



# Introducción a modelo de Flujo de Mínimo Costo: caso camino más corto

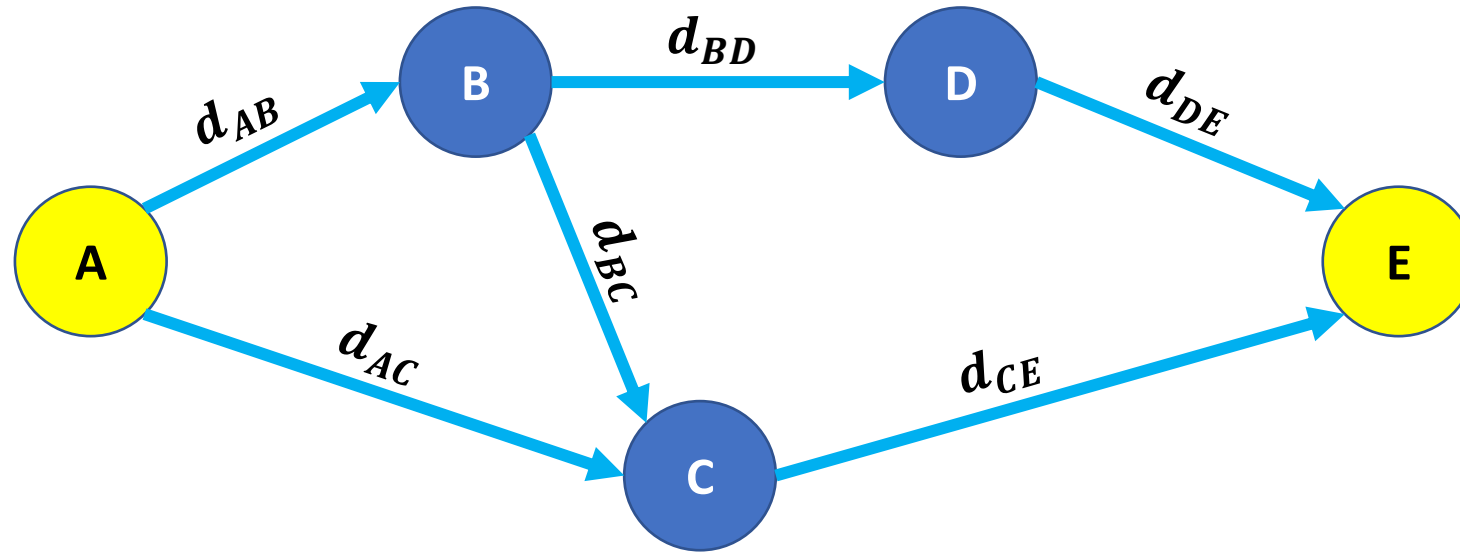
Rodrigo Maranzana

# Camino más corto como FMC

Modelo de programación matemática: **Flujo de Mínimo Costo**

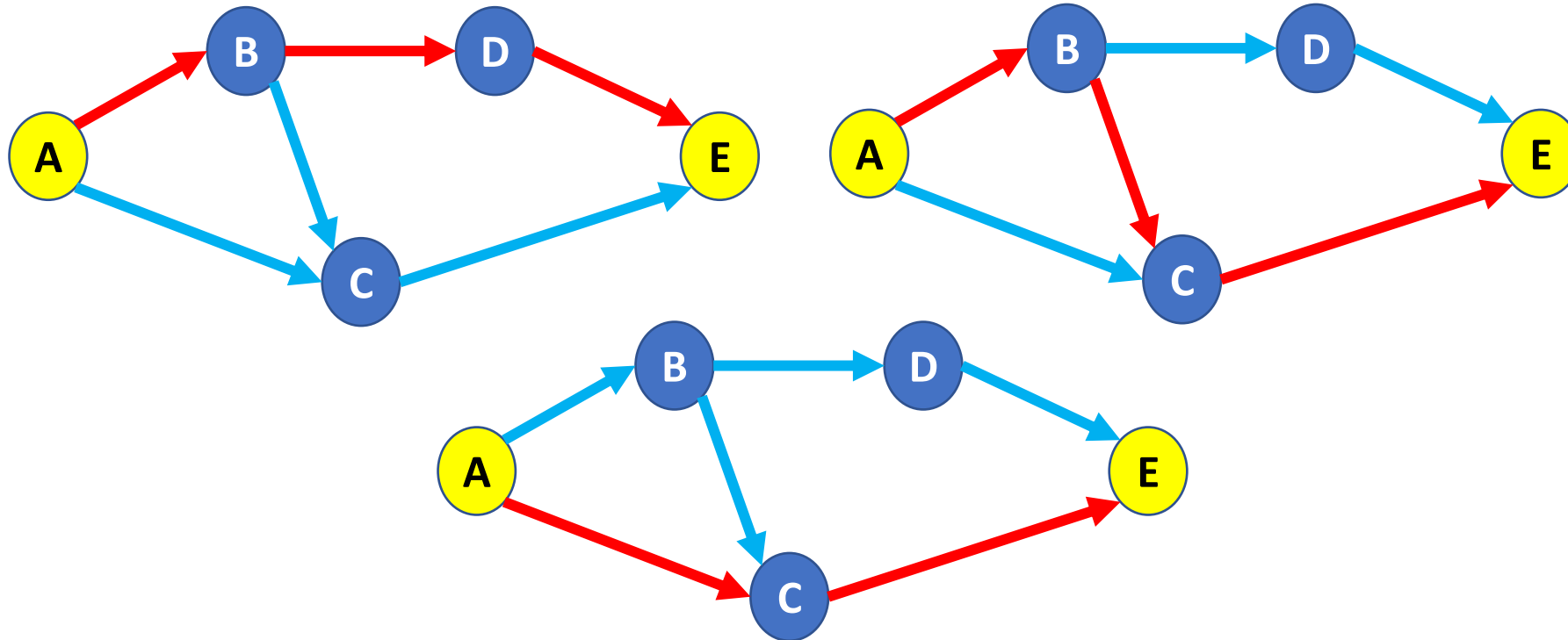
- Resolución simple por algoritmos generalistas.
- Interés matemático teórico.

# Grafo direcccionado

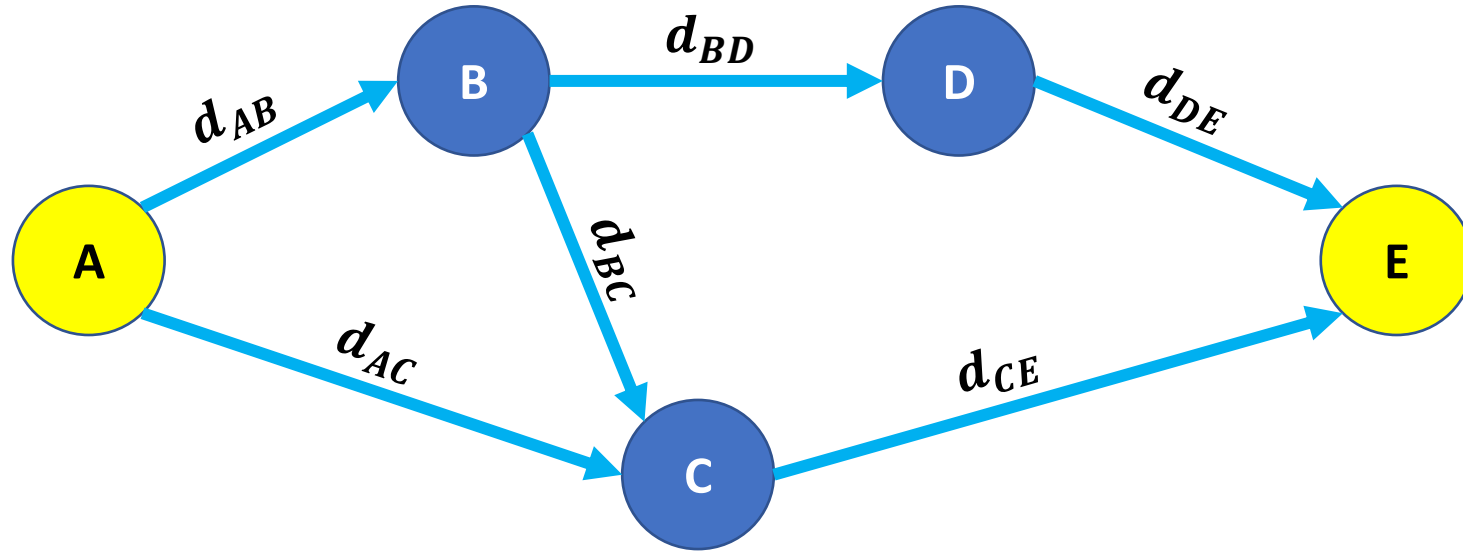


# Camino en grafos direccionados

Camino posibles:



# Caminos en grafos direccionados



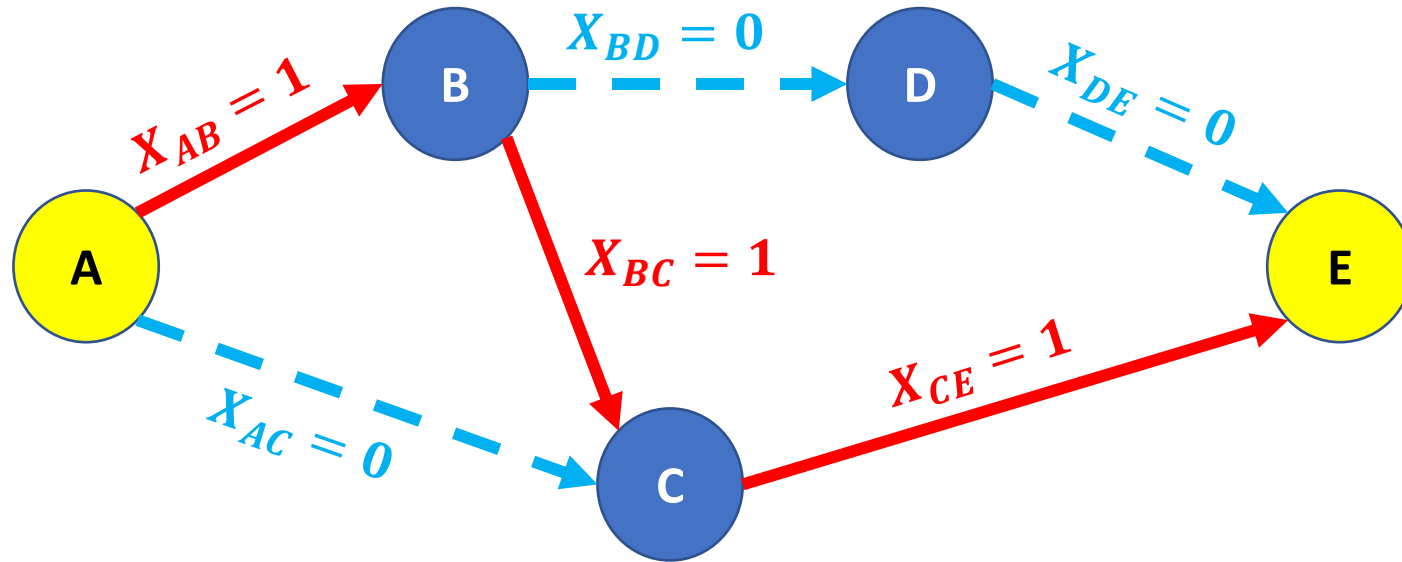
$X_{ij}$  Variable de decisión binaria de ir por camino  $i \rightarrow j$

$X_{ij} = 0$ , no elijo el camino

$X_{ij} = 1$ , elijo el camino

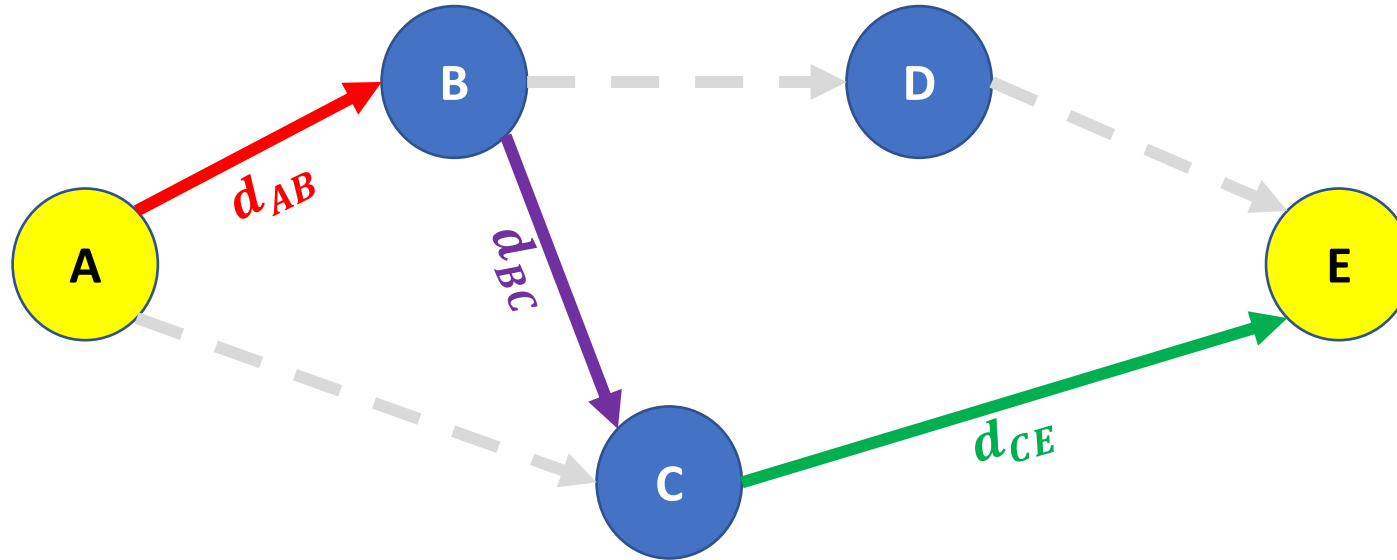
# Caminos en grafos direccionados

Veamos un camino:



# Caminos en grafos direccionados

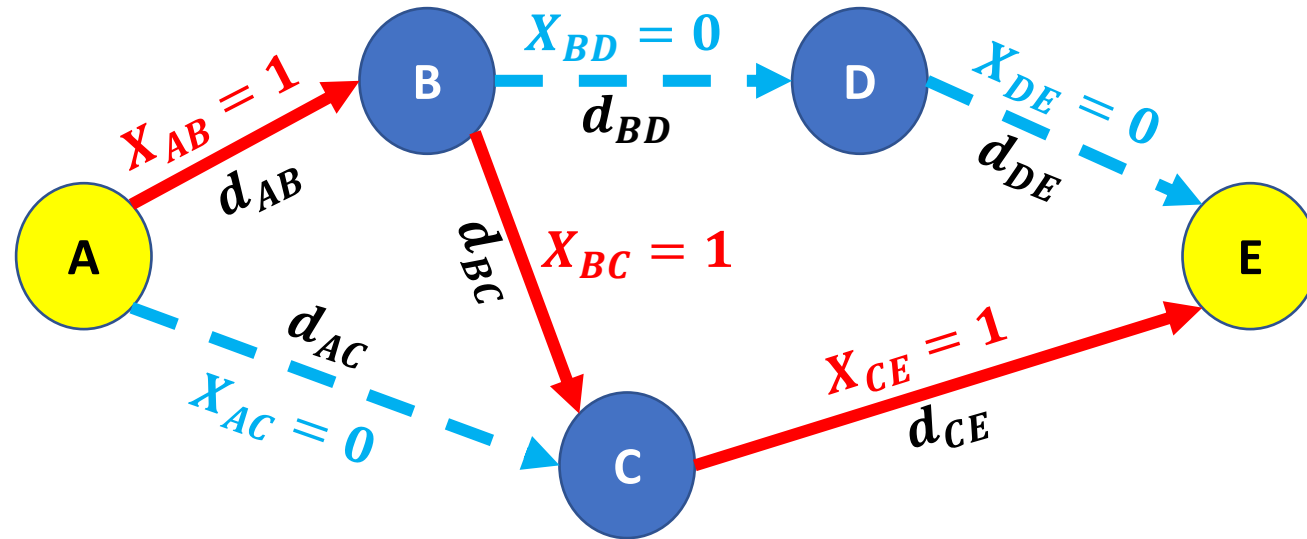
Cálculo de distancia del camino:



$$d_{TOTAL} = d_{AB} + d_{BC} + d_{CE}$$

# Caminos en grafos direccionados

Generalización del cálculo de distancia del camino:



$$d_{TOTAL} = X_{AB} * d_{AB} + X_{BC} * d_{BC} + X_{CE} * d_{CE} + \\ + X_{BD} * d_{BD} + X_{AC} * d_{AC} + X_{DE} * d_{DE}$$



# Caminos en grafos direccionados

Generalización del cálculo de distancia del camino:

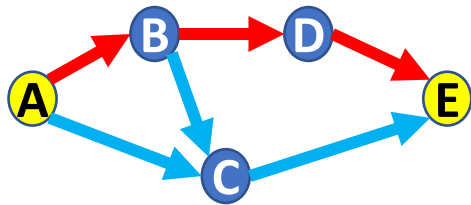
$$d_{TOTAL} = \overset{1}{X_{AB}} * d_{AB} + \overset{1}{X_{BC}} * d_{BC} + \overset{1}{X_{CE}} * d_{CE} + \\ + \underset{0}{X_{BD}} * d_{BD} + \underset{0}{X_{AC}} * d_{AC} + \underset{0}{X_{DE}} * d_{DE}$$

$$d_{TOTAL} = d_{AB} + d_{AB} + d_{CE}$$

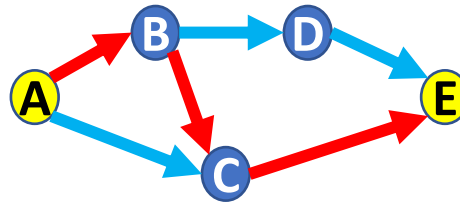
# Camino en grafos direccionados

$$d_{TOTAL} = X_{AB} * d_{AB} + X_{BC} * d_{BC} + X_{CE} * d_{CE} + \\ + X_{BD} * d_{BD} + X_{AC} * d_{AC} + X_{DE} * d_{DE}$$

