Programación entera: casos set cover y warehouse location

Rodrigo Maranzana

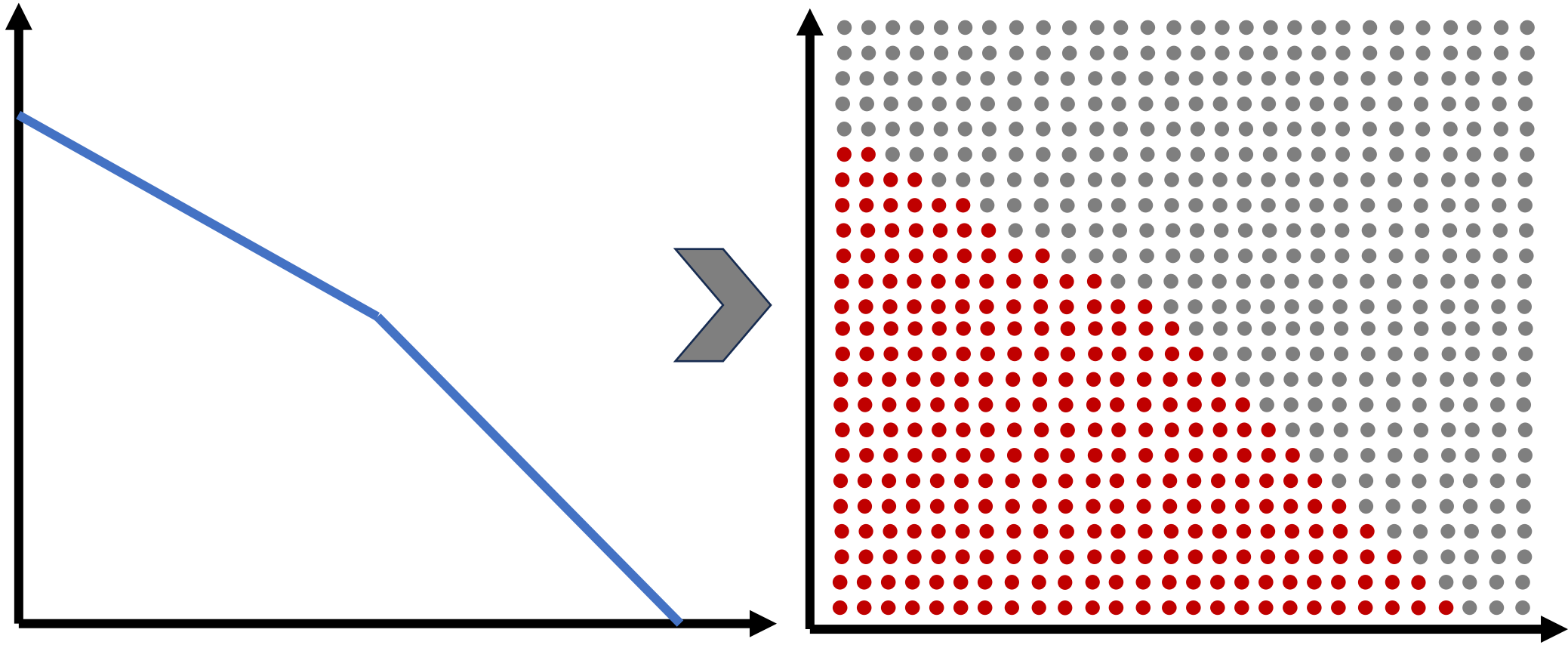
Programación entera

Si las variables de decisión de un modelo son enteras, estamos ante un problema de programación entera.

