Programación Lineal - 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 Lineal

La programación lineal es una técnica de optimización matemática que busca encontrar la mejor solución a problemas donde tanto la función objetivo como las restricciones son lineales. Es una herramienta fundamental en la toma de decisiones y la optimización de recursos.

2. Problema PL general

Vamos a construir un molde, al cual se pueden adaptar los problemas de programación lineal (PL).

Letras y significado:

- **Z**: Función objetivo. Es la medida de la performance que queremos optimizar (maximizar o minimizar).
- x_j: Variable de decisión. Representa el nivel de cada actividad j.
- c_j: Parámetro. Indica el aumento de Z que se obtendría si aumentara la variable de decisión x_j en una unidad.
- b_i: Parámetro. Cantidad del recurso i disponible para asignar.
- a_{ij}: Parámetro. Cantidad del recurso i que se consume por cada unidad de la actividad j.

Resumen:

- **Variables de decisión:** x_j
- **Función objetivo:** Z
- **Parámetros:** c_j, b_i, a_{ij}

3. Problema PL general

Cualquier problema que pueda escribirse de esta forma es PL:

$$x = (x_1, x_2, \ldots, x_n)$$
$$c = (c_1, c_2, \ldots, c_n)$$
$$b = (b_1, b_2, \ldots, b_n)$$

$$a = \begin{pmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1m} \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2m} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & a_{n3} & \cdots & a_{nm} \end{pmatrix}$$

Min Z

Dado

$$Z = cx^t$$
$$ax^t \leq b$$

4. Ejemplo 1: Panes y tortas

Jorge tiene una panadería que vende panes y tortas. Cada pan requiere 1 kg de harina y 0.2 kg de levadura, mientras que cada torta requiere de 0.5 kg de harina, 0.1 kg de levadura y 0.2 kg de azúcar. Con lo que se compra por mes se calcula que se tiene 50 kg de harina para cada dia, 15 kg de levadura y 10 kg de azúcar. La ganancia por cada pan es de \$2.5 y por cada torta \$1.8. Se estima que la demanda es de al menos 20 panes y 15 tortas, pero no podemos producir más de 50 panes o 20 tortas.

4.1. Formulación

Primero identificamos las variables de decisión, que en este caso va a ser la cantidad de panes y tortas que se van a producir.

- x_1: Cantidad de panes a producir
- x_2: Cantidad de tortas a producir

Ahora vamos a necesitar identificar en el texto anterior la función objetivo. La frase "La ganancia por cada pan es de \$2.5 y por cada torta \$1.8" nos dice que la función objetivo es:

$$Z = 2{,}5x_1 + 1{,}8x_2$$

Ahora vamos a identificar las restricciones. La frase "Cada pan requiere 1 kg de harina y 0.2 kg de levadura, mientras que cada torta requiere de 0.5 kg de harina, 0.1 kg de levadura y 0.2 kg de azúcar" nos dice que las restricciones son:

$$\begin{aligned} 1x_1 + 0{,}5x_2 &\leq 50 \\ 0{,}2x_1 + 0{,}1x_2 &\leq 15 \\ 0{,}2x_2 &\leq 10 \end{aligned}$$

Por otro lado tenemos las demandas, en la frase "Se estima que la demanda es de al menos 20 panes y 15 tortas, pero no podemos producir más de 50 panes o 20 tortas" nos dice que las restricciones son:

$$\begin{aligned} x_1 &\geq 20 \\ x_2 &\geq 15 \\ x_1 &\leq 50 \\ x_2 &\leq 20 \end{aligned}$$

Con esto ya tenemos el problema formulado. Nos ocuparemos de resolverlo en otro momento.

5. Ejemplo 2: Wyndor Glass Co.

El siguiente ejemplo está sacado del libro "Introduction to Operations Research" de Hillier y Lieberman, el cual es bibliografía de la materia.