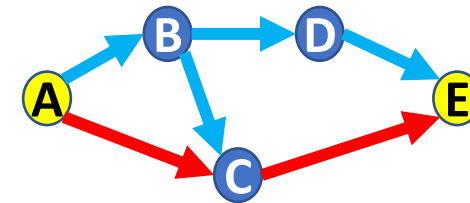
$$X_{AB} = X_{BD} = X_{DE} = 1 \\ X_{AC} = X_{BC} = X_{CE} = 0$$



$$X_{AB} = X_{BC} = X_{CE} = 1 \\ X_{AC} = X_{BD} = X_{DE} = 0$$

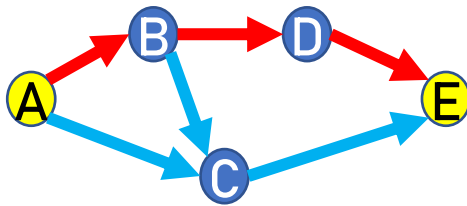


$$X_{AC} = X_{CE} = 1 \\ X_{AB} = X_{BC} = X_{BD} = X_{DE} = 0$$

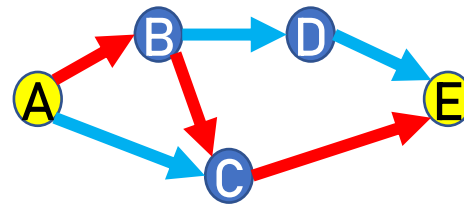


# Camino en grafos direccionados

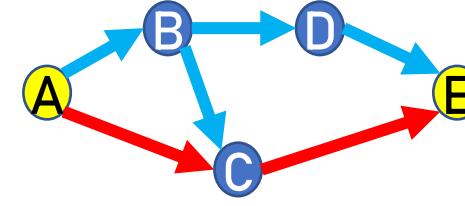
Cálculo del camino más corto:



$d_{TOTAL\ 1}$



$d_{TOTAL\ 2}$



$d_{TOTAL\ 3}$

$$\text{Min} \{d_{TOTAL\ 1}, d_{TOTAL\ 2}, d_{TOTAL\ 3}\}$$

# Función objetivo: camino de mínima distancia

$$d_{TOTAL_k} = X_{AB} * d_{AB} + X_{BC} * d_{BC} + X_{CE} * d_{CE} + \\ + X_{BD} * d_{BD} + X_{AC} * d_{AC} + X_{DE} * d_{DE}$$



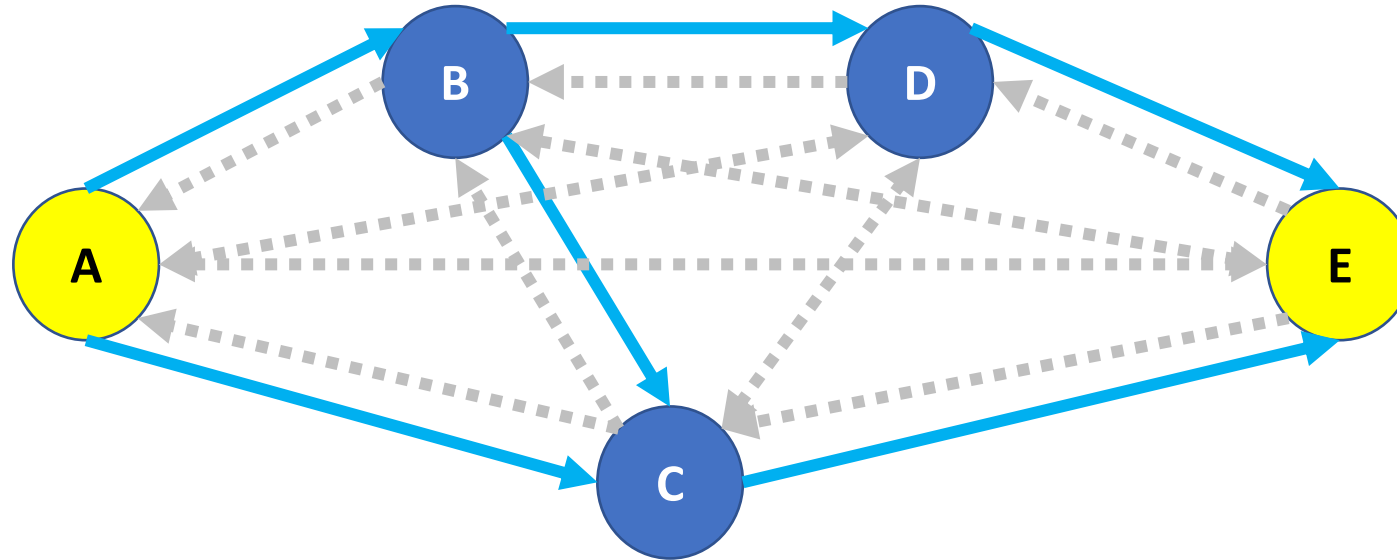
Generalización para encontrar un camino "k"

$$d_{TOTAL_k} = \sum_i \sum_j X_{ij} d_{ij}$$

¿Qué grafo se adapta a esta ecuación?

# Camino en grafos direccionados

Generalización del grafo sin ciclos:

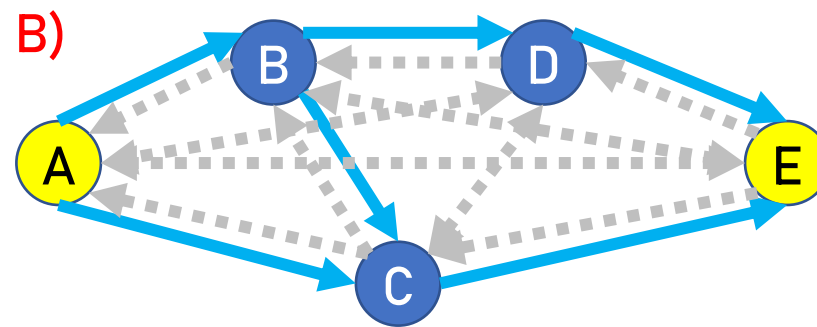
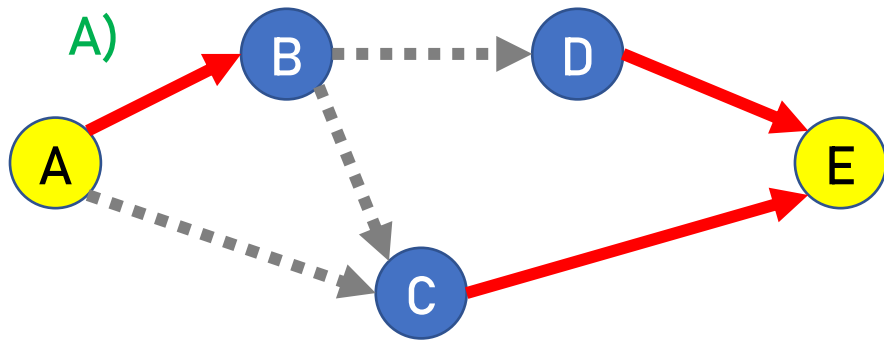


Arcos siempre apagados

# Caminos en grafos direccionados

Problemas de la ecuación anterior:

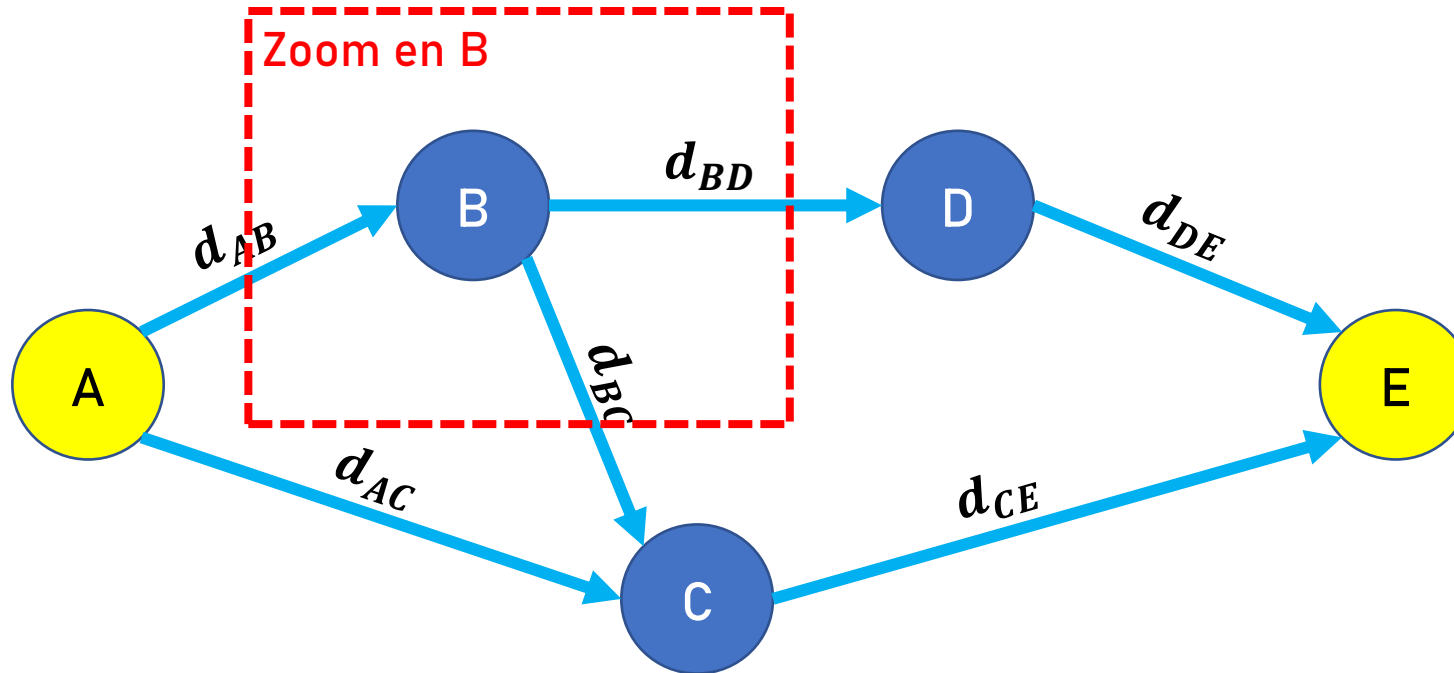
- A) ¿Cómo explico al modelo qué es un camino?
- B) ¿Cómo le digo al modelo qué arcos existen?