$$Min\ C^T X$$

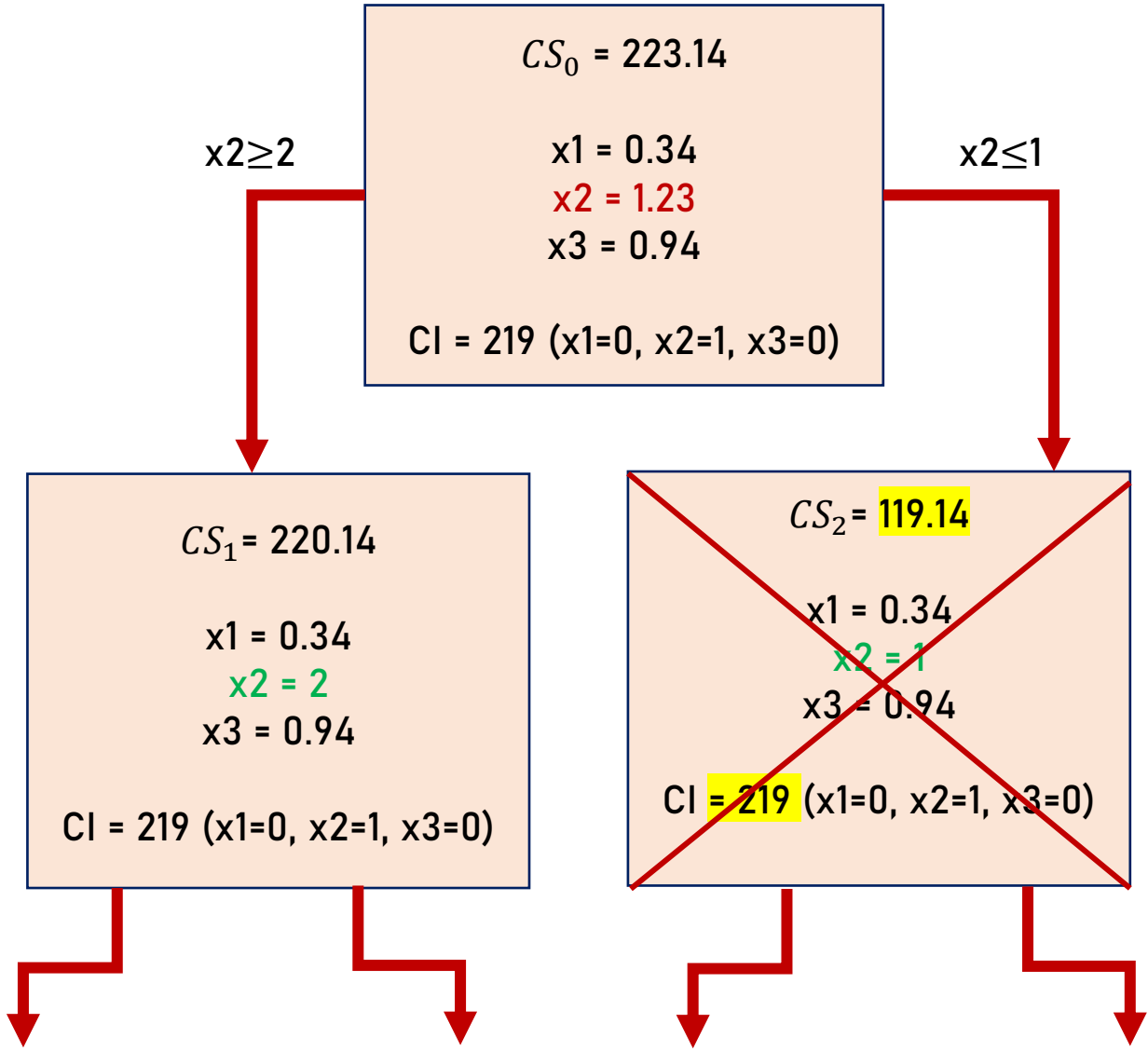
st:

$$AX = b$$
$$X \geq 0$$
$$X \epsilon \mathbb{Z}^d$$



Solvers: Branch & Bound en minimización

1. Resuelve el problema relajado como LP, se guarda resultado Z relajado (cota superior CS).
2. Guarda la cota inferior (CI) del problema, forzando el piso de todas las variables.
3. Selecciona una variable no entera y hace "branching": Parte en 2 ramas: piso (≤) y techo (≥).
4. Agrega restricciones de branching y resuelve ambos problemas como LP.
5. Chequea si optimiza el mejor resultado hasta el momento, nuevos CS.
6. Chequea si las ramas están por encima de la CI, sí es así la actualizan; sino descartan la rama.
7. Actualiza CI si la solución tiene todas variables enteras.
8. Loop



