Guía de Ejercicios
Programación Lineal

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. Ejercicios

1.1. Puntos y Valores Óptimos en Gráficos

Para los siguientes problemas de optimización encontrar gráficamente el/los punto(s) óptimo(s) y el valor óptimo. Para hacer esto pueden graficar las funciones objetivo en el rango de interés (o sea, que por lo menos contenga el conjunto factible), y visualmente identificar el punto y valor óptimo. Para los casos donde haya más de un óptimo, reportar todos los puntos óptimos que existan. Para los casos en que no existe el óptimo, explicar por qué este es el caso (o sea, si es porque el conjunto factible es nulo, o porque la función objetivo no está acotada).

a) $\boxed{\textbf{min} \quad x^2}$

b) $\boxed{\begin{array}{l} \textbf{min} \quad x^2 \\ x \geq 2 \end{array}}$

c) $\boxed{\begin{array}{l} \textbf{max} \quad x \\ 0 \leq x \end{array}}$

d) $\boxed{\begin{array}{l} \textbf{max} \quad x \\ 0 \leq x \\ x \leq 10 \end{array}}$

e) $\boxed{\begin{array}{l} \textbf{max} \quad x \\ x \leq 0 \\ 2 \leq x \end{array}}$

f) $\boxed{\begin{array}{l} \textbf{min} \quad \cos(x) \\ 0 \leq x \\ x \leq 1 \end{array}}$

g) $\boxed{\begin{array}{l} \textbf{min} \quad \cos(x) \\ 0 \leq x \\ x \leq 10 \end{array}}$

1.2. Producción en Fábrica

Una empresa fabrica dos productos A y B. El producto A requiere 2 horas de mano de obra y 3 kg de materia prima, mientras que el producto B requiere 3 horas de mano de obra y 2 kg de materia prima. La empresa dispone de 100 horas de mano de obra y 80 kg de materia prima por semana. El beneficio por unidad es de \$40 para

A y \$30 para B. Determine la cantidad óptima a producir de cada producto para maximizar el beneficio.

1.3. Dieta Nutritiva

Una dietista debe preparar una mezcla nutritiva que contenga al menos 12 unidades de vitamina A y 15 unidades de vitamina B. Dispone de dos tipos de alimentos: el alimento 1 contiene 2 unidades de vitamina A y 3 de B por kg, y cuesta \$4 por kg; el alimento 2 contiene 3 unidades de vitamina A y 1 de B por kg, y cuesta \$3 por kg. ¿Qué cantidad debe usar de cada alimento para minimizar el costo?

1.4. Producción de Sillas y Mesas

Una fábrica produce sillas y mesas. Cada silla requiere 2 unidades de madera y 1 unidad de trabajo, mientras que cada mesa requiere 3 unidades de madera y 2 unidades de trabajo. La fábrica dispone de 60 unidades de madera y 40 unidades de trabajo. La ganancia por silla es de \$20 y por mesa es de \$30. ¿Cuántas sillas y mesas debe producir para maximizar la ganancia?

1.5. Transporte de Mercancías

Una empresa de transporte tiene dos camiones. El camión 1 puede transportar 10 toneladas y recorre 40 km/h, mientras que el camión 2 puede transportar 15 toneladas y recorre 30 km/h. Se necesita transportar al menos 50 toneladas de mercancía y se dispone de 8 horas. El costo por hora del camión 1 es \$100 y del camión 2 es \$120. ¿Cuántos viajes debe hacer cada camión para minimizar el costo?

1.6. Producción de Juguetes

Una empresa produce dos tipos de juguetes: coches y trenes. Cada coche requiere 4 horas en el departamento A y 2 horas en el B, mientras que cada tren requiere 2 horas en A y 5 horas en B. Se dispone de 100 horas en A y 80 horas en B. El beneficio es de \$8 por coche y \$7 por tren. ¿Cuántos juguetes de cada tipo debe producir?

1.7. Producción de Tres Productos

Una empresa produce tres tipos de productos: A, B y C. La siguiente tabla muestra los recursos necesarios por unidad y la disponibilidad total:

Recurso	Producto A	Producto B	Producto C	Disponible
Mano de obra (h)	4	3	5	400
Material (kg)	2	4	3	300
Tiempo máquina (h)	3	2	4	350
Beneficio ($/u)	100	80	120	

¿Cuántas unidades de cada producto debe fabricar para maximizar el beneficio?

1.8. Inversiones

Una empresa de inversiones tiene $100,000 para invertir en tres opciones diferentes. La opción A tiene un rendimiento del 8 % anual pero requiere una inversión mínima de $20,000. La opción B tiene un rendimiento del 12 % anual con un máximo de inversión de $50,000. La opción C tiene un rendimiento del 10 % anual. Por política de la empresa, la inversión en C debe ser al menos el 30 % de la inversión total. ¿Cómo debe distribuir el dinero para maximizar el rendimiento?

1.9. Producción de Muebles

Una fábrica de muebles produce mesas, sillas y estantes. Cada producto requiere madera, tiempo de carpintería y tiempo de acabado según la siguiente tabla:

Recurso	Mesa	Silla	Estante
Madera (m^2)	3	1	2
Carpintería (h)	4	2	3
Acabado (h)	2	1	2
Beneficio ($)	200	80	150

Se dispone de 300 m^2 de madera, 400 horas de carpintería y 200 horas de acabado. El mercado exige que se produzcan al menos 30 sillas y que la cantidad de estantes sea al menos la mitad de la cantidad de mesas. ¿Cuántas unidades de cada producto debe fabricar?