La clave está en agregar: **Restricciones**

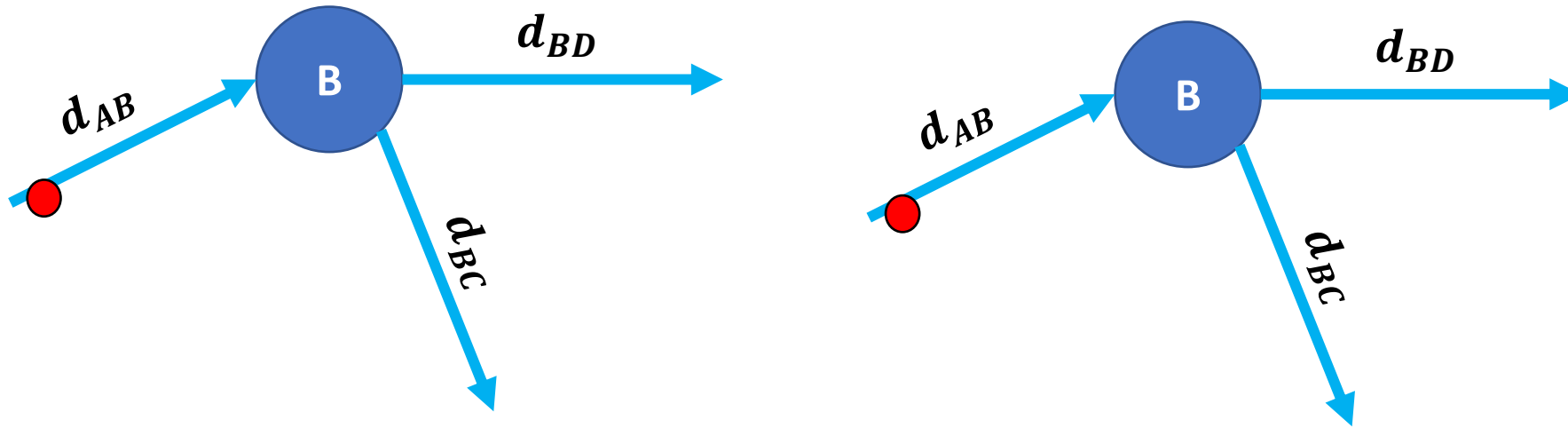
# Flujo de un nodo

Resolvemos problema A



# Flujo de un nodo

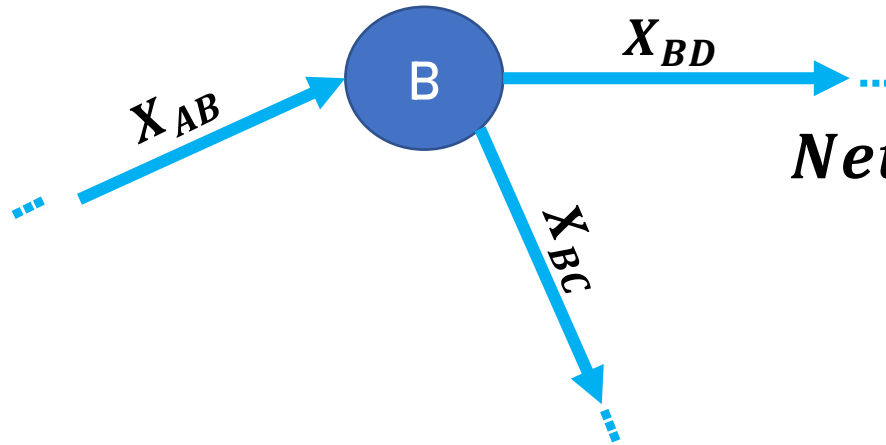
Posibilidades de un agente viajando por el camino:





# Flujo de un nodo

Entradas y salidas de B (1 persona viaja):



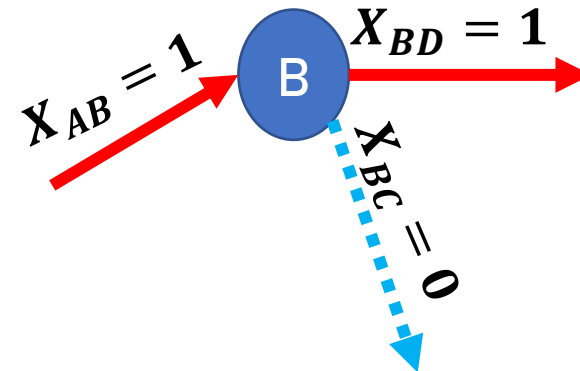
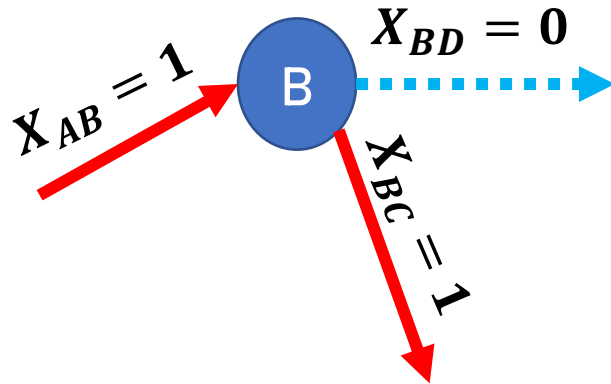
$$Neto_B = Salidas_B - Entradas_B$$

$$0 = Salidas_B - Entradas_B$$

$$0 = X_{BD} + X_{BC} - X_{AB}$$

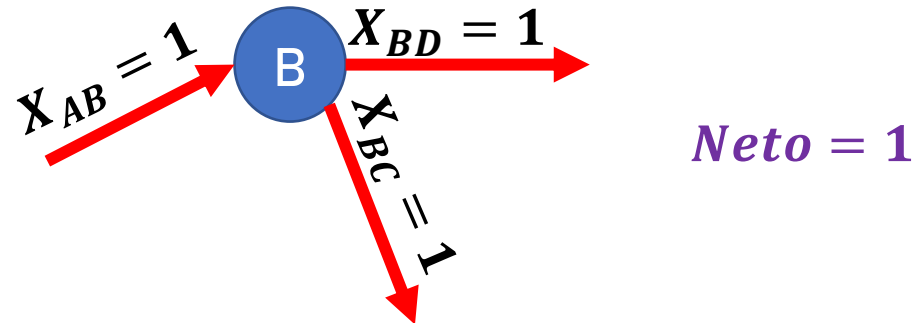
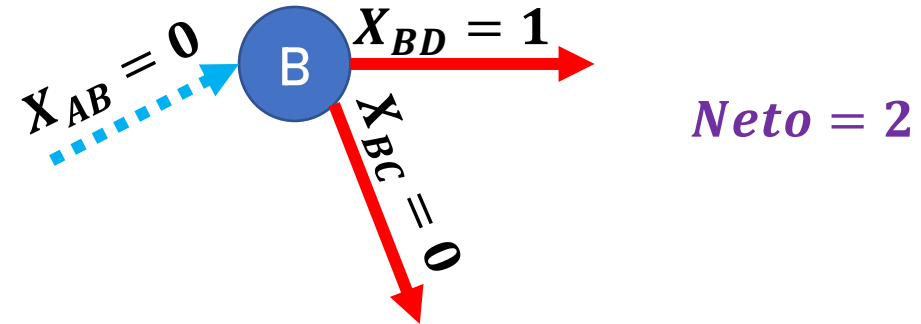
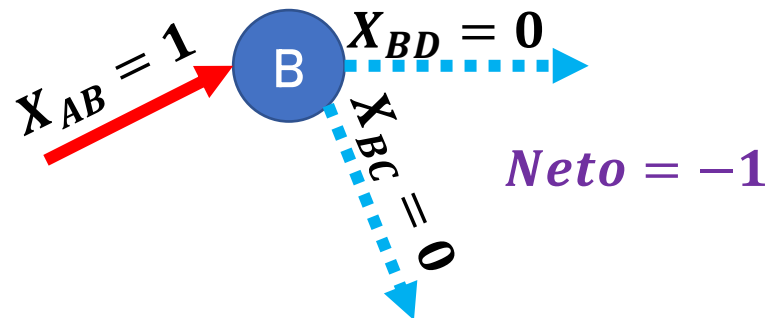
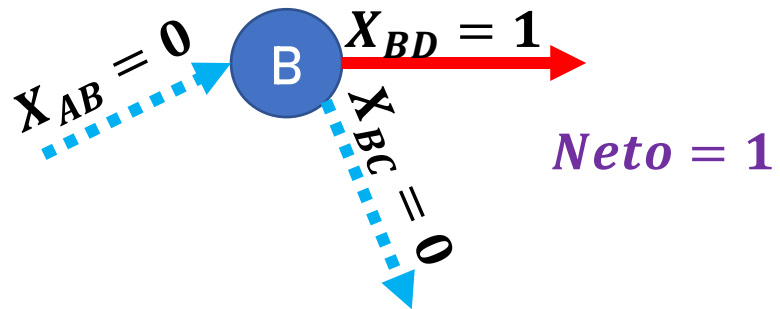
# Flujo de un nodo