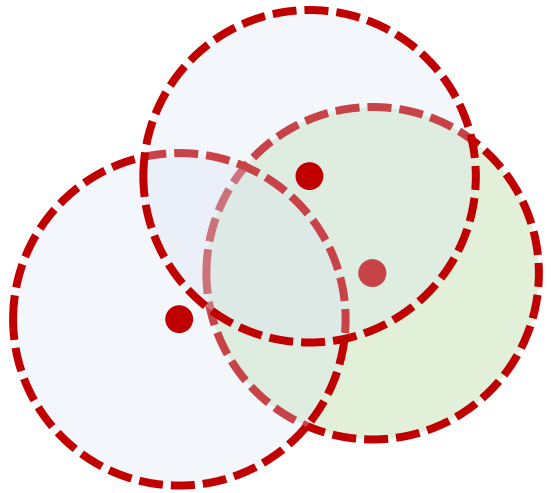
Problema de set cover

En un problema de set cover intentamos seleccionar una cantidad de locaciones o agentes determinados que optimicen una función objetivo.

Los agentes se relacionan a un radio de acción, pueden cubrir una demanda geográfica.

La selección de los agentes, implica prender o apagar variables de decisión, es decir, decidir cubrir o no utilizar el radio de cobertura del agente.

Por lo tanto, estamos ante un problema entero binario.



Problema de set cover

En un recorrido de medio maratón Buenos Aires 21k, existen 12 estaciones de atención básica a los corredores.

Si bien cada una está equipada con el material de auxilio básico; se busca distribuir entre ellas la suficiente cantidad de unidades de Desfibrilador Externo Automático (DEA) de forma que todas las estaciones estén cubiertas dentro de los 4 minutos.

Armar un problema de programación entera que indique la cantidad de equipos DEA a adquirir.

https://maratondebuenosaires.com/

**Este ejemplo es solo didáctico. La optimización de este tipo de elementos requiere mayor nivel de detalle, tolerancias cortas y ajuste a normativas.*

Datos: tiempo entre estaciones

i/j	1	2	3	4	5	6	7	8	9	10	11	12
1	0	4	3	10	15	20	2	6	9	11	8	2
2		0	14	12	6	4	11	9	2	3	10	15
3			0	12	4	5	8	23	2	2	2	6
4				0	6	4	11	9	11	8	6	4
5					0	21	10	3	6	25	10	4
6						0	8	6	10	24	6	8
7							0	5	26	18	10	3
8								0	2	3	5	19
9									0	20	4	13
10										0	10	15
11											0	22
12												0

*Consideramos problema simétrico.

Variables y función objetivo

Variables:

$x_i \in \{0,1\}$ representa colocar ($x_i = 1$) o no ($x_i = 0$) un equipo en la estación i.

Función objetivo:

Minimizar la cantidad de equipos a colocar.

$$Min\ x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7 + x_8 + x_9 + x_{10} + x_{11} + x_{12}$$

Restricciones: estación 1

No superar los 4 minutos entre un DEA y una estación.

Para la estación 1, por lo menos una de las estaciones 1, 2, 3, 7 o 12 tienen que tener un DEA.

$$x_1 + x_2 + x_3 + x_7 + x_{12} \geq 1$$

i/j	1	2	3	4	5	6	7	8	9	10	11	12
1	0	4	3	10	15	20	2	6	9	11	8	2

Restricciones

i/j	1	2	3	4	5	6	7	8	9	10	11	12
1	0	4	3	10	15	20	2	6	9	11	8	2
2	4	0	14	12	6	4	11	9	2	3	10	15
3	3	14	0	12	4	5	8	23	2	2	2	6
4	10	12	12	0	6	4	11	9	11	8	6	4
5	15	6	4	6	0	21	10	3	6	25	10	4
6	20	4	5	4	21	0	8	6	10	24	6	8
7	2	11	8	11	10	8	0	5	26	18	10	3
8	6	9	23	9	3	6	5	0	2	3	5	19
9	9	2	2	11	6	10	26	2	0	20	4	13
10	11	3	2	8	25	24	18	3	20	0	10	15
11	8	10	2	6	10	6	10	5	4	10	0	22
12	2	15	6	4	4	8	3	19	13	15	22	0

