Introducción a la resolución de Problemas de Programación Lineal (PPL) con SageMath

Como sabemos, Sage Jupyter (Jupyter Notebook para SAGE), es un software matemático de propósito general, aplicable a un amplio abanico de ramas de la matemática, la Programación Matemática entre ellas. Vamos a ver una breve introducción sobre la resolución de Problemas de Programación Lineal con SAGE.

- SAGE no resuelve directamente los Problemas de Programación Lineal. En su lugar, llama a librerías específicas externas adecuadas (ó Solvers) para resolver el problema.
- Para problemas de Programación Lineal y Programación Entera (clase MILP, Mixed Integer Linear Problem), SAGE usa por defecto el Solver GLPK. Para los interesados, la lista de solvers para MILP se puede consultar en: http://doc.sagemath.org/html/en/thematic_tutorials/linear_programming.html (actualizar enlace de ser necesario).
- Dicho enlace contiene instrucciones básicas sobre los comandos, sintaxis y opciones disponibles.

Modelar el Problema

Antes de resolver el problema con SAGE, tenemos que modelarlo. En sentido general, modelar un PPL es el proceso trasladar el problema de su contexto original a una formulación matemática complacta, que podamos implementar y resolver con algún software adecuado.

Básicamente, los elementos que intervienen en un PPL son:

- Los **datos**, y parámetros del problema. Conviene organizar los datos (mediante tablas, listas, etc.)
- Las **Variables de decisión:** Hay que definirlas, identificando qué representan, y su naturaleza (continuas, enteras, binarias).
- La **función objetivo**.
- Las **restricciones**. A continuación, vamos a modelar (esto es, formular) y resolver algunos ejemplos con SAGE.

Ejemplo 1. Problema de la Dieta

Es un problema clásico, al ser uno de los primeros problemas resueltos mediante Programación Lineal (en 1947). El problema fué motivado por el propósito del ejército americano de garantizar a sus tropas unos requisitos nutricionales al mínimo coste.

Enunciado. La inspección de los menús diarios que se sirven en el comedor de unas instalaciones deportivas presenta deficiencias en vitaminas A, C, y en la cantidad de fibra aconsejable para garantizar un equilibrio nutricional. La empresa que elabora los menús propone añadir al menú diario un plato de guarnición, compuesto de zanahorias, repollo blanco, y un mixto de de verduras y frutos secos (que llamaremos ***mixto***), de forma que con dicho plato adicional se solventen las definiencias nutricionales señaladas. Los datos sobre la cantidad mínima de nutrientes (vitaminas y fibra) requeridos por plato, su contenido en los alimentos (zanahorias, repollo y mixto), y el coste unitario de los mismos se muestra en la siguiente tabla:

Nutrientes	Zanahoria	Repollo	Mixto	Requisito por plato
Vit. A (mg./kg)	35	0.5	0.5	0.5 mg.
Vit. C (mg./kg)	60	300	10	15 mg
Fibra (g./kg)	30	20	10	4 g
Coste (euros/kg)	0.75	0.5	0.15	

La empresa quiere saber cómo elaborar el plato adicional al coste mínimo, de forma que se satisfagan los requisitos nutricionales exigidos.

Resolución

- **Datos.** Como los datos están ya organizados, primero vamos a itentificarlos mediante conjuntos de índices, por ejemplo:
  - $\{A_1,A_2,A_3\}$, (ó $I=\{1,2,3\}$), son los alimentos: zanahoria, repollo, y mixto.
  - Mientras que $\{N_1,N_2,N_3\}$, (ó $J=\{1,2,3\}$) son los nutrientes: Vitamina A, Vitamina C, y Fibra.
- **Variables de decisión.** x_i, $i=1,2,3$, representa la cantidad de alimento i (en kg.), que debe incluirse en cada plato. Obviamente, $x_i\geq 0$, para todo $i=1,2,3$.
- **Restricciones.** Se debe cumplir el requisito de mínimos para cada nutriente, esto es: $35x_1+0.5x_2+0.5x_3\geq 0.5$, para la Vitamina A, etc. (lo mismo para los demás nutrientes).
- **Función objetivo.** Expresa el coste del plato: $\min Z(x_1,x_2,x_3):=0.75x_1+0.5x_2+1.15x_3$.

El problema queda formulado como sigue:

$$\begin{aligned} \min Z(x) &:= 0.75x_1+0.5x_2+0.15x_3 \\ \text{s.t.} \quad & 35x_1+0.5x_2+0.5x_3\geq 0.5 \\ & 60x_1+300x_2+10x_3\geq 15 \\ & 30x_1+20x_2+10x_3\geq 4 \\ & x_1\geq 0, x_2\geq 0, x_3\geq 0 \end{aligned}$$

In [28]:

```
p=MixedIntegerLinearProgram(maximization=False)
x=p.new_variable(real=True, nonnegative=True)
p.add_constraint(35*x[1] + 0.5*x[2]+0.5*x[3] >= 0.5)
p.add_constraint(60*x[1]+300*x[2]+10*x[3]>=15)
p.add_constraint(30*x[1]+20*x[2]+10*x[3]>=4)
p.set_objective(0.75*x[1]+0.5*x[2]+0.15*x[3])
print ('Valor objetivo=', p.solve())
print ('Solución=', p.get_values(x))
```

```
Valor objetivo= 0.07051091005854176
Solución= {1: 0.00952634379989356, 2: 0.03826503459286855, 3: 0.2948908994145822}
```