$$0 = \textit{Salidas}_B - \textit{Entradas}_B$$



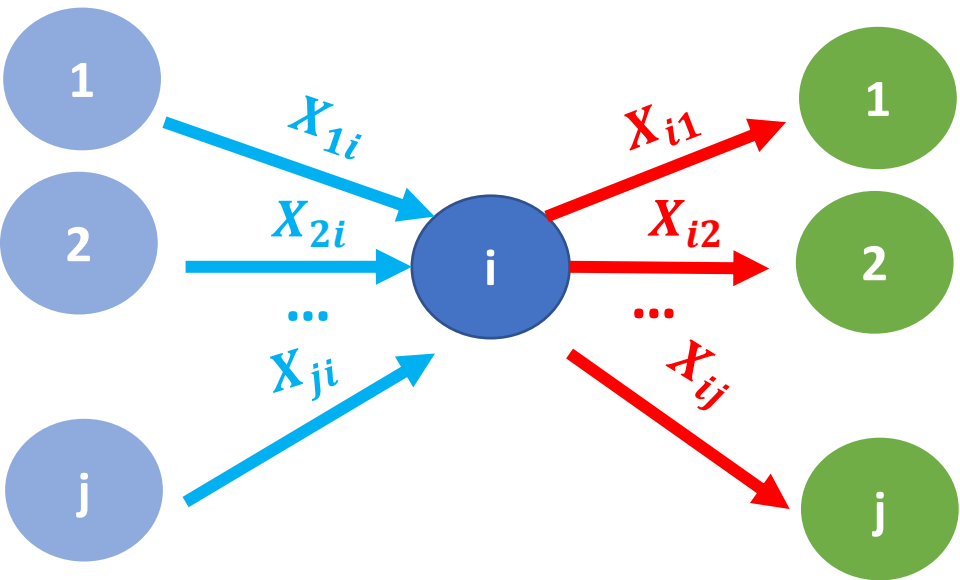
# Flujo de un nodo

$0 \neq \text{Salidas}_B - \text{Entradas}_B$  (no se cumple restricción)



# Flujo de un nodo

Generalizamos la restricción



$$0 = \text{Salidas}_i - \text{Entradas}_i$$

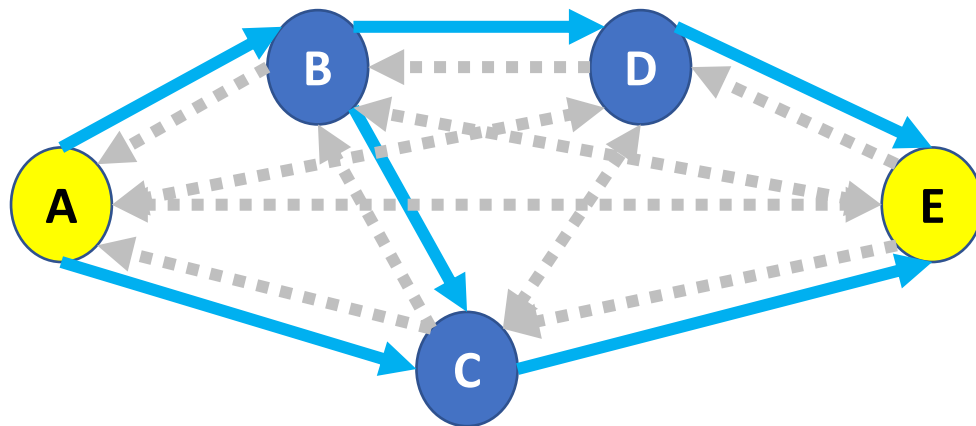
$$0 = \sum_j X_{ij} - \sum_j X_{ji} \quad ; \forall i$$

¡Pero con esto surge el mismo problema B!

¿Cómo le digo qué arcos existen?

# Arquitectura de un grafo: matriz nodo-nodo

Generalización del grafo sin ciclos:

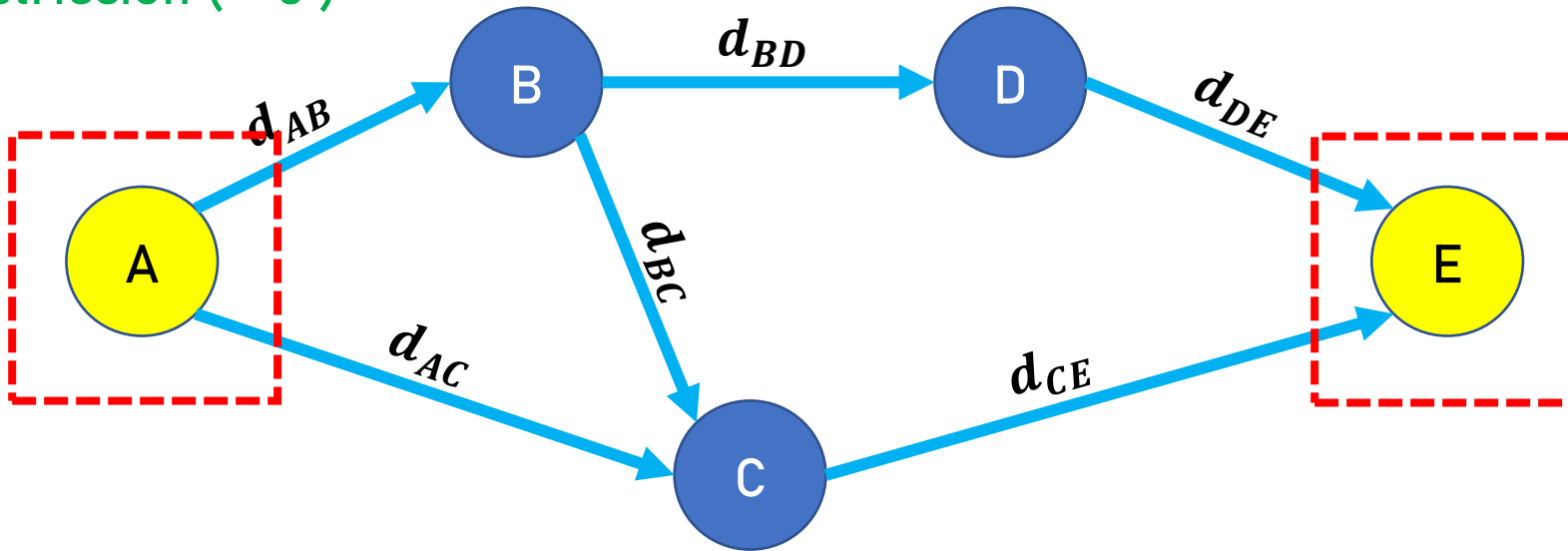

$$A = \begin{matrix} & \begin{matrix} A & B & C & D & E \end{matrix} \\ \begin{matrix} A \\ B \\ C \\ D \\ E \end{matrix} & \begin{bmatrix} 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{matrix}$$

$$0 = \sum_{j, ij \in A} X_{ij} - \sum_{j, ji \in A} X_{ji} \quad ; \quad \forall i$$

# Flujo de nodos extremos

Un problema más:

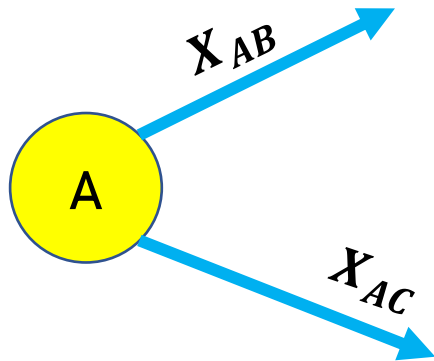
Los nodos A (inicio) y E (final) no tienen forma de cumplir la restricción ( $= 0$ )