Estación 1: $x_1 + x_2 + x_3 + x_7 + x_{12} \geq 1$

Estación 2: $x_1 + x_2 + x_6 + x_9 + x_{10} \geq 1$

Estación 3: $x_1 + x_3 + x_5 + x_9 + x_{10} + x_{11} \geq 1$

Estación 4: $x_4 + x_6 + x_{12} \geq 1$

Estación 5: $x_3 + x_5 + x_8 + x_{12} \geq 1$

Estación 6: $x_2 + x_4 + x_6 \geq 1$

Estación 7: $x_1 + x_7 + x_{12} \geq 1$

Estación 8: $x_5 + x_8 + x_9 + x_{10} \geq 1$

Estación 9: $x_2 + x_3 + x_8 + x_9 + x_{11} \geq 1$

Estación 10: $x_2 + x_3 + x_8 + x_{10} \geq 1$

Estación 11: $x_3 + x_9 + x_{11} \geq 1$

Estación 12: $x_1 + x_4 + x_5 + x_7 + x_{12} \geq 1$

Modelo de optimización

$$Min\ x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7 + x_8 + x_9 + x_{10} + x_{11} + x_{12}$$

s.t.

Estación 1: $x_1 + x_2 + x_3 + x_7 + x_{12} \geq 1$

Estación 2: $x_1 + x_2 + x_6 + x_9 + x_{10} \geq 1$

Estación 3: $x_1 + x_3 + x_5 + x_9 + x_{10} + x_{11} \geq 1$

Estación 4: $x_4 + x_6 + x_{12} \geq 1$

Estación 5: $x_3 + x_5 + x_8 + x_{12} \geq 1$

Estación 6: $x_2 + x_4 + x_6 \geq 1$

Estación 7: $x_1 + x_7 + x_{12} \geq 1$

Estación 8: $x_5 + x_8 + x_9 + x_{10} \geq 1$

Estación 9: $x_2 + x_3 + x_8 + x_9 + x_{11} \geq 1$

Estación 10: $x_2 + x_3 + x_8 + x_{10} \geq 1$

Estación 11: $x_3 + x_9 + x_{11} \geq 1$

Estación 12: $x_1 + x_4 + x_5 + x_7 + x_{12} \geq 1$

$$x_i \epsilon \{0, 1\}$$

Resolución en Python

$Min\ x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7 + x_8 + x_9 + x_{10} + x_{11} + x_{12}$

s.t.

Estación 1: $x_1 + x_2 + x_3 + x_7 + x_{12} \geq 1$
Estación 2: $x_1 + x_2 + x_6 + x_9 + x_{10} \geq 1$
Estación 3: $x_1 + x_3 + x_5 + x_9 + x_{10} + x_{11} \geq 1$
Estación 4: $x_4 + x_6 + x_{12} \geq 1$
Estación 5: $x_3 + x_5 + x_8 + x_{12} \geq 1$
Estación 6: $x_2 + x_4 + x_6 \geq 1$
Estación 7: $x_1 + x_7 + x_{12} \geq 1$
Estación 8: $x_5 + x_8 + x_9 + x_{10} \geq 1$
Estación 9: $x_2 + x_3 + x_8 + x_9 + x_{11} \geq 1$
Estación 10: $x_2 + x_3 + x_8 + x_{10} \geq 1$
Estación 11: $x_3 + x_9 + x_{11} \geq 1$
Estación 12: $x_1 + x_4 + x_5 + x_7 + x_{12} \geq 1$

