# Programación Entera - Parte 1

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. Introducción a la Programación Entera

La **programación entera** (PE) es una rama de la optimización matemática en la cual se busca la mejor solución a problemas cuyas variables de decisión deben tomar valores enteros. Dentro de la PE distinguimos dos grandes familias:

- **Programación Entera Pura**: *todas* las variables son enteras.
- **Programación Mixta Entera**: solo una parte de las variables son enteras (generalmente binarias) y el resto pueden ser continuas.

A diferencia de la Programación Lineal (PL), la PE suele requerir algoritmos de *ramificación y acotación*, *cortes* y otras heurísticas. Esto se debe a que la región factible deja de ser convexa una vez que exigimos integralidad.

### 1.1. ¿Cuándo aparece la PE?

La necesidad de variables enteras surge, por ejemplo, cuando:

- Tomamos decisiones de tipo *sí / no* (abrir o no una planta, invertir o no en un proyecto, etc.).
- Trabajamos con ítems indivisibles (personas, máquinas, micros, etc.).
- Modelamos relaciones lógicas (mutua exclusión, dependencia, *al menos / a lo sumo* $k$, etc.).

## 2. Modelado típico con variables binarias

Las variables binarias $y \in \{0, 1\}$ permiten representar un amplio abanico de restricciones lógicas.

### 2.1. Mutuamente excluyentes

Para que dos actividades $x_1$ y $x_2$ no puedan realizarse simultáneamente, basta con exigir lo siguiente:

$$y_1 + y_2 \leq 1, \qquad y_j \in \{0, 1\}.$$

### 2.2. Dependencia (precedencia)

Si la actividad B solo puede realizarse cuando también se realiza la A lo planteamos de esta manera:

$$y_B \leq y_A.$$

### 2.3. Costos fijos de puesta en marcha

Para abrir un turno, una planta o cualquier recurso con costo fijo $F$ y costo variable $c$, debemos pensarlo así:

$$\begin{aligned} Fy + cx \quad &\text{(costo)} \\ x \leq My \quad &\text{(capacidad)} \end{aligned}$$

## 3. Ejemplos prácticos

### 3.1. Problema de inversión

Los integrantes de una mesa directiva analizan 6 inversiones. Cada una puede realizarse a lo sumo una vez y difieren en el valor presente neto (VPN) que generan y el capital inicial requerido.

| Inversión | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| Ganancia estimada | 15 | 12 | 16 | 18 | 9 | 11 |
| Capital requerido | 38 | 33 | 39 | 45 | 23 | 27 |

El capital disponible es sólo de 100 millones. Las oportunidades 1 y 2 son mutuamente excluyentes, al igual que las 3 y 4. Además, ni la opción 3 ni la 4 pueden hacerse a menos que se lleve a cabo la 1 o la 2. El objetivo es, obviamente, maximizar la ganancia total.

### 3.1.1. Resolución

Una posible formulación utiliza $y_j \in \{0, 1\}$ indicando si se selecciona la inversión $j$. La función objetivo consiste en multiplicar la ganancia de cada proyecto por la variable binaria que indica si se selecciona o no.

$$\text{Max } Z = 15y_1 + 12y_2 + 16y_3 + 18y_4 + 9y_5 + 11y_6$$

Las restricciones quedan de la siguiente manera:

$$\begin{aligned}
&38y_1 + 33y_2 + 39y_3 + 45y_4 + 23y_5 + 27y_6 \leq 100 \quad \text{(capital disponible)} \\
&y_1 + y_2 \leq 1 \quad \text{(mutuamente excluyentes)} \\
&y_3 + y_4 \leq 1 \quad \text{(mutuamente excluyentes)} \\
&y_3 \leq y_1 + y_2 \quad \text{(si se hace la 1 o la 2, se puede hacer la 3 o la 4)} \\
&y_4 \leq y_1 + y_2 \quad \text{(si se hace la 1 o la 2, se puede hacer la 3 o la 4)} \\
&y_j \in \{0, 1\} \quad \forall j \quad \text{(variables binarias)}
\end{aligned}$$

### 3.1.2. Resolución con PICOS

```
import picos

# Crear el problema
P = picos.Problem()

# Variables binarias (una por inversion)
y = picos.BinaryVariable('y', 6)

# Funcion objetivo
P.set_objective('max', 15*y[0] + 12*y[1] + 16*y[2] + 18*y[3] + 9*y
    [4] + 11*y[5])

# Restricciones
P.add_constraint(38*y[0] + 33*y[1] + 39*y[2] + 45*y[3] + 23*y[4] +
    27*y[5] <= 100)
P.add_constraint(y[0] + y[1] <= 1)  # inversiones 1 y 2 excluyentes
P.add_constraint(y[2] + y[3] <= 1)  # inversiones 3 y 4 excluyentes
P.add_constraint(y[2] <= y[0] + y[1])  # precedencia
P.add_constraint(y[3] <= y[0] + y[1])  # precedencia

```

```
19 # Resolver
20 P.options.verbosity = 1
21 P.solve(solver='glpk')
22
23 print(y)          # valores optimos de las variables
24 print(P.value)    # ganancia total optima
```

### 3.1.3. Salida de la consola

```
Binary Program
    maximize: 15*y[0] + 12*y[1] + 16*y[2] + 18*y[3] + 9*y
   [4] + 11*y[5]
    over:
      6 Boolean variables y
    subject to:
      38*y[0] + 33*y[1] + 39*y[2] + 45*y[3] + 23*y[4] +
   27*y[5] <= 100
      y[0] + y[1] <= 1
      y[2] + y[3] <= 1
      y[2] <= y[0] + y[1]
      y[3] <= y[0] + y[1]

[1.00e+0 0.00e+0 0.00e+0 0.00e+0 1.00e+0 1.00e+0]
35.0
```

## 3.2. Asignación de productos a plantas

Una compañía producirá cuatro productos empleando tres plantas con capacidad ociosa. El esfuerzo productivo por unidad es comparable en todas las plantas; por lo tanto, la capacidad disponible se mide en unidades de cualquier producto por día.

| | Costo unitario ($/u) | | | | Capacidad disponible (u/día) |
|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | |
| Planta 1 | 41 | 27 | 28 | 24 | 75 |
| Planta 2 | 40 | 29 | – | 23 | 75 |
| Planta 3 | 37 | 30 | 27 | 21 | 45 |

La demanda diaria proyectada es:

Producto 1: 20 Producto 2: 30 Producto 3: 30 Producto 4: 40

La dirección debe decidir la asignación con dos criterios:

a) **Permitir división**: un producto puede fabricarse en más de una planta (modelo de transporte).

b) **No permitir división**: cada producto se fabrica en una sola planta (modelo de asignación binaria).

Para el caso (b) definimos $y_{ij} = 1$ si el producto $j$ se produce en la planta $i$ y 0 en caso contrario. Luego sumamos una restricción $\sum_i y_{ij} = 1$ para cada producto $j$.

## 3.3. Planificación de producción

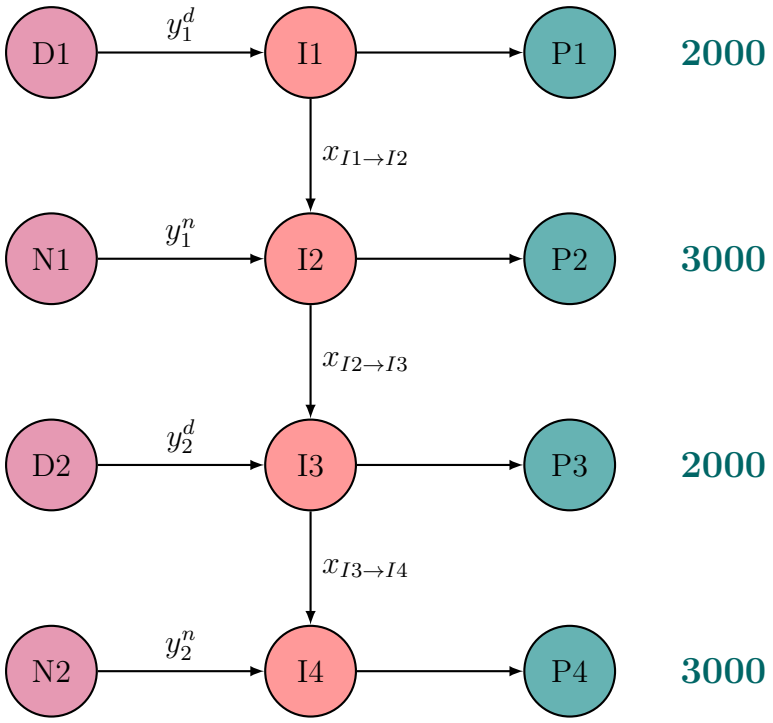
Una empresa opera con dos turnos: diurno y nocturno. Independientemente de la cantidad producida, el costo de poner en marcha la planta es:

- 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.

### 3.3.1. 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:

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

Las primeras restricciones que debemos pensar son las de que todo debe ser mayor que 0:

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

Además, tenemos que agregar una restricción de activación:
Las restricciones son de conservación de flujo (porque todo lo que entra debe salir) son:

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

El parámetro $M$ es un número grande (por ejemplo 10 000) que garantiza la validez de la restricción de activación.

## 3.4. 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.

### 3.4.1. 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 $\geq 0$.

→ $\alpha_{ij}$: 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.

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

### 3.4.2. 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 $\text{cap}_i$.

### 3.4.3. Salida de la consola

```
Mixed Integer Linear Program
    minimize: 20*y[0] + 16*y[1] + 18*y[2] + 14*y[3] + 1.5*
   sum(alfa(0,:)) + 0.8*sum(alfa(1,:)) + 1.3*sum(alfa(2,:)
   ) + 0.6*sum(alfa(3,:))...

Optimal solution found.

[[7.00e+01 5.00e+01 7.00e+01 7.00e+01 7.00e+01]
 [1.00e+00 5.00e+00 5.00e+00 5.00e+00 5.00e+00]
 [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]]

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

## 3.5. Localización de estaciones de bomberos

Se pretende ubicar dos estaciones de bomberos en una comunidad dividida en cinco sectores. No se pueden tener más de una estación en un mismo sector. Una estación solo puede atender a su propio sector y a los otros que le sean asignados. La matriz muestra el tiempo medio de respuesta (minutos) si una estación ubicada en el sector $i$ atiende un incendio en el sector $j$.

| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 | 5 | 12 | 30 | 20 | 15 |
| 2 | 20 | 4 | 15 | 10 | 25 |
| 3 | 15 | 20 | 6 | 15 | 12 |
| 4 | 25 | 15 | 25 | 4 | 10 |
| 5 | 10 | 25 | 15 | 12 | 5 |

La frecuencia promedio de incendios es de 2 por día en el sector 1, 1 por día en el sector 2, 3 por día en el sector 3, 1 por día en el sector 4 y 3 por día en el sector 5.

### 3.5.1. Resolución

La forma de pensarlo es que tenemos las siguientes variables:

→ $y_i$: 1 si se instala una estación en el sector $i$. Variable binaria.

→ $x_{ij}$: 1 si el sector $j$ es atendido por la estación ubicada en $i$. Variable binaria.

Además, tenemos los siguientes vectores:

→ $d = (2, 1, 3, 1, 3)$: Frecuencia promedio de incendios por sector.

La función objetivo es:

$$\text{Min } Z = \sum_{i=1}^{5} \sum_{j=1}^{5} x_{ij} \cdot d_j$$

Para poner las restricciones debemos pensar en que en primer lugar no se puede tener más de una estación en un mismo sector, y en segundo lugar una estación solo puede atender a su propio sector y a los otros que le sean asignados (es decir, son mutuamente excluyentes).

$$\sum_{i=1}^{5} y_{ij} \leq 1 \quad \forall j$$

$$\sum_{j=1}^{5} x_{ij} = 1 \quad \forall i$$

### 3.5.2. Resolución con PICOS