# Flujo de nodos extremos

$$1 = \text{Salidas}_A$$

$$1 = X_{AB} + X_{AC}$$

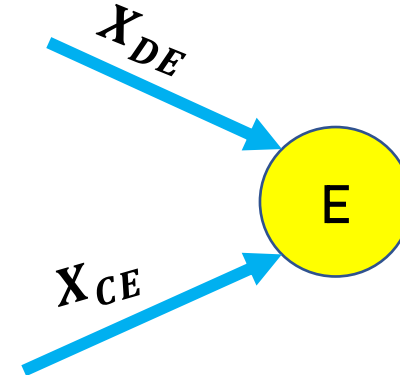


1 = "Sale de A"

Elegir solo un arco

$$-1 = \text{Entradas}_E$$

$$-1 = -X_{DE} - X_{CE}$$



-1 = "Llega a E"

# Restricción de flujo o balance

Generalizamos la restricción

$$b_i = \sum_{j, ij \in A} x_{ij} - \sum_{j, ji \in A} x_{ji} \quad ; \forall i$$

$b_i = 1$  si es inicio

$b_i = -1$  si es final

$b_i = 0$  si es intermedio



# Flujo de Mínimo Costo (FMC) para camino más corto

$$\text{Min} \sum_i \sum_j x_{ij} d_{ij}$$

st:

$$b_i = \sum_{j, ij \in A} x_{ij} - \sum_{j, ji \in A} x_{ji} \quad ; \forall i$$

$b_i = 1$  si es inicio

$b_i = -1$  si es final

$b_i = 0$  si es intermedio

# Forma matricial de FMC: función objetivo

Pensamos las distancias y los arcos como vectores:

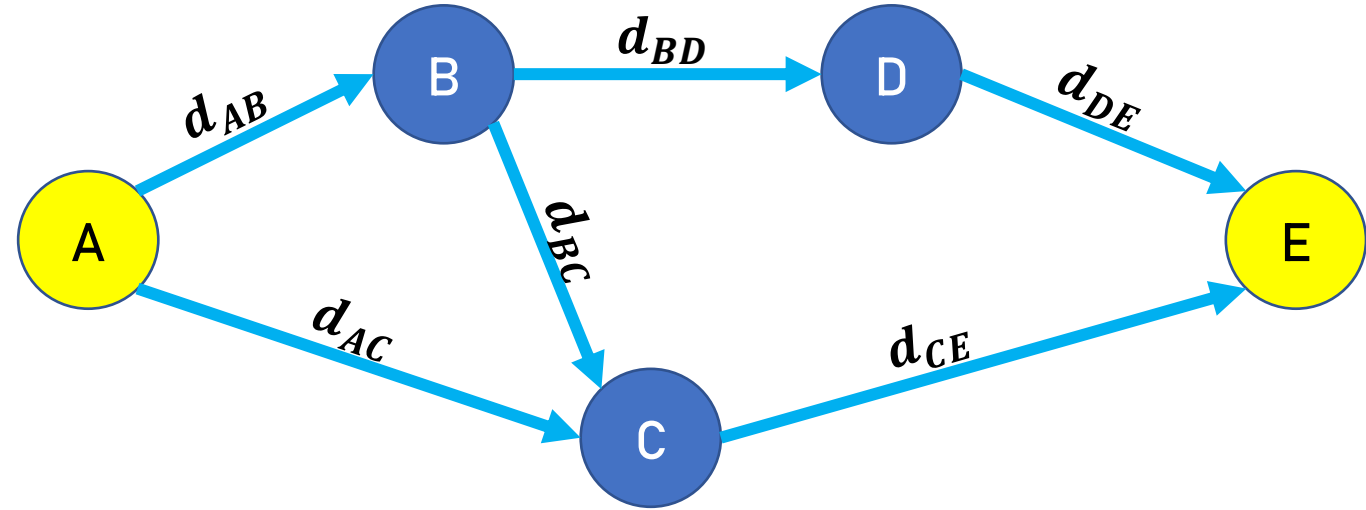
$$C^T = [d_{AB} \quad d_{AC} \quad d_{BD} \quad d_{BC} \quad d_{DE} \quad d_{CE}]$$

$Dim(n^\circ \text{ arcos})$

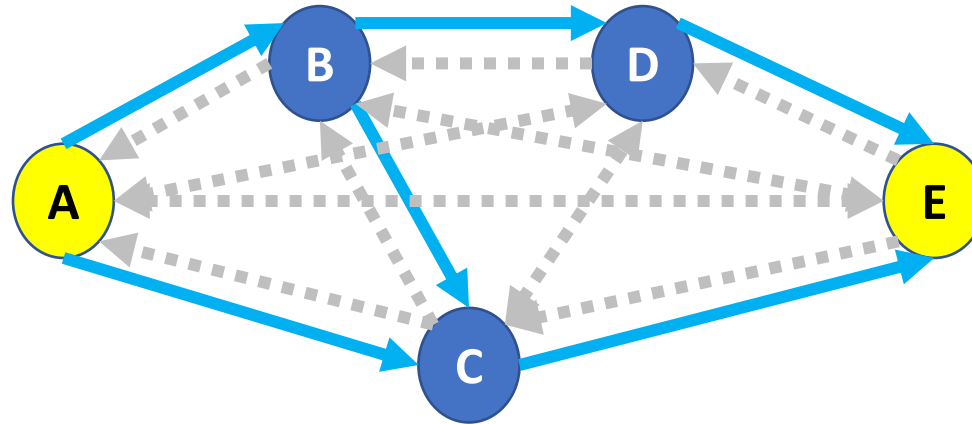
$$X = \begin{bmatrix} x_{AB} \\ x_{AC} \\ x_{BD} \\ x_{BC} \\ x_{DE} \\ x_{CE} \end{bmatrix}$$

$Dim(n^\circ \text{ arcos})$

$$\text{Min} \sum_i \sum_j x_{ij} d_{ij} \rightarrow \boxed{\text{Min } C^T X}$$



# Matriz nodo-nodo a nodo-arco



$$NN =$$

	A	B	C	D	E
A	0	1	1	0	0
B	0	0	1	1	0
C	0	0	0	0	1
D	0	0	0	0	1
E	0	0	0	0	0

Matriz nodo-nodo

$Dim(n^{\circ} \text{ nodos}, n^{\circ} \text{ nodos})$

$$A =$$

	AB	AC	BD	BC	DE	CE
A	1	1	0	0	0	0
B	-1	0	1	1	0	0
C	0	-1	0	-1	0	1
D	0	0	-1	0	1	0
E	0	0	0	0	-1	-1

Matriz nodo-arco

$Dim(n^{\circ} \text{ nodos}, n^{\circ} \text{ arcos})$

# Forma matricial de FMC: restricciones

$$X = \begin{bmatrix} x_{AB} \\ x_{AC} \\ x_{BD} \\ x_{BC} \\ x_{DE} \\ x_{CE} \end{bmatrix}$$

*Dim(nº arcos)*

$$b = \begin{bmatrix} b_A \\ b_B \\ b_C \\ b_D \\ b_E \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ -1 \end{bmatrix}$$

*Dim(nº nodos)*

$b_i = 1$  si es inicio  
 $b_i = -1$  si es final  
 $b_i = 0$  si es intermedio

	<i>AB</i>	<i>AC</i>	<i>BD</i>	<i>BC</i>	<i>DE</i>	<i>CE</i>
<i>A</i>	1	1	0	0	0	0
<i>B</i>	-1	0	1	1	0	0
<i>C</i>	0	-1	0	-1	0	1
<i>D</i>	0	0	-1	0	1	0
<i>E</i>	0	0	0	0	-1	-1

*Dim(nº nodos, nº arcos)*

$$b_i = \sum_{j, ij \in A} x_{ij} - \sum_{j, ji \in A} x_{ji} \quad ; \quad \forall i$$



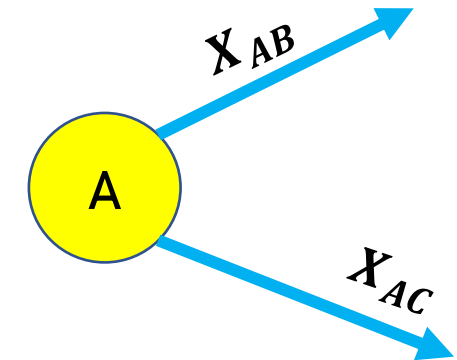
$$AX = b$$

# Funcionamiento de restricción matricial

$$A \quad X = b$$

	<i>AB</i>	<i>AC</i>	<i>BD</i>	<i>BC</i>	<i>DE</i>	<i>CE</i>
<i>A</i>	1	1	0	0	0	0
<i>B</i>	-1	0	1	1	0	0
<i>C</i>	0	-1	0	-1	0	1
<i>D</i>	0	0	-1	0	1	0
<i>E</i>	0	0	0	0	-1	-1

$$\begin{matrix} x_{AB} \\ x_{AC} \\ x_{BD} \\ x_{BC} \\ x_{DE} \\ x_{CE} \end{matrix} = \begin{matrix} 1 \\ 0 \\ 0 \\ 0 \\ -1 \\ -1 \end{matrix}$$



$$\sum_{j, ij \subset A} X_{ij} - \sum_{j, ji \subset A} X_{ji} = b_i \quad ; \quad \forall i$$