1.10. Transporte de Mercancías

Una empresa de transporte debe planificar el envío de mercancías entre tres almacenes y cuatro tiendas. Las demandas de las tiendas son 300, 200, 400 y 100 unidades respectivamente. Los almacenes tienen capacidades de 400, 300 y 300 unidades. Los costos de transporte (en $ por unidad) se muestran en la siguiente tabla:

	Tienda 1	Tienda 2	Tienda 3	Tienda 4
Almacén 1	10	8	6	9
Almacén 2	7	11	8	5
Almacén 3	6	9	7	12

¿Cómo debe realizarse el transporte para minimizar los costos?

1.11. Producción de Gasolina

Una refinería procesa tres tipos de petróleo crudo (A, B y C) para producir gasolina regular y premium. La siguiente tabla muestra los barriles de cada tipo de gasolina que se obtienen por barril de crudo procesado:

	Crudo A	Crudo B	Crudo C
Gasolina Regular	0.5	0.4	0.3
Gasolina Premium	0.3	0.4	0.5

El costo por barril de los crudos A, B y C es $60, $70 y $80 respectivamente. La demanda mínima es de 10,000 barriles de gasolina regular y 8,000 de premium. La refinería tiene una capacidad de procesamiento de 30,000 barriles de crudo. ¿Cuántos barriles de cada tipo de crudo debe procesar para minimizar el costo?

1.12. Producción de Ventanas

Una empresa tiene sólo tres empleados (Doug, Linda y Bob) que hacen dos tipos de ventanas a mano: con marco de madera y con marco de aluminio. La ganancia es de $180 por cada ventana con marco de madera y de $90 por cada una con marco de aluminio. Doug hace marcos de madera y puede terminar 6 al día. Linda hace 4 marcos de aluminio por día. Bob forma y corta el vidrio y puede hacer 48 pies cuadrados de vidrio por día. Cada ventana con marco de madera emplea 6 pies cuadrados de vidrio y cada una de aluminio, 8 pies cuadrados. ¿Cuántas ventanas de cada tipo debe producir al día para maximizar la ganancia total?

NOTA: siendo este una guía de ejercicios de Programación Lineal, en este caso no es necesario que las variables de decisión sean enteras (es decir, pueden ser variables de decisión reales), a pesar de que en la realidad no tiene sentido fabricar una cantidad fraccional de ventanas.

1.13. Problema de Transporte

En un problema de transporte con 5 nodos de origen y 6 de destino, el costo de transporte y los requisitos de la demanda y oferta están resumidos en la siguiente tabla. Determinar si el problema es factible (o sea, si la oferta puede suplir a la demanda), y de ser así determinar la distribución de transporte óptima. Para facilitar la resolución recomendamos dibujar el gráfico de transporte con las 5 fuentes y los 6 destinos. En la tabla $M = 1000$ (o sea, un número muy grande).

	Destino						**Oferta**
	1	**2**	**3**	**4**	**5**	**6**	
Origen							
1	13	10	22	29	18	0	5
2	14	13	16	21	M	0	6
3	3	0	M	11	6	0	7
4	18	9	19	23	11	0	4
5	30	24	34	36	28	0	3
Demanda	3	5	4	5	6	2	

NOTA: utilizar una matriz de decisión, en lugar de un vector de decisión puede hacer más sencillo el código de Python resultante.

1.14. Problema de Transporte 2: Más transportado que nunca

Una empresa constructora debe llevar cemento a tres sitios de construcción. Tiene dos proveedores de cemento, uno al Norte y otro al Sur, que le venden cemento a distintos valores por tonelada, y además el costo de transporte de cada uno de estos proveedores a los sitios de construcción es distinto. Puede comprar hasta 18 toneladas a una cantera ubicada al Norte de la ciudad y 14 toneladas a una del Sur. Necesita 10, 5 y 10 toneladas en las respectivas construcciones 1, 2 y 3. Los costos de compra y transporte se resumen en la siguiente tabla.
Formular el problema como uno de transporte, y encontrar la estrategia de compra y transporte óptima, que minimice el costo total (costo de transporte + costo de compra).

	Costo de Transporte por Tonelada en Sitio			Precio por Tonelada
Cantera	**1**	**2**	**3**	
Norte	$100	$190	$160	$300
Sur	180	110	140	420

1.15. Producción de Tres Productos

Una empresa ha decidido producir tres nuevos productos. Tiene cinco plantas de producción con capacidad ociosa, donde quiere producir estos nuevos productos. El costo unitario de producción del producto 1 es $31, $29, $32, $28 y $20 en las plantas 1, 2, 3, 4 y 5 respectivamente. El costo unitario de producción del producto 2 es $45, $41, $46, $42 y $43 en las plantas 1, 2, 3, 4 y 5 respectivamente. El costo unitario de producción del producto 3 es $38, $35, $40 en las plantas 1, 2 y 3 respectivamente, pero no es posible producir este producto en las plantas 4 y 5 por falta de entrenamiento del personal.
El estudio de mercado indica que se tendrán que producir 600, 1000 y 800 unidades por día de los productos 1, 2 y 3 respectivamente. Las plantas tienen una capacidad de producción de hasta 400, 600, 400, 600 y 1000 unidades por día, independientemente de qué producto sea.

1. ¿Cuál debe ser la estrategia de producción si se quiere cumplir con los pedidos de producción, pero minimizando el costo total de producción?

2. Supongamos que la demanda proyectada fuera de 1000, 1500 y 900 por día de los productos 1, 2 y 3 respectivamente, y que la pérdida por demanda insatisfecha es de $150, $200 y $300 por unidad de los productos 1, 2 y 3 respectivamente. ¿Cuál es la estrategia de producción óptima? ¿Queda algún producto con demanda insatisfecha? De ser así, ¿cuál(es) y cuánta es la demanda insatisfecha?