5.1. Enunciado

Una empresa (Wyndor Glass Co.) produce puertas y ventanas. Deciden poner en producción dos productos nuevos:

- **Producto 1:** Una ventana de 2m de altura con marco de aluminio
- **Producto 2:** Una ventana colgante de 3m con marco de madera

Wyndor tiene 3 fábricas que cumplen funciones diferentes. El **producto 1** requiere producción de la **fábrica 1 y 3** mientras que el **producto 2** requiere de las **plantas 2 y 3**. La división de marketing de la empresa hizo un estudio en el cual llegaron

a la conclusión de que podrían vender la misma cantidad de ambos productos. Sin embargo, como ambos están compitiendo por tiempo de producción en la fábrica 3 (el cual es limitado), la compañía debe **decidir cuánto hacer de cada uno** para obtener el mayor retorno posible. Queremos saber cuánto de cada producto debemos fabricar para maximizar el retorno teniendo en cuenta las horas de producción limitadas de las fábricas.

5.2. Datos del problema

El modo de fabricacion que tiene la compania es del tipo batch, por lo que el rate de produccion (**variables de decision**) son el numero de lotes por semana.

Planta	**Producto 1**	**Producto 2**	**Tiempo disponible**
1	1	0	4
2	0	2	12
3	3	2	18

Producto	**Ganancia por Batch**
Producto 1	\$3000
Producto 2	\$5000

5.3. Formulacion

Antes de armar el modelo, vamos a aprovechar para introducir la notación general que vamos a usar en este tipo de problemas, siguiendo el molde clásico:

- **Z**: Ganancia total (función objetivo a maximizar)
- x_j: Cantidad de batches (lotes) a producir del producto j
- c_j: Ganancia que aporta cada batch del producto j
- b_i: Cantidad de horas disponibles en la planta i
- a_{ij}: Cantidad de horas que necesita el producto j en la planta i

Vamos a formular el problema paso a paso:

1. Primero definimos las variables de decisión:

   - x_1: Numero de lotes por semana del Producto 1
   - x_2: Numero de lotes por semana del Producto 2

2. Luego definimos la funcion objetivo:

$$\text{Maximizar:} \quad Z = 3000x_1 + 5000x_2$$

3. Luego agregamos las restricciones de capacidad de cada planta:

$$\begin{aligned} x_1 &\leq 4 \quad \text{(Planta 1)} \\ 2x_2 &\leq 12 \quad \text{(Planta 2)} \\ 3x_1 + 2x_2 &\leq 18 \quad \text{(Planta 3)} \end{aligned}$$

4. Finalmente, agregamos las restricciones de no negatividad:

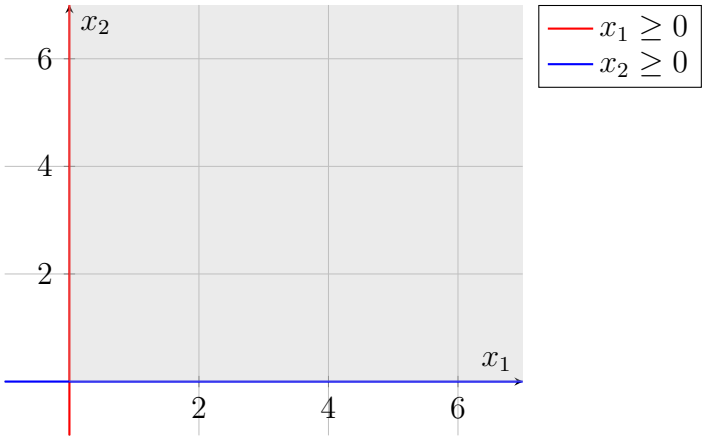
$$\begin{aligned} x_1 &\geq 0 \\ x_2 &\geq 0 \end{aligned}$$

6. Método Simplex

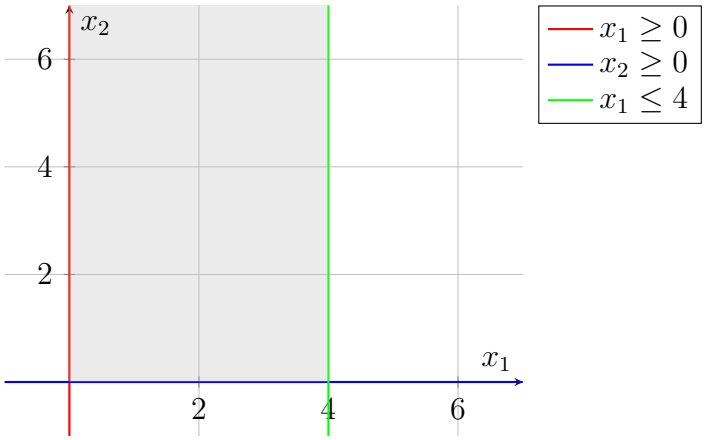
El método simplex es un algoritmo iterativo que se utiliza para resolver problemas de programación lineal. Este método se basa en la idea de que la solución óptima se encuentra en uno de los vértices de la región factible.

6.1. Solución paso a paso

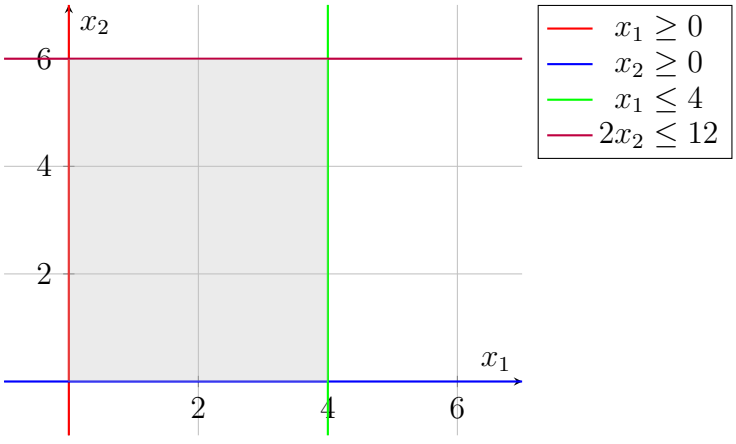
Primero graficamos los ejes y las restricciones de no negatividad, la región gris es lo que llamamos la región factible. Esta se va a ir achicando y construyendo a medida que agregamos las restricciones.



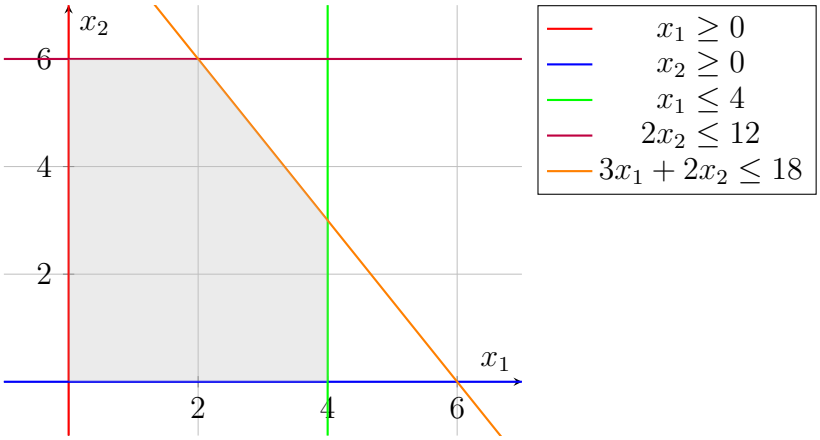
Luego agregamos la restricción $x_1 \leq 4$:



Agregamos la restricción $2x_2 \leq 12$:



Finalmente agregamos la restriccion $3x_1 + 2x_2 \leq 18$:



Para encontrar la solución óptima, evaluamos la función objetivo $Z = 3000x_1 + 5000x_2$ en los vértices de la región factible. En este caso dado que necesitamos que los números sean enteros (porque no podemos producir 2.4 de un producto) lo que hacemos es redondear para abajo. El método simplex evalúa en un punto y se mueve hacia un punto adyacente a evaluar si es mejor o peor. Una vez que llega a un punto que no tiene ningún punto adyacente mejor, se detiene y se dice que encontró la solución óptima.