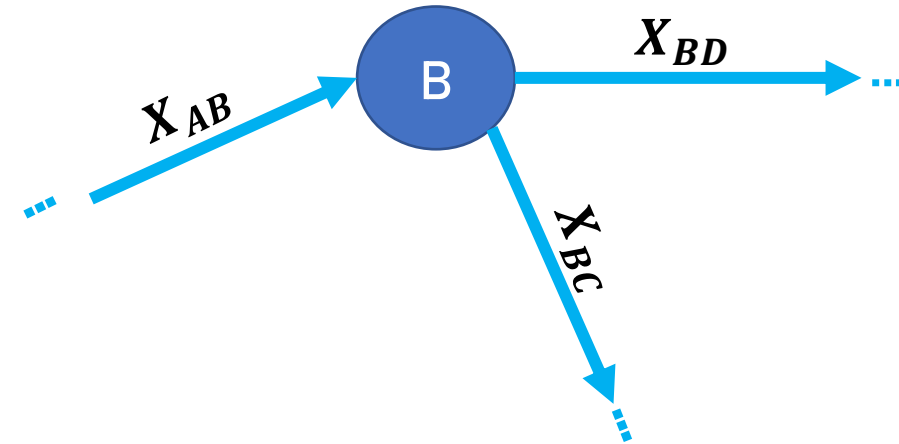
$$X_{AB} + X_{AC} = 1$$

# Funcionamiento de restricción matricial

$$A X = b$$

	<i>AB</i>	<i>AC</i>	<i>BD</i>	<i>BC</i>	<i>DE</i>	<i>CE</i>
<i>A</i>	1	1	0	0	0	0
<i>B</i>	-1	0	1	1	0	0
<i>C</i>	0	-1	0	-1	0	1
<i>D</i>	0	0	-1	0	1	0
<i>E</i>	0	0	0	0	-1	-1

$$\begin{matrix} x_{AB} \\ x_{AC} \\ x_{BD} \\ x_{BC} \\ x_{DE} \\ x_{CE} \end{matrix} = \begin{matrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ -1 \end{matrix}$$



$$\sum_{j, ij \subset A} X_{ij} - \sum_{j, ji \subset A} X_{ji} = b_i \quad ; \quad \forall i$$

$$X_{BD} + X_{BC} - X_{AB} = 0$$

# Flujo de Mínimo Costo (FMC) para camino más corto

$$\text{Min} \sum_i \sum_j X_{ij} d_{ij}$$

st:

$$b_i = \sum_{j, ij \in A} X_{ij} - \sum_{j, ji \in A} X_{ji} \quad ; \quad \forall i$$

$b_i = 1$  si es inicio

$b_i = -1$  si es final

$b_i = 0$  si es intermedio

Condición de positividad:  $X \geq 0$

$$\text{Min} C^T X$$

st:

$$AX = b$$

# Características

En un FMC:

- Función objetivo lineal.
- Restricciones lineales.
- Variables continuas.

El modelo FMC es de programación lineal.



# Características

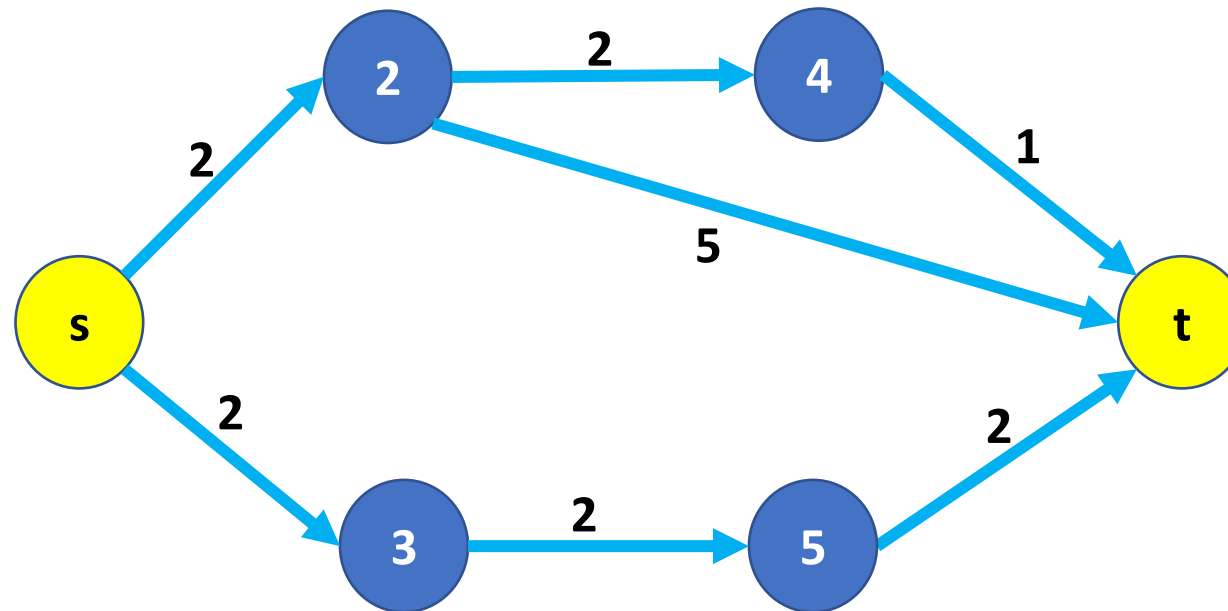
FMC no es únicamente para camino más corto

- Al modificar parámetros en A, b y C, se pueden modelizar muchos casos de optimización de redes:
  - Camino más corto
  - Transporte
  - Transshipment
  - Maximo flujo
  - Asignación
  - Programación de la producción
  - Planificación de proyectos
  - Otros

# Ejemplo

Resolver el camino más corto desde el nodo s al nodo t; respetando la arquitectura del siguiente grafo.

1. *Armar modelo algebraico.*
2. *Armar modelo matricial.*
3. *Resolver ambos modelos en python.*



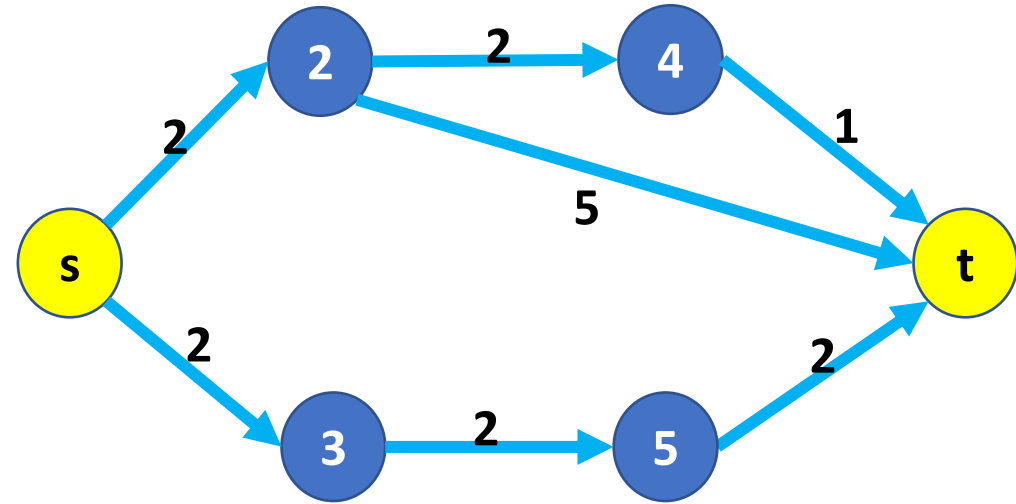
# 1) Modelo FMC matricial

$$\text{Min } C^T X$$

st:

$$AX = b$$

$$X \geq 0$$



Matriz A nodo-arco

	s2	s3	24	2t	35	4t	5t
s	1	1	0	0	0	0	0
2	-1	0	1	1	0	0	0
3	0	-1	0	0	1	0	0
4	0	0	-1	0	0	1	0
5	0	0	0	0	-1	0	1
t	0	0	0	-1	0	-1	-1

Vector b

$$b = \begin{bmatrix} b_s \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_t \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ -1 \end{bmatrix}$$

Vector C

$$C = \begin{bmatrix} c_{s2} \\ c_{s3} \\ c_{24} \\ c_{2t} \\ c_{35} \\ c_{4t} \\ c_{5t} \end{bmatrix} = \begin{bmatrix} 2 \\ 2 \\ 2 \\ 5 \\ 2 \\ 1 \\ 2 \end{bmatrix}$$

$$X = \begin{bmatrix} x_{s2} \\ x_{s3} \\ x_{24} \\ x_{2t} \\ x_{35} \\ x_{4t} \\ x_{5t} \end{bmatrix}$$

