Título de la Práctica: Modelización y resolución de problemas de Programación Lineal.

Profesor Responsable: Inmaculada Rodríguez Martín.

Práctica número: 2

Objetivo:

En esta clase veremos cómo podemos resolver problemas de programación lineal utilizando GLPK (*GNU Linear Programming Kit*). Lo vamos a utilizar desde el sistema operativo Windows con la interfaz denominada GUSEK (*GLPK Under Scite Extended Kit*). También se peude utilizar con *Wine* de *Linux.* Además de resolver un problema de la forma estándar, con GLPK es posible leer modelos de problemas sin necesidad de previamente definir todos los datos del problema. Los contenidos de la clase son los siguientes:

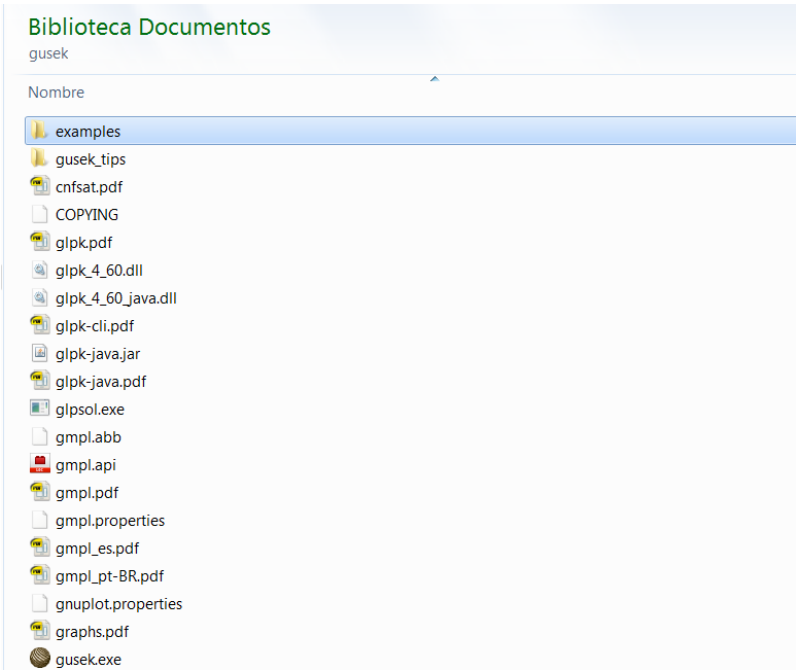
- Instalación de GUSEK.
- Resolución de un problema sencillo utilizando GUSEK.
- Resolución de un problema general de programación lineal.
- Ejercicios.

1. Instalación de GUSEK.

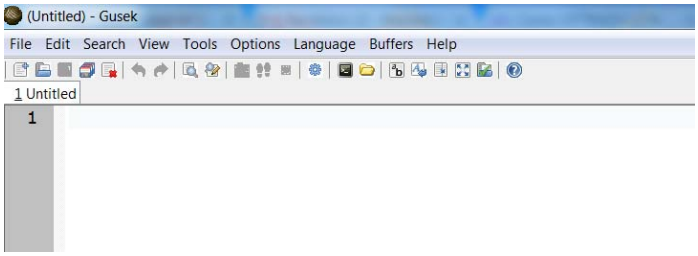
En realidad, GUSEK no es necesario instalarlo. Es una aplicación portable, como indican en su página web (http://gusek.sourceforge.net/gusek.html), basta con descargar un archivo comprimido, descomprimirlo en un directorio (carpeta) y ejecutar el fichero gusek.exe.

Además del ejecutable *gusek.exe*, el directorio contiene:

- Muchos modelos de ejemplo y ficheros de datos. Están incluidos en los subdirectorios *ejemplos* y *gusek_tips*. Los modelos tienen extensión *.mod* y los ficheros de datos *.dat*.
- Otros ficheros ejecutables. Que deben estar junto con gusek.exe y no vamos a utilizar directamente.
- Ficheros de manuales y ayuda. Por ejemplo *gmpl_es.pdf* que es un manual de GMPL (*GNU Mathematical Programming Language*), el lenguaje de modelado de GLPK.
- Otros ficheros.



Si abrimos el programa observamos una pantalla como la siguiente. Iremos viendo poco a poco para qué sirven algunas de las funciones de este programa.



2. Resolución de un problema sencillo utilizando GUSEK.

Resolvamos el siguiente problema de programación lineal.

$$\begin{aligned} \min\ & 4x_1 - 2x_2 + 2x_3 \\ \text{s.a:}\ & 6x_2 + x_3 = 4 \\ & x_1 - x_2 = 2 \\ & x_1 \geq 0, x_2 \geq 0, x_3 \geq 0 \end{aligned}$$

Para ello en el lenguaje de modelado de GMPL habría que poner algo así:

```
/*
Este es mi primer programa en GUSEK
Vamos a resolver un problema de programación lineal sencillo
*/
# Definición de las variables
var x1 >= 0;
var x2 >= 0;
var x3 >= 0;
# Función objetivo
minimize z: 4*x1 - 2*x2 + 2*x3;
# Restricciones
r1:           6*x2 +   x3 = 4;
r2:      x1 -   x2        = 2;
```

```
# Lo solucionamos y mostramos los resultados
solve;
display x1, x2, x3;
end;
```

Observen cómo se ponen los comentarios, qué palabras clave utilizamos y para qué las utilizamos, qué funciones del menú de GUSEK podemos utilizar, etc. Para ver si tiene errores de lenguaje lo podemos hacer desde *Tools→Compile* y para ejecutarlo desde *Tools→Go*.

GLPK permite resolver problemas forzando que algunas variables (o todas) sean enteras. Por ejemplo intenta resolver el mismo ejemplo forzando que la variable `x1` sea entera (`var x1 >= 0, integer;`).

Se puede definir vectores y matrices para los parámetros y para los datos. De esta forma podemos separar el modelo de los datos:

```
/*
Este es mi segundo programa en GUSEK
Vamos a resolver un problema anterior donde parametrizamos los datos y los separamos del modelo.
*/
#Definición de parámetros
param c{1..3};
param A{1..2, 1..3};
param b{1..2};
# Definición de las variables
var x{1..3}, >= 0;
# Función objetivo
minimize z:  c[1]*x[1] +  c[2]*x[2] +  c[3]*x[3];
# Restricciones
r1:    A[1,1]*x[1] + A[1,2]*x[2] + A[1,3]*x[3] = b[1];
r2:    A[2,1]*x[1] + A[2,2]*x[2] + A[2,3]*x[3] = b[2];
# Lo solucionamos y mostramos los resultados
solve;
display x;

#  Proporcionamos los datos del problema (Parámetros c, A y b)
data;
# Al definir un vector indicamos los índices
param c:=   1   4
           2   -2
           3   2;
# Al definir una matriz indicamos los índices por columna y por fila
param A:    1   2   3 :=
        1   0   6   1
        2   1   -1  0;
param b:=   1   4
           2   2;
end;
```

Aún más sencillo, la función objetivo y las restricciones se pueden poner de forma más compacta utilizando sumatorios con la palabra clave en GMPL **sum**. Así, el ejemplo anterior quedaría de la siguiente forma:

```
# Función objetivo
minimize z:  sum{j in 1..3} c[j]*x[j];
# Restricciones
s.t. retric{i in 1..2}: sum{j in 1..3} A[i,j] * x[j] = b[i];
```

GUSEK permite que el conjunto de los datos esté en otro fichero (que por defecto tiene el mismo nombre que el modelo pero con extensión *.dat*).

Otra gran ventaja que tiene el lenguaje de modelado de problemas de programación lineal es que se pueden definir conjuntos (cuyos elementos pueden ser numéricos o de cadena) y definir parámetros y variables de decisión con respecto a estos conjuntos. Para ello pueden ver (entre otros muchos) el ejemplo que trae GUSEK llamado *diet.mod*.

Ejercicios:

1) Resuelve el siguiente problema de Programación Lineal:

$$\begin{aligned} \min\ & -4x_1 - 2x_2 + 2x_3 + x_4 - x_5 \\ \text{s.a: } & 2x_1 - 2x_2 + 4x_3 + x_4 = 10 \\ & -x_1 + 4x_2 - 2x_3 + x_5 = 16 \\ & x_j \geq 0, \forall j \in \{1,\ldots,5\} \end{aligned}$$

2) Resuelve el siguiente problema de Programación Lineal:

$$\begin{aligned} \min\ & -3x_1 + 2x_2 - x_3 \\ \text{s.a: } & 2x_1 + x_2 - x_3 = 8 \\ & -2x_1 + 2x_2 + x_3 = 6 \\ & x_j \geq 0, \forall j \in \{1,2,3\} \end{aligned}$$

3) Resuelve el siguiente problema de programación Lineal:

$$\begin{aligned} \max\ & 4x_1 + 3x_2 \\ s.a: & \ 2x_1 + 3x_2 \leq 8 \\ & 6x_1 - x_2 \geq 4 \\ & x_1 \geq 0, x_2 \geq 0 \end{aligned}$$

3. Resolver cualquier problema de Programación Lineal general en su forma estándar.

Como hemos visto, el lenguaje de modelado permite la definición de conjuntos de índices. Vamos a utilizar esta característica para modelar un problema que esté dado en su forma estándar:

$$\min\ c^t x$$
$$s.a:\ Ax = b$$
$$x \geq 0$$

Trata de hacerlo tú mismo. A continuación te damos algunos consejos para realizarlo:

- Definimos los parámetros m y n que corresponden al número de restricciones y variables, respectivamente.
- Por facilitar la escritura podemos definir unos conjuntos `set I:= {1..m};` y `set J:= {1..n};` que corresponden al conjunto de filas y columnas del problema.
- Utiliza el comando `sum` para poner la función objetivo y restricciones.
- Para el conjunto de datos define los parámetros m y n, los vectores b y c y la matriz A.
- Puedes mirar los ejemplos que trae GUSEK para ayudarte.

4. Resolución del problema de Transporte.

Utilizando el modelo de programación lineal en su forma estándar podemos resolver cualquier problema que esté representado de esta forma. Por ejemplo el siguiente problema que hemos visto en clase:

Tres almacenes ofertan un producto que se demanda en cinco mercados. Las cantidades ofertadas en origen, las demandas de los destinos (medidas ambas en cientos de kilogramos) y los costos de transporte (euros por unidad de producto) entre cada almacén y cada mercado, aparecen en la siguiente tabla:

	M1	M2	M3	M4	M5	Ofertas
Al1	3	5	2	4	6	38
Al2	5	2	7	6	3	44
Al3	4	6	8	9	5	69
Demandas	43	29	37	14	28	

Interesa diseñar una política de transporte, que minimice los costos globales, atendiendo las demandas de los mercados y respetando las ofertas de los almacenes

Las variables de decisión deben ser:

x_{ij} = cantidad de producto que se ha de transportar desde el almacén i al mercado j, para $i = 1,2,3$ y $j = 1, \dots, 5$.

