Programación Entera - Parte 2

Investigación Operativa, Universidad de San Andrés

Si encuentran algún error en el documento o hay alguna duda, mandenmé un mail a rodriguezf@udesa.edu.ar y lo revisamos.

1. Programación Mixta

En la clase anterior vimos que podíamos tener problemas en donde las variables sean de tipo sí / no, y que existían casos en donde podiamos tener en un mismo problema ambas cosas. Por ejemplo, un problema en donde tenemos que decidir si se opera una planta o no, y después cuanto produce cada planta. Vamos a ver justo ese ejemplo a continuación.

1.1. Planificación de producción

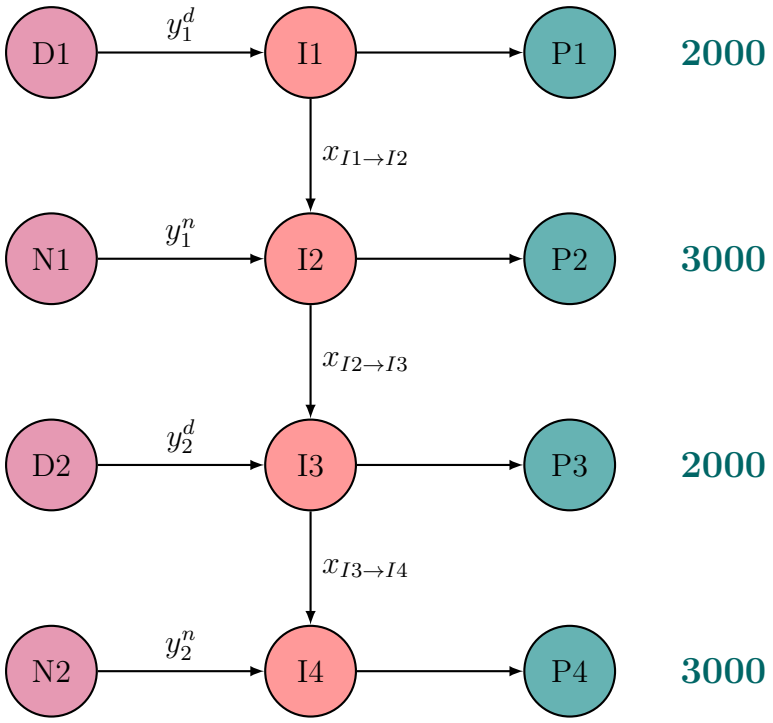
Una empresa opera con dos turnos: diurno y nocturno. Independientemente de la cantidad producida, el costo de poner en marcha la planta es:

- Turno diurno: \$8 000.
- Turno nocturno: \$4 500.

La demanda para los próximos dos días es de 2000 unidades para el día uno, 3000 unidades para la noche del día uno, 2000 unidades para el día dos y 3000 unidades para la noche del día dos. El costo de almacenaje es de \$1 por unidad por día. Se busca minimizar el costo total cumpliendo demanda. La entrega es inmediata.

1.1.1. Resolución

Acá debemos pensar como que tenemos cuatro oportunidades donde producir, dado que la entrega es inmediata. Además, tenemos tres oportunidades de almacenar y cuatro momentos en donde debemos recibir. Lo podemos gráficar de la siguiente manera para que quede más claro:



No olvidemos el costo de iniciar la producción, que es de \$8 000 para el turno diurno y \$4 500 para el nocturno, los cuales podemos plantear como dos variables:

$$c_d = 8000 \quad \text{y} \quad c_n = 4500$$

Nos queda entonces la siguiente función objetivo:

$$\text{Min } Z = c_d(y_1^d + y_2^d) + c_n(y_1^n + y_2^n) + \sum_{i=1}^{4} x_{Ii\to I(i+1)}$$

Si se produce o no es una cuestión de que la variable sea binaria, y lo que se almacena no puede ser negativo.

$$y_i^d, y_i^n \in \{0, 1\} \quad \forall i$$
$$x_{Ii\to I(i+1)} \geq 0 \quad \forall i$$

Las restricciones son de conservación de flujo (porque todo lo que entra debe salir) son:

$$
\begin{aligned}
y_1^d - X_{I1\to I2} &= X_{I1\to P_1} \\
y_1^n + X_{I1\to I2} - X_{I2\to I3} &= X_{I2\to P_2} \\
y_2^d + X_{I2\to I3} - X_{I3\to I4} &= X_{I3\to P_3} \\
y_2^n + X_{I3\to I4} &= X_{I4\to P_4}
\end{aligned}
$$

1.1.2. Resolución con PICOS

```python
import picos
import numpy as np

# Datos
costo_diurno = 8000
costo_nocturno = 4500
demanda = [2000, 3000, 2000, 3000]  # D1, N1, D2, N2
costo_almacenaje = 1

# Variables
P = picos.Problem()

# Variables binarias: si se produce en cada turno
y_d = picos.BinaryVariable('y_d', 2)  # diurno (dia 1 y dia 2)
y_n = picos.BinaryVariable('y_n', 2)  # nocturno (noche 1 y noche 2)

# Variables de produccion (cuantas unidades se producen en cada
    turno)
x_d = picos.RealVariable('x_d', 2, lower=0)
x_n = picos.RealVariable('x_n', 2, lower=0)

# Variables de inventario (cuanto se almacena al final de cada
    periodo, 3 momentos)
s = picos.RealVariable('s', 3, lower=0)

# Objetivo: costos fijos de produccion + costos de almacenaje
P.set_objective('min',
    costo_diurno * picos.sum(y_d) +
    costo_nocturno * picos.sum(y_n) +
    costo_almacenaje * picos.sum(s)
)
```

```
30
31 # Restricciones de produccion solo si se activa el turno
32 tipo_M = 10000
33 for i in range(2):
34     P.add_constraint(x_d[i] <= tipo_M * y_d[i])
35     P.add_constraint(x_n[i] <= tipo_M * y_n[i])
36
37 # Balance de inventario
38 # Periodo 0: D1
39 P.add_constraint(x_d[0] - demanda[0] == s[0])
40 # Periodo 1: N1
41 P.add_constraint(x_n[0] + s[0] - demanda[1] == s[1])
42 # Periodo 2: D2
43 P.add_constraint(x_d[1] + s[1] - demanda[2] == s[2])
44 # Periodo 3: N2
45 P.add_constraint(x_n[1] + s[2] - demanda[3] == 0)
46
47 P.solve(solver='glpk')
48 print('Produccion diurna:', np.round(x_d.value, 2))
49 print('Produccion nocturna:', np.round(x_n.value, 2))
50 print('Turnos diurnos activados:', [int(round(y_d[i].value)) for i
    in range(2)])
51 print('Turnos nocturnos activados:', [int(round(y_n[i].value)) for i
    in range(2)])
52 print('Inventario final en cada periodo:', np.round(s.value, 2))
53 print('Costo total:', round(P.value, 2))
```

1.1.3. Salida de la consola

```
Produccion diurna:
[[2000.]
[   0.]]
Produccion nocturna:
[[5000.]
[3000.]]
Turnos diurnos activados:
[1, 0]
Turnos nocturnos activados:
[1, 1]
Inventario final en cada periodo:
[[   0.]
[2000.]
```

```
[   0.]]
Costo total: 19000.0
```

1.2. Planificación eléctrica

Una empresa proveedora de energía debe satisfacer en el primer año una demanda de 80 millones de kWh. La demanda crece 20 millones de kWh por año, llegando al año cinco con una demanda de 160 millones de kWh. Se dispone de cuatro tipos de plantas con las características que se muestran a continuación:

Planta	Capacidad (millones kWh)	Costo construcción ($M)	Costo operativo anual ($M)
1	70	20	1.5
2	50	16	0.8
3	60	18	1.3
4	40	14	0.6

Se debe decidir en qué año construir cada planta (variable binaria) y cuándo operarla (otra variable binaria) para minimizar el costo total descontado.

1.2.1. Resolución

La forma de pensarlo es que tenemos las siguientes variables:

→ x_{ij}: Energía generada por la planta i en el año j (millones kWh). Variable continua ≥ 0.

→ α_{ij}: 1 si la planta i está operativa en el año j; 0 en caso contrario. Variable binaria.

→ y_i: 1 si la planta i se construye en algún momento. Variable binaria.

Además, tenemos los siguientes vectores:

→ $d = (80, 100, 120, 140, 160)$: Demanda de energía (millones kWh).

