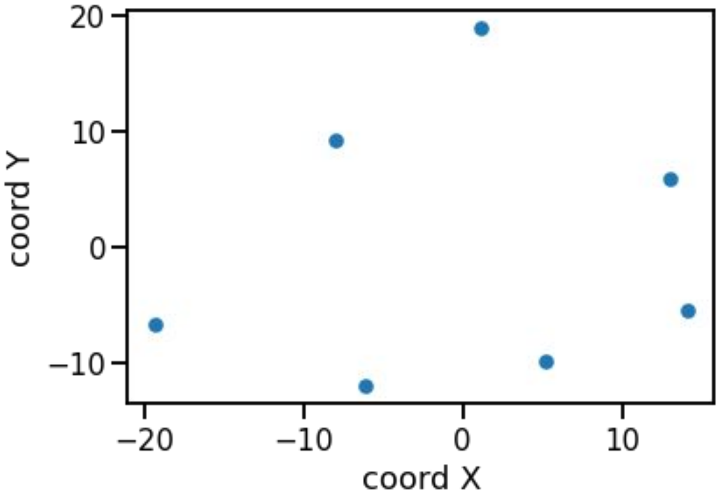
$$\max_x \; c^T x \quad \text{s.t.} \quad Ax \leq b, \quad x \geq 0, \quad x \in \mathbb{R}^d$$

Un problema de programación entera puede relajarse al transformar las variables de decisión enteras a variables de decisión reales continuas. La solución obtenida de P.L. puede redondearse a una solución factible en programacion entera.

Programacion entera: set covering

En un parque nacional existen 7 bases con equipamiento para los guarda bosques. La tabla 1 muestra la distancia en minutos que existe entre cada base del parque nacional. El objetivo es equipar las bases con equipos de drones para poder monitorear posibles focos de incendio. Se quiere adquirir una cantidad minima de equipos que asegure que cada base estará cubierta con un drone hasta 15 minutos de vuelo.





Formular un problema de programación entera que indique cuantos equipos deben adquirirse y en que bases debieran instalarse.

Programacion entera: set covering

$$\min\ x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7$$

$$x \in \{0, 1\}$$

Programacion entera: set covering

	B1	B2	B3	B4	B5	B6	B7
B1	0	10	20	30	30	20	20
B2	10	0	25	35	20	10	30
B3	20	25	0	15	30	20	15
B4	30	35	15	0	15	25	40
B5	30	20	30	15	0	14	20
B6	20	10	20	25	14	0	30
B7	20	30	15	40	20	30	0

$$
\begin{aligned}
x_1 + x_2 \quad & \geq 1 \\
x_1 + x_2 + x_6 & \geq 1 \\
x_3 + x_4 + x_7 & \geq 1 \\
x_3 + x_4 + x_5 & \geq 1 \\
x_4 + x_5 + x_6 & \geq 1 \\
x_2 + x_5 + x_6 & \geq 1 \\
x_3 + x_7 \quad & \geq 1
\end{aligned}
$$

Programacion entera: set covering

$$\min x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7$$

$$x \in \{0, 1\}$$

$$\begin{aligned} x_1 + x_2 \quad &\geq 1 \\ x_1 + x_2 + x_6 &\geq 1 \\ x_3 + x_4 + x_7 &\geq 1 \\ x_3 + x_4 + x_5 &\geq 1 \\ x_4 + x_5 + x_6 &\geq 1 \\ x_2 + x_5 + x_6 &\geq 1 \\ x_3 + x_7 \quad &\geq 1 \end{aligned}$$

Programacion entera: set covering

```python
import pulp
# definimos si es un problema de minimizacion o maximizacion
set_covering = LpProblem("Primal", LpMinimize)

# definimos las variables de decision, el tipo de variable y la cota i
x1 = LpVariable('x1', lowBound=0, cat='Integer')
x2 = LpVariable('x2', lowBound=0, cat='Integer')
x3 = LpVariable('x3', lowBound=0, cat='Integer')
x4 = LpVariable('x4', lowBound=0, cat='Integer')
x5 = LpVariable('x5', lowBound=0, cat='Integer')
x6 = LpVariable('x6', lowBound=0, cat='Integer')
x7 = LpVariable('x7', lowBound=0, cat='Integer')

# primero agregamos la funcion objetivo
set_covering += x1 + x2 + x3 + x4 + x5 + x6 + x7 , "Funcion objetivo"

# luego agregamos restricciones
set_covering += x1 + x2 >= 1 , "Nodo 1"
set_covering += x1 + x2 + x6 >= 1, "Nodo 2"
set_covering += x3 + x4 + x7 >= 1, "Nodo 3"
set_covering += x3 + x4 + x5 >= 1, "Nodo 4"
set_covering += x4 + x5 + x6 >= 1, "Nodo 5"
set_covering += x2 + x5 + x6 >= 1, "Nodo 6"
set_covering += x3 + x7 >= 1, "Nodo 7"

# Resolver el problema con el solver de PULP
set_covering.solve()
```

$$\min x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7$$

$$x \in \{0, 1\} \qquad \begin{aligned} x_1 + x_2 \quad &\geq 1 \\ x_1 + x_2 + x_6 &\geq 1 \\ x_3 + x_4 + x_7 &\geq 1 \\ x_3 + x_4 + x_5 &\geq 1 \\ x_4 + x_5 + x_6 &\geq 1 \\ x_2 + x_5 + x_6 &\geq 1 \\ x_3 + x_7 \quad &\geq 1 \end{aligned}$$

ioperativ_clase21_integerprogramming_setcovering.ipynb

Programacion entera: set covering

```
# valor de la funcion objetivo
print(value(set_covering.objective))
3.0

# valor de las variable de decision en el punto optimo
variables_decision = np.array([[
  set_covering.variables()[0].varValue,
  set_covering.variables()[1].varValue,
  set_covering.variables()[2].varValue,
  set_covering.variables()[3].varValue,
  set_covering.variables()[4].varValue,
  set_covering.variables()[5].varValue,
  set_covering.variables()[6].varValue
  ]])

print(variables_decision)
[[1. 0. 0. 0. 1. 0. 1.]]
```

Programacion entera: set covering

	B1	B2	B3	B4	B5	B6	B7
B1	0	10	20	30	30	20	20
B2	10	0	25	35	20	10	30
B3	20	25	0	15	30	20	15
B4	30	35	15	0	15	25	40
B5	30	20	30	15	0	14	20
B6	20	10	20	25	14	0	30
B7	20	30	15	40	20	30	0

$$x_1 = 1$$
$$x_2 = 0$$
$$x_3 = 0$$
$$x_4 = 0$$
$$x_5 = 1$$
$$x_6 = 0$$
$$x_7 = 1$$