Este modelo se puede poner de la forma estándar de la siguiente manera:

$$\min\ 2x_{11}+5x_{12}+2x_{13}+4x_{14}+6x_{15}+5x_{21}+2x_{22}+7x_{23}+6x_{24}+3x_{25}+4x_{31}+6x_{32}+8x_{33}+9x_{34}+5x_{35}$$

$$s.a:\ x_{11}+x_{12}+x_{13}+x_{14}+x_{15} = 38$$
$$x_{21}+x_{22}+x_{23}+x_{24}+x_{25} = 44$$
$$x_{31}+x_{32}+x_{33}+x_{34}+x_{35} = 69$$
$$x_{11}+x_{21}+x_{31} = 43$$
$$x_{12}+x_{22}+x_{32} = 29$$
$$x_{13}+x_{23}+x_{33} = 37$$
$$x_{14}+x_{24}+x_{34} = 14$$
$$x_{15}+x_{25}+x_{35} = 28$$
$$x_{ij} \geq 0, i = 1,2,3, j = 1,2,3,4,5$$

Que es un caso particular del problema de transporte que en general lo podemos formular de la siguiente forma:

$$\min \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij}$$
$$s.a: \sum_{j=1}^{n} x_{ij} = a_i,\ i = 1,\ldots,m$$
$$\sum_{i=1}^{m} x_{ij} = b_j,\ j = 1,\ldots,n$$
$$x_{ij} \geq 0,\ i = 1,\ldots,m,\ j = 1,\ldots,n$$

Veamos cómo podemos diseñar un modelo en GUSEK que nos permita resolver cualquier problema del transporte dados los parámetros a_i, b_j y c_{ij} para $i = 1, \ldots, m$ y $j = 1, \ldots, n$.

Después de definir los parámetros m y n de forma similar al apartado anterior, podemos definir (por simplicidad) los siguientes conjuntos de índices:

```
set I:= {1..m}; # Conjunto de almacenes
set J:= {1..n}; # Conjunto de mercados
```

Después declaramos los vectores a y b y la matriz c.

```
param a{i in I}; # Suministros de los almacenes
param b{j in J}; # Demandas de los mercados
param c{i in I, j in J}; # Coste de enviar una unidad del almacén i al mercado j
```

Y ahora la variable x está indexada con dos índices, luego la declaramos de la siguiente forma:

```
var x{i in I, j in J} >= 0; # cantidad de producto que se ha de transportar desde el almacén i al mercado j
```

Es hora de poner el modelo (función objetivo y restricciones).

```
minimize cost: sum{i in I, j in J} c[i,j] * x[i,j]; # Coste total de transporte
s.t. suministro{i in I}: sum{j in J} x[i,j] = a[i]; # El suministro del origen i es a[i]
s.t. demanda{j in J}: sum{i in I} x[i,j] = b[j];  # La demanda del destino j es b[j]
```

Ya solo queda introducir los datos del problema que queremos resolver. Esto se hace de forma similar a la sección anterior.

```
data;
# Al definir un vector indicamos los índices
param m:= 3;
param n:= 5;

#  Proporcionamos los datos del problema (Parámetros a, b y c)
param a:=   1   38
            2   44
            3   69;

param b:=   1   43
            2   29
            3   37
            4   14
            5   28;
param c:    1   2   3   4   5 :=
        1   3   5   2   4   6
        2   5   2   7   6   3
        3   4   6   8   9   5;
end;
```

Ejercicio: Resolver el problema del transporte donde se establece una familia de restricciones adicional en la que de cualquier almacén i a cualquier mercado j no se puede enviar más de K unidades de producto. Resolverlo para el caso particular de $K = 30$. ¿Cómo será el valor óptimo de este problema igual, mayor o menor que en el problema sin esta restricción?

5. Cuestionario.

Para evaluar el rendimiento de la práctica, tienes que realizar un pequeño cuestionario tipo test que está accesible en el Campus Virtual ULL de OPTIMIZACIÓN. El cuestionario se puntúa de 0 a 10 y esa nota se trasladará luego a la siguiente escala: un suspenso equivale a un 0, un aprobado a un apto- (1), un notable a un apto (1,6) y un sobresaliente a un apto+ (2).

Nota Cuestionario	Calificación Cualitativa	Calificación para evaluación continua
[0,4)	No apto	0 puntos
[4,6)	Apto-	1 punto
[6,8)	Apto	1,6 puntos
[8,10)	Apto+	2 puntos