→ $cap = (70, 50, 60, 40)$: Capacidad de las plantas (millones kWh).

→ $c = (20, 16, 18, 14)$: Costo de construcción de las plantas (millones $).

→ $o = (1{,}5, 0{,}8, 1{,}3, 0{,}6)$: Costo operativo anual de las plantas (millones $).

La función objetivo entonces consiste en multiplicar el costo de construcción de cada planta por la variable binaria que indica si se construye o no, y sumarle el costo operativo anual de cada planta por la variable binaria que indica si se opera o no.

$$\text{Min } Z = \sum_{i=0}^{3} c_i \, y_i + \sum_{i=0}^{3} \sum_{j=0}^{4} o_i \, \alpha_{ij}$$

Las restricciones son:

(R1) **Demanda anual:** para cada año j se debe cubrir la demanda.

$$\sum_{i} x_{ij} \;\geq\; d_j \quad \forall\, j$$

(R2) **Capacidad de planta:**

$$x_{ij} \;\leq\; \text{cap}_i \, \alpha_{ij} \quad \forall\, i,\, j$$

(R3) **Operar sólo si se construyó:**

$$\alpha_{ij} \;\leq\; y_i \quad \forall\, i,\, j$$

(R4) **Permanencia**: una vez que está operativa, no se puede apagar en años posteriores.

$$\alpha_{i,j+1} \;\geq\; \alpha_{ij} \quad \forall\, i,\, j = 1, \ldots, T-1$$

1.2.2. Resolución con PICOS

```
import picos

# ---- Datos ----
demand = picos.Constant('demanda',  [80, 100, 120, 140, 160])
limits  = picos.Constant('cap',     [70, 50, 60, 40])
c       = picos.Constant('c',       [20, 16, 18, 14])
o       = picos.Constant('o',       [1.5, 0.8, 1.3, 0.6])

# ---- Variables ----
T, P = 5, 4                         # anios, plantas
x = picos.RealVariable('x',  (P, T), lower=0)
```

```
alfa = picos.BinaryVariable('alfa', (P, T))
y = picos.BinaryVariable('y', P)

p = picos.Problem()

# ---- Restricciones ----
# (R1) Demanda anual
for j in range(T):
    p.add_constraint(picos.sum(x[:, j]) >= demand[j])

# (R2) Capacidad de planta
for i in range(P):
    p.add_constraint(picos.sum(x[i, :]) <= 1e4*y[i])

    for j in range(T):
        p.add_constraint(x[i, j] <= 1e4*alfa[i, j])
        p.add_constraint(x[i, j] - limits[i] <= 1e4*(1-alfa[i, j]))

# (R3) Permanencia
for i in range(P):
    for j in range(T-1):
        p.add_constraint(alfa[i, j+1] >= alfa[i, j])

# (R4) Operacion implica construccion
for i in range(P):
    for j in range(T):
        p.add_constraint(alfa[i, j] <= y[i])

# ---- Objetivo ----
p.set_objective('min', c[0]*y[0] + c[1]*y[1] + c[2]*y[2] + c[3]*y[3]
    +
                    o[0]*picos.sum(alfa[0, :]) + o[1]*picos.sum(
   alfa[1, :]) +
                    o[2]*picos.sum(alfa[2, :]) + o[3]*picos.sum(
   alfa[3, :]))

# ---- Solve ----
p.options.verbosity = 1
p.solve(solver='glpk')

print(x)           # energia generada
print(p.value)     # costo total minimo
```

Notar que el 1e4 (el número 1000 en notación científica) es para que no se rompa la convexidad, se puede poner cualquier numero grande. La explicación mas larga es

que si no ponemos un numero grande, la restricción $x_{ij} \leq \text{cap}_i \, \alpha_{ij}$ puede ser que x_{ij} sea mayor que cap_i.

1.2.3. Salida de la consola

```
[ 7.00e+01  5.00e+01  7.00e+01  7.00e+01  7.00e+01]
[ 1.00e+01  5.00e+01  5.00e+01  5.00e+01  5.00e+01]
[ 0.00e+00  0.00e+00  0.00e+00  0.00e+00  0.00e+00]
[ 0.00e+00  0.00e+00  0.00e+00  2.00e+01  4.00e+01]
62.7
```