$x_i \epsilon \{0, 1\}$

```python
lp01 = pulp.LpProblem("set-cover", pulp.LpMinimize)

# Sets:
estaciones = range(1, 13)

# Variables:
x = pulp.LpVariable.dicts('x', estaciones, 0, None, cat='Binary')

# Función objetivo:
lp01 += pulp.lpSum(x), 'Z'

# Restricciones:
lp01 += x[1] + x[2] + x[3] + x[7] + x[12] ⩾ 1
lp01 += x[1] + x[2] + x[6] + x[9] + x[10] ⩾ 1
lp01 += x[1] + x[3] + x[5] + x[9] + x[10] + x[11] ⩾ 1
lp01 += x[4] + x[6] + x[12] ⩾ 1
lp01 += x[3] + x[5] + x[8] + x[12] ⩾ 1
lp01 += x[2] + x[4] + x[6] ⩾ 1
lp01 += x[1] + x[7] + x[12] ⩾ 1
lp01 += x[5] + x[8] + x[9] + x[10] ⩾ 1
lp01 += x[2] + x[3] + x[8] + x[9] + x[11] ⩾ 1
lp01 += x[2] + x[3] + x[8] + x[10] ⩾ 1
lp01 += x[3] + x[9] + x[11] ⩾ 1
lp01 += x[1] + x[4] + x[5] + x[7] + x[12] ⩾ 1

# Resolucion:
lp01.solve()
```

Resolución en Python

```
>> Optimal
>>
>> x_1 = 0
>> x_10 = 0
>> x_11 = 0
>> x_12 = 1
>> x_2 = 1
>> x_3 = 0
>> x_4 = 0
>> x_5 = 0
>> x_6 = 0
>> x_7 = 0
>> x_8 = 0
>> x_9 = 1
>>
>> Función objetivo: 3
```

i/j	1	2	3	4	5	6	7	8	9	10	11	12
1	0	4	3	10	15	20	2	6	9	11	8	2
2	4	0	14	12	6	4	11	9	2	3	10	15
3	3	14	0	12	4	5	8	23	2	2	2	6
4	10	12	12	0	6	4	11	9	11	8	6	4
5	15	6	4	6	0	21	10	3	6	25	10	4
6	20	4	5	4	21	0	8	6	10	24	6	8
7	2	11	8	11	10	8	0	5	26	18	10	3
8	6	9	23	9	3	6	5	0	2	3	5	19
9	9	2	2	11	6	10	26	2	0	20	4	13
10	11	3	2	8	25	24	18	3	20	0	10	15
11	8	10	2	6	10	6	10	5	4	10	0	22
12	2	15	6	4	4	8	3	19	13	15	22	0

Se seleccionan 3 de las 12 estaciones: {2, 9, 12} para ser asignadas con dispositivos DEA.

Relajación del problema

$$x_i \epsilon \{0, 1\} \longrightarrow \begin{array}{l} x_i \epsilon \mathbb{R} \\ x_i \geq 0 \end{array}$$

Se transforma en un problema de programación lineal

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

Relajación del problema

```
>> Optimal
>>
>> x_1 = 0.00
>> x_10 = 0.00
>> x_11 = 0.00
>> x_12 = 1.00
>> x_2 = 1.00
>> x_3 = 0.00
>> x_4 = 0.00
>> x_5 = 0.00
>> x_6 = 0.00
>> x_7 = 0.00
>> x_8 = 0.00
>> x_9 = 1.00
>>
>> Función objetivo: 3.0
```

El problema arroja el mismo resultado.

Pero no todos pueden relajarse. A veces, el solver asigna valores a las variables que son imposibles para el problema real.

Ej: $x_9 = 0.82$

Introducción a restricciones condicionales

Un modelo de programación matemática no puede leerse secuencialmente en sus restricciones.

No admite control de flujo convencional, ya que no hay causalidad entre las restricciones. Ej: activar restricción “i” si “j”

Para resolver estas situaciones existen las restricciones condicionales.

Introducción a restricciones condicionales

Siendo M un número muy grande, y una variable binaria y x real.