- $(0, 0)$: $Z = 0$
- $(0, 6)$: $Z = 30000$ (mejor que el anterior)
- $(2, 6)$: $Z = 36000$ (mejor que el anterior)
- $(4, 3)$: $Z = 27000$ (peor, podríamos frenar)
- $(4, 0)$: $Z = 12000$

Por lo tanto, la solución óptima es producir 2 lotes del Producto 1 y 6 lotes del Producto 2, obteniendo una ganancia de $36000.

7. Implementación con PICOS

PICOS es una interfaz de Python para varios solucionadores de optimización. Acá abajo vemos cómo sería formular y resolver problemas de programación lineal utilizando esta librería, con el ejemplo de Wyndor Glass Co.:

```python
import picos
import numpy as np

# Crear un problema
P = picos.Problem()

# Definir variables
x = picos.RealVariable('x', 2)  # Vector de 2 variables

# Definir funcion objetivo
P.set_objective('max', 3000*x[0] + 5000*x[1])

# Agregar restricciones
P.add_constraint(x[0] <= 4)
P.add_constraint(2*x[1] <= 12)
P.add_constraint(3*x[0] + 2*x[1] <= 18)

P.options.verbosity = 1 # para ver los mensajes de debug

print(P)

P.solve(solver='glpk')
print(x)
print(P.value)
```

7.1. Salida de la consola

```
Linear Program
    maximize: 3000*x[0] + 5000*x[1]
    over:
      2x1 real variable x
    subject to:
      x[0] <= 4
      2*x[1] <= 12
      3*x[0] + 2*x[1] <= 18

[2.0, 6.0]
36000.0
```

Viendo la salida de la consola podemos ver entonces que la solución óptima es que se hagan 2 unidades del producto 1 y 6 unidades del producto 2, y que el valor óptimo de la función objetivo es 36000. La primera parte donde se explica el problema es

por poner el P.options.verbosity = 1.

7.2. Planteo alternativo en PICOS

```
P = picos.Problem()

x = picos.RealVariable('x', 2)

A = np.array([[1,0],
              [0,2],
              [3,2]])
c = np.array([3000,5000])
b = np.array([4,12,18])

c = picos.Constant('c', c)
A = picos.Constant('A', A)
b = picos.Constant('b', b)

P.set_objective('max', c|x)

P.add_constraint(A*x <= b)

P.options.verbosity = 1

print(P)

P.solve(solver='glpk')
print(x)
print(P.value)
```

¡Cuidado con las dimensiones! En PICOS, cuando usamos `picos.Constant(nombre, array, shape)` hay que asegurarse que las dimensiones coincidan con el modelo matemático. Así, todos los problemas se pueden escribir de la misma forma que en la teoría.

8. Ejercicios Extra

8.1. Producción de lámparas

Una compañía produce dos tipos de lámparas (productos 1 y 2) que requieren partes de metal y componentes eléctricos. La administración desea determinar cuántas unidades de cada producto debe fabricar para maximizar la ganancia. Para cada unidad

del producto 1 se requieren 1 unidad de partes de metal y 2 unidades de componentes eléctricos. Para cada unidad del producto 2 se necesitan 3 unidades de partes de metal y 2 unidades de componentes eléctricos. La compañía tiene 200 unidades de partes de metal y 300 de componentes eléctricos. Cada unidad del producto 1 da una ganancia de \$1 y cada unidad del producto 2, hasta 60 unidades, da una ganancia de \$2.

8.1.1. Planteo del modelo:

Sean x_1 la cantidad de lámparas del producto 1 y x_2 la cantidad del producto 2.

$$\begin{aligned} \text{Maximizar:} \quad & Z = x_1 + 2x_2 \\ \text{Sujeto a:} \quad & x_1 + 3x_2 \leq 200 \\ & 2x_1 + 2x_2 \leq 300 \\ & x_2 \leq 60 \\ & x_1 \geq 0,\ x_2 \geq 0 \end{aligned}$$

8.1.2. Resolución en Python usando PICOS (forma directa)

```python
import picos
import numpy as np

P = picos.Problem()

x = picos.RealVariable('x', 2)

# Matriz de restricciones
A = np.array([
    [1, 3],  # partes de metal
    [2, 2],  # componentes electricos
    [0, 1]   # limite de producto 2
])
b = np.array([200, 300, 60])
c = np.array([1, 2])

A = picos.Constant('A', A)
b = picos.Constant('b', b)
c = picos.Constant('c', c)

P.set_objective('max', c | x)
P.add_constraint(A * x <= b)
P.add_constraint(x >= 0)
```

```
24
25 P.options.verbosity = 1
26
27 P.solve(solver='glpk')
28 print(x)
29 print(P.value)
```

```
[ 90.00e+0  60.00e+0]
210.0
```

8.1.3. Resolución en Python usando PICOS (forma manual)

```
1  import picos
2
3  P = picos.Problem()
4
5  x1 = picos.RealVariable('x1', 1)
6  x2 = picos.RealVariable('x2', 1)
7
8  P.set_objective('max', x1 + 2*x2)
9  P.add_constraint(x1 + 3*x2 <= 200)
10 P.add_constraint(2*x1 + 2*x2 <= 300)
11 P.add_constraint(x2 <= 60)
12 P.add_constraint(x1 >= 0)
13 P.add_constraint(x2 >= 0)
14
15 P.options.verbosity = 1
16
17 P.solve(solver='glpk')
18 print(f'x1 = {x1.value}, x2 = {x2.value}')
19 print(P.value)
```

```
x1 = 90.0, x2 = 60.0
210.0
```

8.2. Dieta óptima

Mi dieta requiere que toda la comida que consuma provenga de uno de los cuatro "grupos básicos de alimentos" (torta de chocolate, helado, gaseosa y cheesecake). Actualmente, los siguientes cuatro alimentos están disponibles para el consumo: brownies, helado de chocolate, gaseosa y cheesecake de ananá. Cada brownie cuesta 50 centavos, cada porción de helado de chocolate cuesta 20 centavos, cada botella de

gaseosa cuesta 30 centavos y cada porción de cheesecake de ananá cuesta 80 centavos. Cada día, debo ingerir al menos 500 calorías, 6 oz de chocolate, 10 oz de azúcar y 8 oz de grasa. El contenido nutricional por unidad de cada alimento se muestra en la siguiente tabla:

Alimento	**Calorías**	**Chocolate (oz)**	**Azúcar (oz)**	**Grasa (oz)**
Brownie	400	3	2	2
Helado de chocolate	200	2	2	4
Gaseosa	150	0	4	1
Cheesecake de ananá	500	0	4	5

8.2.1. Planteo del modelo:

Las **variables de decisión** en este caso va a ser cuanto cómo de cada cosa. Es importante entender que las variables de decisión son las que se pueden controlar, y que las restricciones son las que no se pueden controlar.

- x_1: cantidad de brownies a consumir por día
- x_2: cantidad de porciones de helado de chocolate a consumir por día
- x_3: cantidad de botellas de gaseosa a consumir por día
- x_4: cantidad de porciones de cheesecake de ananá a consumir por día

La **función objetivo** va a ser minimizar el costo total de la dieta, obviamente. Voy a entonces multiplicar cada variable de decisión por su costo y sumarlo todo.

$$\text{Minimizar:} \quad Z = 0{,}5x_1 + 0{,}2x_2 + 0{,}3x_3 + 0{,}8x_4$$

Las **restricciones** van a ser las que nos limitan a lo que podemos comer. Vamos a ir una por una:

1. Primero, necesitamos al menos 500 calorías por día. Cada brownie aporta 400 calorías, cada helado 200, cada gaseosa 150 y cada cheesecake 500. Entonces:

$$400x_1 + 200x_2 + 150x_3 + 500x_4 \geq 500 \qquad \text{(calorías)}$$

2. Necesitamos al menos 6 oz de chocolate. Solo los brownies (3 oz) y el helado (2 oz) contienen chocolate:

$$3x_1 + 2x_2 + 0x_3 + 0x_4 \geq 6 \qquad \text{(chocolate)}$$

3. Necesitamos al menos 10 oz de azúcar. Los brownies aportan 2 oz, el helado 2 oz, la gaseosa 4 oz y el cheesecake 4 oz:

$$2x_1 + 2x_2 + 4x_3 + 4x_4 \geq 10 \qquad \text{(azúcar)}$$

4. Necesitamos al menos 8 oz de grasa. Los brownies aportan 2 oz, el helado 4 oz, la gaseosa 1 oz y el cheesecake 5 oz:

$$2x_1 + 4x_2 + 1x_3 + 5x_4 \geq 8 \qquad \text{(grasa)}$$

5. Finalmente, no podemos consumir cantidades negativas de ningún alimento:

$$x_1 \geq 0, \quad x_2 \geq 0, \quad x_3 \geq 0, \quad x_4 \geq 0$$

8.2.2. Resolución en Python usando PICOS (forma directa)

```python
import picos
import numpy as np

# Crear el problema
P = picos.Problem()

# Definir variables
x = picos.RealVariable('x', 4)  # Vector de 4 variables

# Definir funcion objetivo
P.set_objective('min', 0.5*x[0] + 0.2*x[1] + 0.3*x[2] + 0.8*x[3])

# Agregar restricciones
P.add_constraint(400*x[0] + 200*x[1] + 150*x[2] + 500*x[3] >= 500)
    # calorias
P.add_constraint(3*x[0] + 2*x[1] >= 6)  # chocolate
P.add_constraint(2*x[0] + 2*x[1] + 4*x[2] + 4*x[3] >= 10)  # azucar
P.add_constraint(2*x[0] + 4*x[1] + 1*x[2] + 5*x[3] >= 8)  # grasa
P.add_constraint(x >= 0)  # no negatividad

# Resolver
P.solve(solver='glpk')

```

```python
# Mostrar resultados
print("Solucion optima:")
print(f"Brownies: {x[0].value:.2f}")
print(f"Helado: {x[1].value:.2f}")
print(f"Gaseosa: {x[2].value:.2f}")
print(f"Cheesecake: {x[3].value:.2f}")
print(f"Costo total: ${P.value:.2f}")
```

```
Solucion optima:
Brownies: 0.00
Helado: 3.00
Gaseosa: 1.00
Cheesecake: 0.00
Costo total: $0.90
```

8.2.3. Resolución en Python usando PICOS (forma manual)

```python
import picos
import numpy as np

# Crear el problema
P = picos.Problem()

# Definir variables
x = picos.RealVariable('x', 4)

# Definir coeficientes
c = np.array([0.5, 0.2, 0.3, 0.8])  # costos
A = np.array([
    [400, 200, 150, 500],  # calorias
    [3, 2, 0, 0],          # chocolate
    [2, 2, 4, 4],          # azucar
    [2, 4, 1, 5]           # grasa
])
b = np.array([500, 6, 10, 8])  # requerimientos minimos

# Definir funcion objetivo usando producto punto
P.set_objective('min', c @ x)

# Agregar restricciones usando matrices
for i in range(len(b)):
    P.add_constraint(A[i] @ x >= b[i])

# Agregar restricciones de no negatividad
```

```
P.add_constraint(x >= 0)

# Resolver
P.solve(solver='glpk')

# Mostrar resultados
print("Solucion optima:")
print(f"Brownies: {x[0].value:.2f}")
print(f"Helado: {x[1].value:.2f}")
print(f"Gaseosa: {x[2].value:.2f}")
print(f"Cheesecake: {x[3].value:.2f}")
print(f"Costo total: ${P.value:.2f}")
```

```
Solucion optima:
Brownies: 0.00
Helado: 3.00
Gaseosa: 1.00
Cheesecake: 0.00
Costo total: $0.90
```

9. Casos Especiales

- **Solución única:** El óptimo se encuentra en un único vértice
- **Múltiples soluciones:** El óptimo se encuentra en una arista o cara
- **Problema no acotado:** No existe un valor óptimo finito
- **Problema infactible:** No existe solución que satisfaga todas las restricciones