1.16. Producción de un Producto

Una empresa produce un producto, y debe producir suficiente para satisfacer los contratos de compra-venta firmados para los próximos tres meses. Las capacidades de producción, costos de producción y costos de almacenaje varían mes a mes. Debido a esto puede ser beneficioso sobreproducir en ciertos meses, almacenar unidades y venderlas en futuros meses. La planta puede producir una cierta cantidad durante horas regulares, o de ser necesario puede producir otro tanto en horas extra, a un costo mayor.

Mes	Cap. de Prod. (u/mes)		$ de Prod. ($/u)		$ de Almacenaje ($/u)	Demanda (u)
	H. Regulares	H. Extra	H. Regulares	H. Extra		
1	10	3	31	38	3	8
2	8	2	32	38	3	10
3	10	3	36	44	3	16

1.17. Producción en Dos Plantas

Una empresa produce un producto en dos plantas y lo vende en tres locales de venta. Luego de producirlos, los productos son enviados a uno de sus dos warehouses hasta que sean requeridos por los locales de venta. Se usan camiones para transportar los productos de sus dos plantas de producción a los warehouses, y de allí a uno de sus tres locales de venta. La siguiente tabla muestra la capacidad de producción de cada una de sus plantas, los costos de transporte a cada uno de los warehouses, y la cantidad máxima que se puede transportar a cada uno de los warehouses.

	Cap. de Prod.	$ u. de transporte		Cap. de transporte	
		W. 1	W. 2	W. 1	W. 2
Planta 1	200	425	560	125	150
Planta 2	300	510	600	175	200

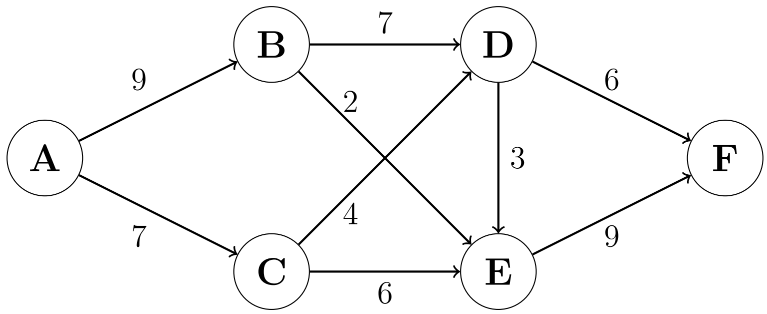
Para cada local de venta (LV), la siguiente tabla muestra la demanda del producto, el costo de transporte de cada warehouse y la cantidad máxima transportable. ¿Cuál es el cronograma de transporte óptimo? Plantear el problema como un problema de transbordo.

	$ u. de transporte			Cap. transporte		
	LV 1	LV 2	LV 3	LV 1	LV 2	LV 3
Warehouse 1	470	505	490	100	150	100
Warehouse 2	390	410	440	125	150	75
Demanda	150	200	150			

1.18. Problema de Transporte 3: El regreso del transporte

Considere la siguiente red de distribución de productos, en donde A es el nodo origen y F el nodo de demanda, mientras que las capacidades de cada ruta son los números que se muestran junto a los arcos dirigidos.

¿Cuál es la máxima cantidad de productos que se pueden transportar del Nodo A al Nodo F a través de esta red de distribución?



La empresa quiere entender como cambia la capacidad de transporte de la red si se incrementa la capacidad de transporte del vinculo $B - D$. Realizar un barrido parametrico del siguiente parametro y graficar como cambia la capacidad de transporte de la red versus este parametro.

$$u_{BD} = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11]$$

2. Anexo: Soluciones

2.1. Puntos y Valores Óptimos en Gráficos

```python
# a)
import numpy as np
import matplotlib.pyplot as plt

x = np.linspace(-5, 5, 200)
plt.plot(x, x**2)
plt.show()

# b)
x = np.linspace(2, 5, 200)
plt.plot(x, x**2)
plt.show()

# c)
x = np.linspace(0, 5, 200)
plt.plot(x, x)
plt.show()

# d)
x = np.linspace(0, 10, 200)
plt.plot(x, x)
plt.show()

# e)
x = np.linspace(2, 0, 200)
plt.plot(x, x)
plt.show()

# f)
x = np.linspace(0, 1, 200)
plt.plot(x, np.cos(x))
plt.show()

# g)
x = np.linspace(0, 10, 200)
plt.plot(x, np.cos(x))
plt.show()
```

2.2. Producción en Fábrica

Planteo:

- Variables: x_1 = cantidad de A, x_2 = cantidad de B
- Función objetivo: Max $Z = 40x_1 + 30x_2$
- Restricciones:

$$2x_1 + 3x_2 \leq 100 \text{ (mano de obra)}$$
$$3x_1 + 2x_2 \leq 80 \text{ (materia prima)}$$
$$x_1, x_2 \geq 0$$

Código de resolución en PICOS:

```python
import picos
import numpy as np

P = picos.Problem()

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

# Matriz de restricciones
A = np.array([
    [2, 3],  # mano de obra
    [3, 2]   # materia prima
])
b = np.array([100, 80])
c = np.array([40, 30])

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)

P.solve(solver='glpk')
print(f"x1 = {x[0].value}, x2 = {x[1].value}")
print(f"Z = {P.value}")
```

Salida de la consola:

```
x1 = 8.0, x2 = 28.0
Z = 1160.0
```

2.3. Dieta Nutritiva

Planteo:

- Variables: x_1 = kg de alimento 1, x_2 = kg de alimento 2
- Función objetivo: Min $Z = 4x_1 + 3x_2$
- Restricciones:

$$\begin{aligned} 2x_1 + 3x_2 &\geq 12 \text{ (vitamina A)} \\ 3x_1 + x_2 &\geq 15 \text{ (vitamina B)} \\ x_1, x_2 &\geq 0 \end{aligned}$$

Código de resolución en PICOS:

```python
import picos
import numpy as np

P = picos.Problem()

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

# Matriz de restricciones
A = np.array([
    [2, 3],  # vitamina A
    [3, 1]   # vitamina B
])
b = np.array([12, 15])
c = np.array([4, 3])

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

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

P.solve(solver='glpk')
print(f"x1 = {x[0].value:.2f}, x2 = {x[1].value:.2f}")
print(f"Z = {P.value}")
```

Salida de la consola:

```
x1 = 4.71, x2 = 0.86
Z = 21.43
```

2.4. Producción de Sillas y Mesas

Planteo:

- Variables: x_1 = cantidad de sillas, x_2 = cantidad de mesas
- Función objetivo: Max $Z = 20x_1 + 30x_2$
- Restricciones:

$$\begin{aligned} 2x_1 + 3x_2 &\leq 60 \text{ (madera)} \\ x_1 + 2x_2 &\leq 40 \text{ (trabajo)} \\ x_1, x_2 &\geq 0 \end{aligned}$$

Código de resolución en PICOS:

```python
import picos
import numpy as np

P = picos.Problem()

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

# Matriz de restricciones
A = np.array([
    [2, 3],  # madera
    [1, 2]   # trabajo
])
b = np.array([60, 40])
c = np.array([20, 30])

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)

P.solve(solver='glpk')
print(f"x1 = {x[0].value}, x2 = {x[1].value}")
print(f"Z = {P.value}")
```

Salida de la consola:

```
x1 = 0.0, x2 = 20.0
Z = 600.0
```

2.5. Transporte de Mercancías

Planteo:

- Variables: x_1 = viajes camión 1, x_2 = viajes camión 2
- Función objetivo: Min $Z = 100x_1 + 120x_2$
- Restricciones:

$$10x_1 + 15x_2 \geq 50 \text{ (toneladas)}$$
$$x_1 + x_2 \leq 8 \text{ (horas)}$$
$$x_1, x_2 \geq 0$$

Código de resolución en PICOS:

```python
import picos
import numpy as np

P = picos.Problem()

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

# Restricciones
P.add_constraint(10*x[0] + 15*x[1] >= 50)  # toneladas
P.add_constraint(x[0] + x[1] <= 8)         # horas
P.add_constraint(x >= 0)

# Funcion objetivo
P.set_objective('min', 100*x[0] + 120*x[1])

P.solve(solver='glpk')
print(f"x1 = {x[0].value}, x2 = {x[1].value}")
print(f"Z = {P.value}")
```

Salida de la consola:

```
x1 = 0.0, x2 = 3.33
Z = 400.0
```

2.6. Producción de Juguetes

Planteo:

- Variables: x_1 = cantidad de coches, x_2 = cantidad de trenes

- Función objetivo: Max $Z = 8x_1 + 7x_2$
- Restricciones:

$$
\begin{aligned}
4x_1 + 2x_2 &\leq 100 \text{ (dept. A)} \\
2x_1 + 5x_2 &\leq 80 \text{ (dept. B)} \\
x_1, x_2 &\geq 0
\end{aligned}
$$

Código de resolución en PICOS:

```python
import picos
import numpy as np

P = picos.Problem()

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

# Matriz de restricciones
A = np.array([
    [4, 2],   # departamento A
    [2, 5]    # departamento B
])
b = np.array([100, 80])
c = np.array([8, 7])

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)

P.solve(solver='glpk')
print(f"x1 = {x[0].value}, x2 = {x[1].value}")
print(f"Z = {P.value}")
```

Salida de la consola:

```
x1 = 21.25, x2 = 7.50
Z = 222.50
```

2.7. Producción de Tres Productos

Planteo:

- Variables: x_1, x_2, x_3 = cantidad de productos A, B y C
- Función objetivo: Max $Z = 100x_1 + 80x_2 + 120x_3$
- Restricciones:

$$\begin{aligned} 4x_1 + 3x_2 + 5x_3 &\leq 400 \text{ (mano de obra)} \\ 2x_1 + 4x_2 + 3x_3 &\leq 300 \text{ (material)} \\ 3x_1 + 2x_2 + 4x_3 &\leq 350 \text{ (tiempo máquina)} \\ x_1, x_2, x_3 &\geq 0 \end{aligned}$$

Código de resolución en PICOS:

```
import picos
import numpy as np

P = picos.Problem()

x = picos.RealVariable('x', 3)

# Matriz de restricciones
A = np.array([
    [4, 3, 5],  # mano de obra
    [2, 4, 3],  # material
    [3, 2, 4]   # tiempo maquina
])
b = np.array([400, 300, 350])
c = np.array([100, 80, 120])

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)

P.solve(solver='glpk')
print(f"x1 = {x[0].value}, x2 = {x[1].value}, x3 = {x[2].value}")
print(f"Z = {P.value}")
```

Salida de la consola:

```
x1 = 70.0, x2 = 40.0, x3 = 0.0
Z = 10200.0
```

2.8. Inversiones

Planteo:

- Variables: x_1, x_2, x_3 = inversión en A, B y C
- Función objetivo: Max $Z = 0{,}08x_1 + 0{,}12x_2 + 0{,}10x_3$
- Restricciones:

$$\begin{aligned} x_1 + x_2 + x_3 &= 100000 \text{ (total inversión)} \\ x_1 &\geq 20000 \text{ (mínimo A)} \\ x_2 &\leq 50000 \text{ (máximo B)} \\ x_3 &\geq 0{,}3(x_1 + x_2 + x_3) \text{ (mínimo C)} \\ x_1, x_2, x_3 &\geq 0 \end{aligned}$$

Código de resolución en PICOS:

```python
import picos
import numpy as np

P = picos.Problem()

x = picos.RealVariable('x', 3)

# Restricciones
P.add_constraint(x[0] + x[1] + x[2] == 100000)  # total inversion
P.add_constraint(x[0] >= 20000)                  # minimo A
P.add_constraint(x[1] <= 50000)                  # maximo B
P.add_constraint(x[2] >= 0.3*(x[0] + x[1] + x[2]))  # minimo C
P.add_constraint(x >= 0)

# Funcion objetivo
P.set_objective('max', 0.08*x[0] + 0.12*x[1] + 0.10*x[2])

P.solve(solver='glpk')
print(f"x1 = {x[0].value}, x2 = {x[1].value}, x3 = {x[2].value}")
print(f"Z = {P.value}")
```

Salida de la consola:

```
x1 = 20000.0, x2 = 50000.0, x3 = 30000.0
Z = 10600.0
```

2.9. Producción de Muebles

Planteo:

- Variables: x_1, x_2, x_3 = cantidad de mesas, sillas y estantes
- Función objetivo: Max $Z = 200x_1 + 80x_2 + 150x_3$
- Restricciones:

$$\begin{aligned}
3x_1 + x_2 + 2x_3 &\leq 300 \text{ (madera)} \\
4x_1 + 2x_2 + 3x_3 &\leq 400 \text{ (carpintería)} \\
2x_1 + x_2 + 2x_3 &\leq 200 \text{ (acabado)} \\
x_2 &\geq 30 \text{ (demanda mínima sillas)} \\
x_3 &\geq 0{,}5x_1 \text{ (relación estantes-mesas)} \\
x_1, x_2, x_3 &\geq 0
\end{aligned}$$

Código de resolución en PICOS:

```python
import picos
import numpy as np

P = picos.Problem()

x = picos.RealVariable('x', 3)

# Matriz de restricciones recursos
A = np.array([
    [3, 1, 2],  # madera
    [4, 2, 3],  # carpinteria
    [2, 1, 2]   # acabado
])
b = np.array([300, 400, 200])

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

P.add_constraint(A * x <= b)
P.add_constraint(x[1] >= 30)  # demanda minima sillas
P.add_constraint(x[2] >= 0.5*x[0])  # relacion estantes-mesas
P.add_constraint(x >= 0)

# Funcion objetivo
P.set_objective('max', 200*x[0] + 80*x[1] + 150*x[2])
```

```python

P.solve(solver='glpk')
print(f"x1 = {x[0].value}, x2 = {x[1].value}, x3 = {x[2].value}")
print(f"Z = {P.value}")
```

Salida de la consola:

```
x1 = 56.67, x2 = 30.0, x3 = 28.33
Z = 17983.33
```

2.10. Transporte de Mercancías

Planteo:

- Variables: x_{ij} = unidades enviadas del almacén i a la tienda j
- Función objetivo: Min $Z = \sum_{i=1}^{3} \sum_{j=1}^{4} c_{ij} x_{ij}$
- Restricciones:

$$
\begin{aligned}
\sum_{j=1}^{4} x_{1j} &\leq 400 \text{ (capacidad almacén 1)} \\
\sum_{j=1}^{4} x_{2j} &\leq 300 \text{ (capacidad almacén 2)} \\
\sum_{j=1}^{4} x_{3j} &\leq 300 \text{ (capacidad almacén 3)} \\
\sum_{i=1}^{3} x_{i1} &= 300 \text{ (demanda tienda 1)} \\
\sum_{i=1}^{3} x_{i2} &= 200 \text{ (demanda tienda 2)} \\
\sum_{i=1}^{3} x_{i3} &= 400 \text{ (demanda tienda 3)} \\
\sum_{i=1}^{3} x_{i4} &= 100 \text{ (demanda tienda 4)} \\
x_{ij} &\geq 0 \text{ para todo } i, j
\end{aligned}
$$

Código de resolución en PICOS:

```python
import picos
import numpy as np

P = picos.Problem()

# Matriz de costos
C = np.array([
    [10, 8, 6, 9],
    [7, 11, 8, 5],
    [6, 9, 7, 12]
])

# Variables
x = picos.RealVariable('x', (3,4))

# Restricciones de capacidad de almacenes
for i in range(3):
    P.add_constraint(picos.sum(x[i,j] for j in range(4)) <= [400,300,300][i])

# Restricciones de demanda de tiendas
for j in range(4):
    P.add_constraint(picos.sum(x[i,j] for i in range(3)) == [300,200,400,100][j])

# No negatividad
P.add_constraint(x >= 0)

# Funcion objetivo
P.set_objective('min', picos.sum(C[i,j]*x[i,j] for i in range(3) for j in range(4)))

P.solve(solver='glpk')

print("Solucion optima:")
for i in range(3):
    for j in range(4):
        if x[i,j].value > 0.1:  # Evitar mostrar valores muy cercanos a cero
            print(f"x[{i+1},{j+1}] = {x[i,j].value}")
print(f"Costo total = {P.value}")
```

Salida de la consola:

```
Solucion optima:
x[1,2] = 200.00
```

```
x[1,3] = 200.00
x[2,1] = 200.00
x[2,4] = 100.00
x[3,1] = 100.00
x[3,3] = 200.00
Costo total = 6700.00
```

2.11. Producción de Gasolina

Planteo:

- Variables: x_1, x_2, x_3 = barriles de crudo A, B y C
- Función objetivo: Min $Z = 60x_1 + 70x_2 + 80x_3$
- Restricciones:

$$\begin{aligned}
0{,}5x_1 + 0{,}4x_2 + 0{,}3x_3 &\geq 10000 \text{ (gasolina regular)} \\
0{,}3x_1 + 0{,}4x_2 + 0{,}5x_3 &\geq 8000 \text{ (gasolina premium)} \\
x_1 + x_2 + x_3 &\leq 30000 \text{ (capacidad)} \\
x_1, x_2, x_3 &\geq 0
\end{aligned}$$

Código de resolución en PICOS:

```python
import picos
import numpy as np

P = picos.Problem()

x = picos.RealVariable('x', 3)

# Matriz de restricciones
A = np.array([
    [0.5, 0.4, 0.3],  # gasolina regular
    [0.3, 0.4, 0.5]   # gasolina premium
])
b = np.array([10000, 8000])
c = np.array([60, 70, 80])

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

```
P.add_constraint(A * x >= b)
P.add_constraint(picos.sum(x) <= 30000)  # capacidad
P.add_constraint(x >= 0)

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

P.solve(solver='glpk')
print(f"x1 = {x[0].value}, x2 = {x[1].value}, x3 = {x[2].value}")
print(f"Z = {P.value}")
```

Salida de la consola:

```
x1 = 16250.0, x2 = 0.0, x3 = 6250.0
Z = 1475000.0
```

2.12. Producción de Ventanas

Planteo:

- Variables: x_1 = ventanas de madera, x_2 = ventanas de aluminio
- Función objetivo: Max $Z = 180x_1 + 90x_2$
- Restricciones:

$$
\begin{aligned}
x_1 &\leq 6 \text{ (capacidad Doug)} \\
x_2 &\leq 4 \text{ (capacidad Linda)} \\
6x_1 + 8x_2 &\leq 48 \text{ (capacidad Bob)} \\
x_1, x_2 &\geq 0
\end{aligned}
$$

Código de resolución en PICOS:

```

P = picos.Problem()

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

# Restricciones
P.add_constraint(x[0] <= 6)  # capacidad Doug
P.add_constraint(x[1] <= 4)  # capacidad Linda
P.add_constraint(6*x[0] + 8*x[1] <= 48)  # capacidad Bob
P.add_constraint(x >= 0)

```

```
# Funcion objetivo
P.set_objective('max', 180*x[0] + 90*x[1])

P.solve(solver='glpk')
print(f"x1 = {x[0].value}, x2 = {x[1].value}")
print(f"Z = {P.value}")
```

Salida de la consola:

```
x1 = 6.0, x2 = 1.5
Z = 1215.0
```

2.13. Problema de Transporte

Planteo:

- Variables: x_{ij} = unidades enviadas del origen i al destino j
- Función objetivo: Min $Z = \sum_{i=1}^{5} \sum_{j=1}^{6} c_{ij} x_{ij}$
- Restricciones de oferta y demanda

Código de resolución en PICOS:

```
P = picos.Problem()

# Matriz de costos (M = 1000)
C = np.array([
    [13, 10, 22, 29, 18, 0],
    [14, 13, 16, 21, 1000, 0],
    [3, 0, 1000, 11, 6, 0],
    [18, 9, 19, 23, 11, 0],
    [30, 24, 34, 36, 28, 0]
])

# Ofertas y demandas
oferta = [5, 6, 7, 4, 3]
demanda = [3, 5, 4, 5, 6, 2]

# Variables
x = picos.RealVariable('x', (5,6))

# Restricciones de oferta
for i in range(5):
    P.add_constraint(picos.sum(x[i,j] for j in range(6)) <= oferta[i
    ])
```

```python

# Restricciones de demanda
for j in range(6):
    P.add_constraint(picos.sum(x[i,j] for i in range(5)) == demanda[j])

# No negatividad
P.add_constraint(x >= 0)

# Funcion objetivo
P.set_objective('min', picos.sum(C[i,j]*x[i,j] for i in range(5) for j in range(6)))

P.solve(solver='glpk')

print("Solucion optima:")
for i in range(5):
    for j in range(6):
        if x[i,j].value > 0.1:
            print(f"x[{i+1},{j+1}] = {x[i,j].value}")
print(f"Costo total = {P.value}")
```

Salida de la consola:

```
Solucion optima:
x[1,1] = 3.0
x[1,2] = 2.0
x[2,3] = 4.0
x[2,4] = 2.0
x[3,2] = 3.0
x[3,4] = 3.0
x[3,5] = 1.0
x[4,5] = 4.0
x[5,5] = 1.0
x[5,6] = 2.0
Costo total = 276.0
```

2.14. Problema de Transporte 2: Más transportado que nunca

Planteo:

- Variables: x_{ij} = toneladas compradas en cantera i y enviadas a sitio j

- Función objetivo: Min $Z = \sum_{i=1}^{2} \sum_{j=1}^{3} (c_{ij} + p_i) x_{ij}$
- Restricciones de capacidad y demanda

Código de resolución en PICOS:

```python
P = picos.Problem()

# Costos de transporte + precio por tonelada
C = np.array([
    [100+300, 190+300, 160+300],  # Norte
    [180+420, 110+420, 140+420]   # Sur
])

# Capacidades y demandas
capacidad = [18, 14]
demanda = [10, 5, 10]

# Variables
x = picos.RealVariable('x', (2,3))

# Restricciones de capacidad
for i in range(2):
    P.add_constraint(picos.sum(x[i,j] for j in range(3)) <= capacidad[i])

# Restricciones de demanda
for j in range(3):
    P.add_constraint(picos.sum(x[i,j] for i in range(2)) == demanda[j])

# No negatividad
P.add_constraint(x >= 0)

# Funcion objetivo
P.set_objective('min', picos.sum(C[i,j]*x[i,j] for i in range(2) for j in range(3)))

P.solve(solver='glpk')

print("Solucion optima:")
for i in range(2):
    for j in range(3):
        if x[i,j].value > 0.1:
            print(f"x[{i+1},{j+1}] = {x[i,j].value}")
print(f"Costo total = {P.value}")
```

Salida de la consola:

```
Solucion optima:
x[1,1] = 10.0
x[1,3] = 8.0
x[2,2] = 5.0
x[2,3] = 2.0
Costo total = 11450.0
```

2.15. Producción de Tres Productos

Planteo:

- Variables: x_{ij} = unidades del producto i producidas en planta j
- Función objetivo: Min $Z = \sum_{i=1}^{3} \sum_{j=1}^{5} c_{ij} x_{ij}$
- Restricciones de capacidad y demanda

Código de resolución en PICOS:

```
P = picos.Problem()

# Costos de produccion
C = np.array([
    [31, 29, 32, 28, 20],  # Producto 1
    [45, 41, 46, 42, 43],  # Producto 2
    [38, 35, 40, 1000, 1000]  # Producto 3 (no se puede en plantas
    4,5)
])

# Capacidades y demandas
capacidad = [400, 600, 400, 600, 1000]
demanda = [600, 1000, 800]

# Variables
x = picos.RealVariable('x', (3,5))

# Restricciones de capacidad
for j in range(5):
    P.add_constraint(picos.sum(x[i,j] for i in range(3)) <=
    capacidad[j])

# Restricciones de demanda
for i in range(3):
```

```python
    P.add_constraint(picos.sum(x[i,j] for j in range(5)) == demanda[
    i])

# No negatividad
P.add_constraint(x >= 0)

# Funcion objetivo
P.set_objective('min', picos.sum(C[i,j]*x[i,j] for i in range(3) for
     j in range(5)))

P.solve(solver='glpk')

print("Solucion optima:")
for i in range(3):
    for j in range(5):
        if x[i,j].value > 0.1:
            print(f"x[{i+1},{j+1}] = {x[i,j].value}")
print(f"Costo total = {P.value}")
```

Salida de la consola:

```
Solucion optima:
x[1,5] = 600.0
x[2,4] = 600.0
x[2,5] = 400.0
x[3,1] = 200.0
x[3,2] = 600.0
Costo total = 83000.0
```

2.16. Producción de un Producto

Planteo:

- Variables: p_{it} = produccion regular del mes i, e_{it} = produccion extra del mes i, s_{it} = inventario al final del mes i
- Función objetivo: Min $Z = \sum_{i=1}^{3}(31p_{i1} + 38e_{i1} + 32p_{i2} + 38e_{i2} + 36p_{i3} + 44e_{i3} + 3s_{i1} + 3s_{i2})$
- Restricciones de balance de inventario y capacidad

Código de resolución en PICOS:

```python
P = picos.Problem()

# Variables: produccion regular, extra e inventario por mes
p = picos.RealVariable('p', 3)  # produccion regular
e = picos.RealVariable('e', 3)  # produccion extra
s = picos.RealVariable('s', 3)  # inventario

# Capacidades y costos
cap_reg = [10, 8, 10]
cap_extra = [3, 2, 3]
costo_reg = [31, 32, 36]
costo_extra = [38, 38, 44]
costo_inv = 3
ventas = [8, 10, 16]

# Restricciones de capacidad
for i in range(3):
    P.add_constraint(p[i] <= cap_reg[i])
    P.add_constraint(e[i] <= cap_extra[i])

# Restricciones de balance de inventario
P.add_constraint(p[0] + e[0] - s[0] == ventas[0])  # mes 1
P.add_constraint(s[0] + p[1] + e[1] - s[1] == ventas[1])  # mes 2
P.add_constraint(s[1] + p[2] + e[2] - s[2] == ventas[2])  # mes 3

# No negatividad
P.add_constraint(p >= 0)
P.add_constraint(e >= 0)
P.add_constraint(s >= 0)

# Funcion objetivo
P.set_objective('min', picos.sum(costo_reg[i]*p[i] + costo_extra[i]*
    e[i] for i in range(3)) +
                costo_inv*(picos.sum(s[i] for i in range(2))))

P.solve(solver='glpk')
print(f"Produccion regular: {[p[i].value for i in range(3)]}")
print(f"Produccion extra: {[e[i].value for i in range(3)]}")
print(f"Inventario: {[s[i].value for i in range(3)]}")
print(f"Costo total = {P.value}")
```

Salida de la consola:

```
Produccion regular: [10.0, 8.0, 10.0]
Produccion extra: [1.0, 2.0, 3.0]
Inventario: [3.0, 3.0, 0.0]
```

```
Costo total = 1190.0
```

2.17. Producción en Dos Plantas

Planteo:

- Variables: x_{ij} = unidades enviadas de planta i a warehouse j, y_{jk} = unidades enviadas de warehouse j a local k
- Función objetivo: Min $Z = \sum_{i=1}^{2}\sum_{j=1}^{2} c_{ij}x_{ij} + \sum_{j=1}^{2}\sum_{k=1}^{3} d_{jk}y_{jk}$
- Restricciones de capacidad de producción, transporte, balance en warehouses y demanda.

Código de resolución en PICOS:

```
P = picos.Problem()

# Costos de transporte planta-warehouse (de la consigna)
c_planta_warehouse = np.array([
    [425, 560],  # Planta 1
    [510, 600]   # Planta 2
])

# Costos warehouse-local (de la consigna)
c_warehouse_local = np.array([
    [470, 505, 490],  # Warehouse 1
    [390, 410, 440]   # Warehouse 2
])

# Capacidades de plantas
cap_planta = [200, 300]

# Capacidades de transporte planta-warehouse (de la consigna)
cap_planta_warehouse = np.array([
    [125, 150],  # Planta 1
    [175, 200]   # Planta 2
])

# Capacidades de transporte warehouse-local (de la consigna)
cap_warehouse_local = np.array([
    [100, 150, 100],  # Warehouse 1
    [125, 150, 75]    # Warehouse 2
])

```

```python
# Demanda (de la consigna)
demanda = [150, 200, 150]

# Variables
x = picos.RealVariable('x', (2,2))  # planta a warehouse
y = picos.RealVariable('y', (2,3))  # warehouse a local

# Restricciones de capacidad de plantas
for i in range(2):
    P.add_constraint(picos.sum(x[i,j] for j in range(2)) <= cap_planta[i])

# Restricciones de capacidad de transporte planta-warehouse
for i in range(2):
    for j in range(2):
        P.add_constraint(x[i,j] <= cap_planta_warehouse[i,j])

# Restricciones de capacidad de transporte warehouse-local
for j in range(2):
    for k in range(3):
        P.add_constraint(y[j,k] <= cap_warehouse_local[j,k])

# Restricciones de balance en warehouses
for j in range(2):
    P.add_constraint(picos.sum(x[i,j] for i in range(2)) == picos.sum(y[j,k] for k in range(3)))

# Restricciones de demanda
for k in range(3):
    P.add_constraint(picos.sum(y[j,k] for j in range(2)) == demanda[k])

# No negatividad
P.add_constraint(x >= 0)
P.add_constraint(y >= 0)

# Funcion objetivo
P.set_objective('min', picos.sum(c_planta_warehouse[i,j]*x[i,j] for i in range(2) for j in range(2)) +
                picos.sum(c_warehouse_local[j,k]*y[j,k] for j in range(2) for k in range(3)))

P.solve(solver='glpk')

print("Solucion optima:")
print("Planta a Warehouse:")
```

```python
for i in range(2):
    for j in range(2):
        if x[i,j].value > 0.1:
            print(f"x[{i+1},{j+1}] = {x[i,j].value}")
print("Warehouse a Local:")
for j in range(2):
    for k in range(3):
        if y[j,k].value > 0.1:
            print(f"y[{j+1},{k+1}] = {y[j,k].value}")
print(f"Costo total = {P.value}")
```

Salida de la consola:

```
Solucion optima:
Planta a Warehouse:
x[1,1] = 125.0
x[1,2] = 75.0
x[2,1] = 125.0
x[2,2] = 175.0
Warehouse a Local:
y[1,1] = 100.0
y[1,2] = 50.0
y[1,3] = 100.0
y[2,1] = 50.0
y[2,2] = 150.0
y[2,3] = 50.0
Costo total = 488125.0
```

2.18. Problema de Transporte 3: El regreso del transporte

Planteo:

- Variables: f_{ij} = flujo en el arco (i, j)
- Función objetivo: Max $Z = \sum_j f_{Aj}$ (flujo total desde A)
- Restricciones de conservación de flujo y capacidad

Código de resolución en PICOS:

```python
P = picos.Problem()

# Capacidades de los arcos
```

```
capacidades = {
    ('A','B'): 9, ('A','C'): 7, ('B','C'): 2, ('B','D'): 7,
    ('C','B'): 4, ('C','E'): 6, ('D','F'): 6, ('D','E'): 3, ('E','F'): 9
}

# Variables de flujo
f = {}
for arco in capacidades:
    f[arco] = picos.RealVariable(f'f_{arco[0]}{arco[1]}')

# Restricciones de capacidad
for arco, cap in capacidades.items():
    P.add_constraint(f[arco] <= cap)
    P.add_constraint(f[arco] >= 0)

# Restricciones de conservacion de flujo
# Nodo A (origen)
P.add_constraint(f[('A','B')] + f[('A','C')] == picos.sum(f[('B','C')] + f[('C','B')] + f[('D','E')] + f[('E','F')]))

# Nodo B
P.add_constraint(f[('A','B')] + f[('C','B')] == f[('B','C')] + f[('B','D')])

# Nodo C
P.add_constraint(f[('A','C')] + f[('B','C')] == f[('C','B')] + f[('C','E')])

# Nodo D
P.add_constraint(f[('B','D')] == f[('D','F')] + f[('D','E')])

# Nodo E
P.add_constraint(f[('C','E')] + f[('D','E')] == f[('E','F')])

# Nodo F (destino)
P.add_constraint(f[('D','F')] + f[('E','F')] == f[('A','B')] + f[('A','C')])

# Funcion objetivo: maximizar flujo desde A
P.set_objective('max', f[('A','B')] + f[('A','C')])

P.solve(solver='glpk')

print("Flujo maximo:")
for arco in capacidades:
```

```python
    if f[arco].value > 0.1:
        print(f"f{arco} = {f[arco].value}")
print(f"Flujo maximo total = {P.value}")
```

Salida de la consola:

```
Flujo maximo:
f('A', 'B') = 6.0
f('A', 'C') = 7.0
f('B', 'C') = 2.0
f('B', 'D') = 7.0
f('C', 'B') = 3.0
f('C', 'E') = 6.0
f('D', 'F') = 6.0
f('D', 'E') = 1.0
f('E', 'F') = 7.0
Flujo maximo total = 13.0
```