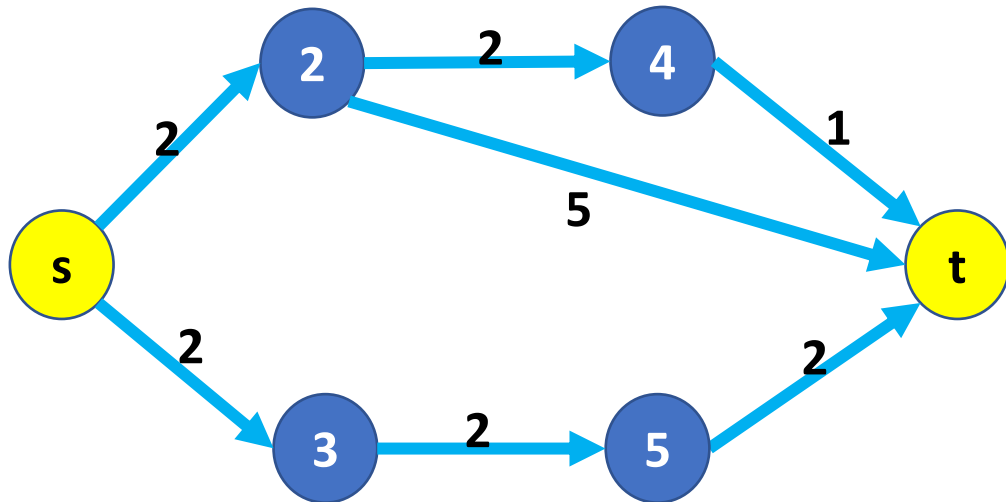
## 2) Modelo FMC algebraico

$$\text{Min} \sum_i \sum_j X_{ij} d_{ij}$$

st:

$$b_i = \sum_{j, ij \in A} X_{ij} - \sum_{j, ji \in A} X_{ji} \quad ; \forall i$$

$$X \geq 0$$



$$\text{Min } 2x_{s2} + 2x_{s3} + 2x_{24} + 5x_{2t} + 2x_{35} + 1x_{4t} + 2x_{5t}$$

st:

**s**  $x_{s2} + x_{s3} = 1$

**2**  $x_{24} + x_{2t} - x_{s2} = 0$

**3**  $x_{35} - x_{s3} = 0$

**4**  $x_{4t} - x_{24} = 0$

**5**  $x_{5t} - x_{35} = 0$

**2**  $-x_{2t} - x_{4t} - x_{5t} = -1$

$$x_{ij} \geq 0$$

# 3) Modelo FMC matricial en python

Usamos la librería numpy para cargar vectores y matrices

Matriz A nodo-arco

	s2	s3	24	2t	35	4t	5t
s	1	1	0	0	0	0	0
2	-1	0	1	1	0	0	0
3	0	-1	0	0	1	0	0
4	0	0	-1	0	0	1	0
5	0	0	0	0	-1	0	1
t	0	0	0	-1	0	-1	-1

```
import numpy

# Matriz nodo-arco.
NN = np.array(
    [[ 1, 1, 0, 0, 0, 0, 0],
     [-1, 0, 1, 1, 0, 0, 0],
     [ 0, -1, 0, 0, 1, 0, 0],
     [ 0, 0, -1, 0, 0, 1, 0],
     [ 0, 0, 0, 0, -1, 0, 1],
     [ 0, 0, 0, -1, 0, -1, -1]]
)
```

# 3) Modelo FMC matricial en python

Usamos la librería numpy para cargar vectores y matrices

Vector b

$$b = \begin{bmatrix} b_s \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_t \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ -1 \end{bmatrix}$$

Vector C

$$C = \begin{bmatrix} c_{s2} \\ c_{s3} \\ c_{24} \\ c_{2t} \\ c_{35} \\ c_{4t} \\ c_{5t} \end{bmatrix} = \begin{bmatrix} 2 \\ 2 \\ 2 \\ 5 \\ 2 \\ 1 \\ 2 \end{bmatrix}$$

```
# Vector de costos o distancias.  
C = np.array([2, 2, 2, 5, 2, 1, 2])  
  
# Vector de términos del lado derecho.  
beq = np.array([1, 0, 0, 0, 0, -1])
```

# 3) Modelo FMC matricial en python

Cargamos las cotas del problema

$$X \geq 0$$

```
# Obtenemos cantidad de arcos:
dim_arcos = A.shape[1]

# Cargamos las cotas  $x \geq 0$  como lista de tuplas, una para cada arco:
cotas = []

for arco_i in range(0, dim_arcos):

    cotas.append((0, np.inf))
```

# 3) Modelo FMC matricial en python

Resolvemos con `scipy.optimize.linprog`

```
# Resolvemos con scipy linprog:  
res = linprog(C, A_eq=A, b_eq=b, bounds=cotas)  
  
# Imprimimos solución:  
print(f'Arcos seleccionados: {res.x}')print(f'Mínima distancia: {res.fun}')
```

$$\begin{array}{ll} \text{Min} & C^T X \\ \text{st:} & \\ & AX = b \\ & X \geq 0 \end{array}$$



# 3) Modelo FMC matricial en python

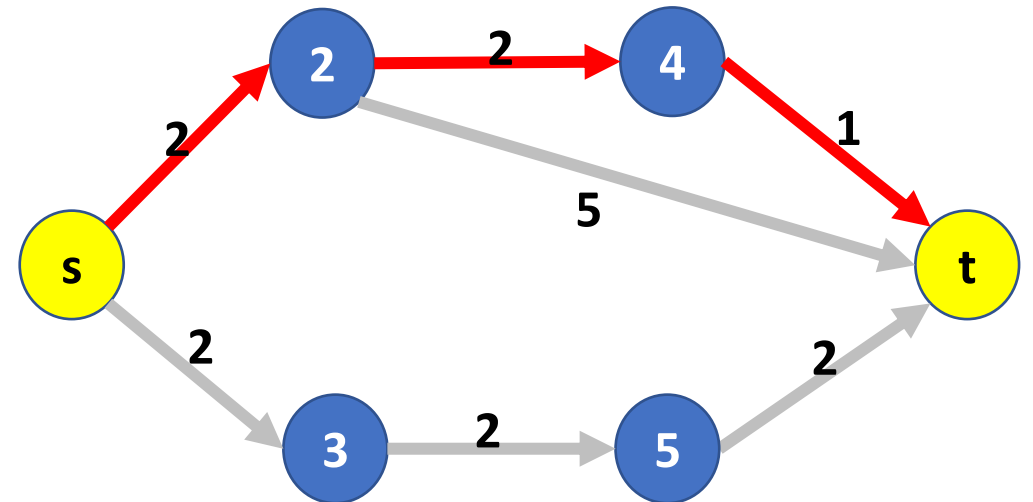
Resolvemos con `scipy.optimize.linprog`

```
# Resolvemos con scipy linprog:
res = linprog(C, A_eq=A, b_eq=b, bounds=cotas)

# Imprimimos solución:
print(f'Arcos seleccionados: {res.x}')
print(f'Mínima distancia: {res.fun}')
```

```
>> Arcos seleccionados: [ 1.  0.  1.  0.  0.  1.  0.]
>> Mínima distancia: 5.0
```

$$X = \begin{bmatrix} x_{s2} \\ x_{s3} \\ x_{24} \\ x_{2t} \\ x_{35} \\ x_{4t} \\ x_{5t} \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$$



# 3) Modelo FMC algebraico en python

## Cargamos el modelo con la librería PuLP

$$\text{Min } 2x_{s2} + 2x_{s3} + 2x_{24} + 5x_{2t} + 2x_{35} + 1x_{4t} + 2x_{5t}$$

st:

$$x_{s2} + x_{s3} = 1$$

$$x_{24} + x_{2t} - x_{s2} = 0$$

$$x_{35} - x_{s3} = 0$$

$$x_{4t} - x_{24} = 0$$

$$x_{5t} - x_{35} = 0$$

$$-x_{2t} - x_{4t} - x_{5t} = -1$$

$$x_{ij} \geq 0$$

```
import pulp

lp01 = pulp.LpProblem("ejercicio-inicial-fmc-shortest-path", pulp.LpMinimize)

# Arcos:
arcos = ['s2', 's3', '24', '2t', '35', '4t', '5t']

# Variables:
X = pulp.LpVariable.dicts('x', arcos, 0, None, cat='Continuous')

# Función objetivo:
lp01 += 2*X['s2'] + 2*X['s3'] + 2*X['24'] + 5*X['2t'] + 2*X['35'] + 1*X['4t'] + 2*X['5t'], "Z"

# # Restricciones:
lp01 += X['s2'] + X['s3'] == 1
lp01 += X['24'] + X['2t'] - X['s2'] == 0
lp01 += X['35'] - X['s3'] == 0
lp01 += X['4t'] - X['24'] == 0
lp01 += X['5t'] - X['35'] == 0
lp01 += -X['2t'] - X['4t'] - X['5t'] == -1
```

# 3) Modelo FMC algebraico en python

## Solución

```
# Resolución:
lp01.solve()

# Imprimimos el status del problema:
print(pulp.LpStatus[lp01.status])

# Imprimimos las variables en su valor óptimo:
for variable in lp01.variables():
    print("%s = %.2f" % (variable.name, variable.varValue))

# Imprimimos el funcional óptimo:
print(f'Función objetivo: {pulp.value(lp01.objective)}')
```

```
>>Optimal
>>x_24 = 1.00
>>x_2t = 0.00
>>x_35 = 0.00
>>x_4t = 1.00
>>x_5t = 0.00
>>x_s2 = 1.00
>>x_s3 = 0.00
>>Función objetivo: 5.0
```