Podemos escribir una restricción condicional como:

$$x \leq My$$

Esto se lee como: "si x entonces y"

De forma negativa:

$$x \leq M(1 - y)$$

Esto se lee como: "si x entonces no y"

Introducción a restricciones condicionales

Ejemplo:

“Se decide comprar un robot (decisión y) siempre que haya cantidad producida (decisión x)”

$$x \leq My$$

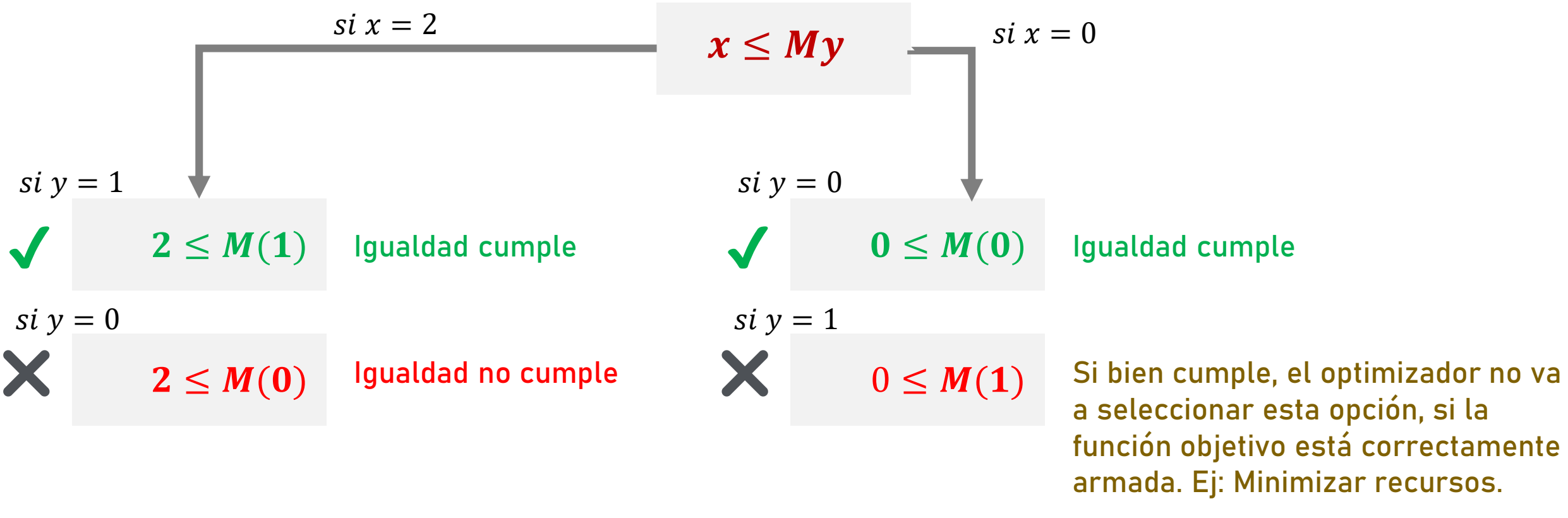
“Se decide no tercerizar (decisión y) siempre que haya cantidad producida (decisión x)”

$$x \leq M(1 - y)$$

Introducción a restricciones condicionales

Siendo M un número muy grande, y una variable binaria y x real.

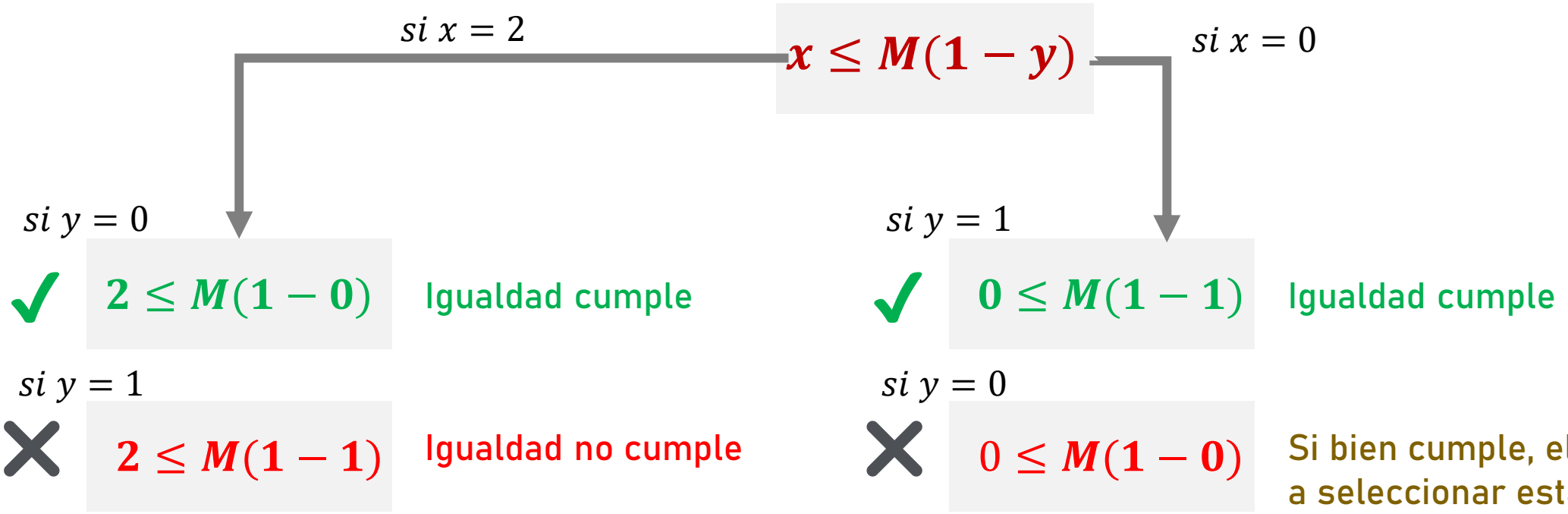
Podemos escribir una restricción condicional como:



Introducción a restricciones condicionales

Siendo M un número muy grande, y una variable binaria y x real.

Podemos escribir una restricción condicional como:



Problema de warehouse location

Existen “i” ubicaciones donde se puede invertir en un centro de distribución. Cada ubicación puede proveer de demanda a determinados clientes “j”.

Existe un costo c_{ij} de transportar mercadería de un centro i a un cliente j.

Se debe decidir las ubicaciones de los centros de distribución, sabiendo que cada centro requiere un costo fijo de operación k_i.

Además, se pide calcular el flujo de productos desde los centros a los clientes.

La optimización implica un costo variable y un costo fijo.

Problema de warehouse location

Tabla de costos

origen / destino	Destino 1	Destino 2	Destino 3	Destino 4
Centro 1	434	523	640	850
Centro 2	323	480	670	770
Centro 3	997	680	390	590

origen / destino	Costo operativo
Centro 1	55.000
Centro 2	45.000
Centro 3	48.000

Tabla de cantidades a enviar

	Destino 1	Destino 2	Destino 3	Destino 4	oferta
Centro 1	x_{11}	x_{12}	x_{13}	x_{1m}	175
Centro 2	x_{21}	x_{22}	x_{23}	x_{2m}	100
Centro 3	x_{31}	x_{32}	x_{33}	x_{3m}	125
					total oferta: 400
demanda	80	70	70	80	total demanda: 300

Modelo de optimización

Es un problema de transporte con componentes condicionales asociados a la apertura o no de centros de distribución

$$Min \sum_i \sum_j c_{ij} x_{ij} + \sum_i k_j y_i$$

s.t.

$$\sum_j x_{ij} \leq a_i \qquad \forall i$$

$$\sum_i x_{ij} = b_j \qquad \forall j$$

$$\sum_j x_{ij} \leq M y_i \qquad \forall i$$

$$x \geq 0;\ x \epsilon \mathbb{R}$$

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

Conjuntos (sets)

i: nodos oferentes

j: nodos demandantes

Parámetros

b_j: demanda

a_i: oferta

c_{ij}: costo del arco de i a j

k_i: costo de operación del centro i

M: número muy grande

Variables de decisión

x_{ij}: cantidad de producto a enviar de i a j

y_i: decisión de abrir el centro i

Modelo de optimización

$$Min \sum_i \sum_j c_{ij} x_{ij} + \sum_i k_j y_i$$

s.t.

$$\sum_j x_{ij} \leq a_i \qquad \forall i$$

$$\sum_i x_{ij} = b_j \qquad \forall j$$

$$\sum_j x_{ij} \leq M y_i \qquad \forall i$$

$$x \geq 0;\ x \epsilon \mathbb{R}$$
$$y \in \{0,1\}$$

Función objetivo

$$Min \quad 434x_{11} + 523x_{12} + 640x_{13} + 850x_{14} + 323x_{21} + 480x_{22} + 670x_{23} + 770x_{24} + 997x_{31} + 680x_{32} + 390x_{33} + 590x_{34} + 55000y_1 + 45000y_2 + 48000y_3$$

s.t.

Restricciones de cumplimiento de oferta

$$x_{11} + x_{12} + x_{13} + x_{14} \leq 175$$
$$x_{21} + x_{22} + x_{23} + x_{24} \leq 100$$
$$x_{31} + x_{32} + x_{33} + x_{34} \leq 125$$

Restricciones de cumplimiento de demanda

$$x_{11} + x_{21} + x_{31} = 80$$
$$x_{12} + x_{22} + x_{32} = 70$$
$$x_{13} + x_{23} + x_{33} = 70$$
$$x_{14} + x_{24} + x_{34} = 80$$

Restricciones condicionales

$$x_{11} + x_{12} + x_{13} + x_{14} \leq My_1$$
$$x_{21} + x_{22} + x_{23} + x_{24} \leq My_2$$
$$x_{31} + x_{32} + x_{33} + x_{34} \leq My_3$$

$$x \geq 0;\ x \epsilon \mathbb{R}$$
$$y \in \{0,1\}$$

Solución con Python

```python
lp01 = pulp.LpProblem("set-cover", pulp.LpMinimize)

# Sets:
arcos = ['11', '12', '13', '14', '21', '22', '23', '24', '31', '32', '33', '34']
centros = range(3)
M = 99999

# Variables:
x = pulp.LpVariable.dicts('x', arcos, 0, None, cat='Continous')
y = pulp.LpVariable.dicts('y', centros, 0, None, cat='Binary')

# Función objetivo:
lp01 += 434 * x['11'] + 523 * x['12'] + 640 * x['13'] + 850 * x['14'] + \
        323 * x['21'] + 480 * x['22'] + 670 * x['23'] + 770 * x['24'] + \
        997 * x['31'] + 680 * x['32'] + 390 * x['33'] + 590 * x['34'] + \
        55000 * y[0] + 45000 * [1] + 48000 * y[2], 'Z'
```

```python
# Restricciones:
## Cumplimiento de oferta
lp01 += x['11'] + x['12'] + x['13'] + x['14'] <= 175
lp01 += x['21'] + x['22'] + x['23'] + x['24'] <= 100
lp01 += x['31'] + x['32'] + x['33'] + x['34'] <= 125

## Cumplimiento de demanda
lp01 += x['11'] + x['21'] + x['31'] == 80
lp01 += x['12'] + x['22'] + x['32'] == 70
lp01 += x['13'] + x['23'] + x['33'] == 70
lp01 += x['14'] + x['24'] + x['34'] == 80

## Restricciones condicionales
lp01 += x['11'] + x['12'] + x['13'] + x['14'] <= M * y[0]
lp01 += x['21'] + x['22'] + x['23'] + x['24'] <= M * y[1]
lp01 += x['31'] + x['32'] + x['33'] + x['34'] <= M * y[2]


# Resolucion:
lp01.solve()
```

Solución con Python

```
>> Optimal
>>
>> x_11 = 80
>> x_12 = 70
>> x_13 = 25
>> x_14 = 0
>> x_21 = 0
>> x_22 = 0
>> x_23 = 0
>> x_24 = 0
>> x_31 = 0
>> x_32 = 0
>> x_33 = 45
>> x_34 = 80
>> y_0 = 1
>> y_1 = 0
>> y_2 = 1
>>
>> Función objetivo: 255080.0
```