El valor de la función objetivo en la solución óptima es $Z^*=0.0705109100585\simeq 0.07$ euros.
Y en la solución óptima, las cantidades de cada alimento (en kg.) son: $x^*_1=0.0095263\ldots$ kg., $x^*_2=0.038265\ldots$ kg., y $x^*_3=0.2948908\ldots$ kg.
Expresado en gramos, cada plato debe contener 9.526 g. de zanahorias, 38.26 g. de repollo, y 294.89 g. de mixto.

Si expresamos el problema de forma matricial:

$$\begin{array}{ll} \min & Z(x):=c^t x \\ \text{s.t.} & Ax\geq b \\ & x\geq 0 \end{array}$$

podemos implementarlo como sigue:

In [29]:

```
costes=[0.75, 0.5, 0.15]
A=matrix([ [35, 0.5, 0.5], [60, 300, 10], [30, 20, 10]  ])
b=[0.5, 15, 4]
p1=MixedIntegerLinearProgram(maximization=False)
x=p1.new_variable(real=True, nonnegative=True)
for i in range (len(b)):
    p1.add_constraint(p1.sum(A[i, j]*x[j] for j in range(len(costes)))
>=b[i])
p1.set_objective(p1.sum(costes[j]*x[j] for j in range(len(costes))))
print ('Valor objetivo=', p1.solve())
print ('Solución=', p1.get_values(x))
```

```
Valor objetivo= 0.07051091005854176
Solución= {0: 0.00952634379989356, 1: 0.03826503459286855, 2: 0.294890
8994145822}
```

(Hay que tener en cuenta que SAGE comienza la indexación de las variables en 0).

O bien para visualizar mejor los resultados:

In [30]:

```
print ('Valor objetivo=', p1.solve())
sol=p1.get_values(x)
for key in sorted(sol.keys()):
    print ('x',key+1, '=', sol[key])
```

```
Valor objetivo= 0.07051091005854176
x 1 = 0.00952634379989356
x 2 = 0.03826503459286855
x 3 = 0.2948908994145822
```

Ejemplo 2. Problema 6 del Boletín de problemas

Una empresa fabrica tres productos, que denotaremos por 1, 2 y 3. Cada producto requiere de un tiempo de producción en tres departamentos D_j, $j=1,2,3$. La siguienta tabla muestra las horas que cada unidad del producto i consume en el departamento j:

	P_1	P_2	P_3
D_1	3	4	2
D_2	2	1	2
D_3	1	3	3

Los departamentos tienen una producción horaria total de 600, 400, y 300 horas, respectivamente. Si cada unidad del producto $i=1,2,3$, genera un beneficio de 2, 4, y 2.5 euros, respectivamente, encontrar la combinación óptima de producción.

Resolución

Es un problema del tipo de distribución óptima de bienes y recursos, bien para maximizar beneficios, ó para minimizar costes, etc. Como en este caso los datos están ya organizados, vamos a definir las variables de decisión, teniendo en cuenta que $I=\{1,2,3\}$ identifica a los productos, y $J=\{1,2,3\}$ a los departamentos:

- Variables de decisión: x_i=Unidades del producto i a fabricar, con $i=1,2,3$. Claramente $x_i \geq 0$, para $i=1,2,3$, y todas tienen que ser variables **enteras**, por la descripción y contexto del problema. Estamos por tanto con un **Problema de Programación Lineal Entera** (en donde las variables de decisión deben ser enteras).
- Restricciones: El tiempo total disponible de cada departamento no se puede rebasar.
- Función objetivo: el beneficio según el número de productos fabricados: $Z(x):=2x_1+4x_2+2.5x_3$

El problema queda formulado como sigue:

$$\begin{aligned} \max Z(x) := \; & 2x_1+4x_2+2.5x_3 \\ \text{s.t.} \; & 3x_1+4x_2+2x_3 \leq 600 \\ & 2x_1+x_2+2x_3 \leq 400 \\ & x_1+3x_2+3x_3 \leq 300 \\ & x_i \geq 0, \text{ y entera, } i=1,2,3 \end{aligned}$$

In [31]:

```
p3=MixedIntegerLinearProgram()
x=p3.new_variable(integer=True,nonnegative=True)
p3.add_constraint(3*x[1] + 4*x[2]+2*x[3] <= 600)
p3.add_constraint(2*x[1]+x[2]+2*x[3]<=400)
p3.add_constraint(x[1]+3*x[2]+3*x[3]<=300)
p3.set_objective(2*x[1]+4*x[2]+2.5*x[3])
print ('Valor objetivo=', p3.solve())
print ('Solución=', p3.get_values(x))
```

```
Valor objetivo= 480.0
Solución= {1: 120.0, 2: 60.0, 3: 0.0}
```

In [ ]: