Programación Lineal Entera

ESCUELA TÉCNICA SUPERIOR DE
INGENIERÍA INFORMÁTICA
Departamento de Lenguajes y Sistemas Informáticos
Curso 2020-2021

Contenido

1. Programación Lineal Entera (PLE)
2. Problema de la Mochila
3. Detalles de implementación
4. Redes de flujo generalizadas

1. Programación Lineal Entera

- **Problemas de optimización**
  - Se definen por un conjunto de restricciones y una función objetivo
  - La función objetivo permite calcular un valor para cada una de las posibles soluciones
  - Se busca la solución que, cumpliendo con las restricciones del problema, minimice (o maximice) el valor de la función objetivo
- Los problemas que no son de optimización pero buscan una o varias soluciones que cumplan las restricciones del problema pueden transformarse en problemas de optimización
- En algunos casos la complejidad del algoritmo que resuelve el problema de optimización es demasiado alta
  - Por ejemplo cuando el algoritmo es exhaustivo debe encontrar todas las posibles soluciones para poder garantizar que ha encontrado el óptimo

1. Programación Lineal Entera

- La técnica de Programación Lineal (PL) consiste en modelar un problema, normalmente de optimización, mediante un conjunto de restricciones lineales sobre variables de tipo real más una función objetivo (a maximizar o minimizar)
- El nuevo problema se denomina problema de PL y puede ser resuelto por el algoritmo del Simplex (complejidad polinomial)
- El problema de PL puede ser completado con otras restricciones como:
  - Algunas variables pueden tomar valores enteros
  - Algunas variables pueden tomar valores binarios
  - Dado un conjunto de variables sólo *k* de ellas pueden tomar, en la solución, valores distintos de cero
- Con estas restricciones tendríamos un problema de Programación Lineal Entera (PLE), donde algunas variables del problema pueden tomar valores enteros o binarios

1. Programación Lineal Entera

- PLE tiene un mayor poder expresivo que PL pero su complejidad computacional es más alta que polinomial
- Para resolver problemas PLE se usa una mezcla de relajación lineal más algoritmos de vuelta atrás (más detalles en el apartado 4)
- Un PLE se compone de:
  - Un conjunto de variables reales
  - Un conjunto de variables enteras
  - Un conjunto de variables binarias
  - Una función objetivo que será una combinación lineal de las variables
  - Un conjunto de restricciones (mayor, menor, igual, mayor o igual, menor o igual) entre combinaciones lineales de variables

2. Catálogo de problemas

❑ Problema de la Mochila:

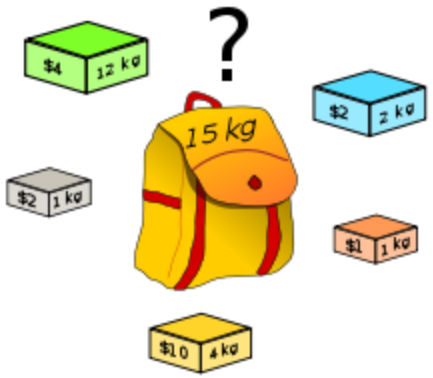
Parte de un multiconjunto de objetos $M = (Lo,m)$, donde Lo es una lista de objetos de tamaño n y m una lista de enteros del mismo tamaño donde m_i indica el número de repeticiones del objeto en la posición i

Cada objeto ob_i de la lista tiene un peso y un valor unitario:

$$ob_i = (w_i, v_i)$$

La mochila tiene una capacidad C

Objetivo: ubicar en la mochila el máximo número unidades de cada objeto que quepan en la mochila para que el valor sea máximo



2. Catálogo de problemas

- **Problema de la Mochila**:

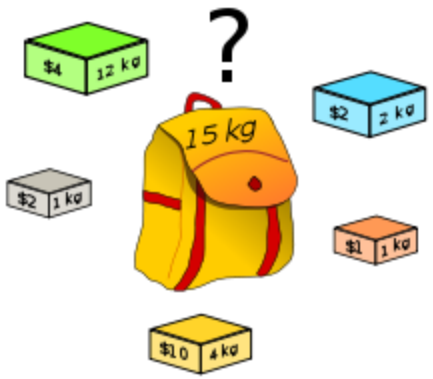
Si x_i es el número de unidades del objeto *i* en la mochila, el problema puede enunciarse como un problema de PLE:

$$\max \sum_{i=0}^{n-1} x_i v_i$$

$$\sum_{i=0}^{n-1} x_i w_i \leq C$$

$$x_i \leq m_i, \quad i \in [0, n-1]$$

$$int \quad x_i, \quad i \in [0, n-1]$$



2. Catálogo de problemas

- **Problema de la Mochila**: Ejemplo

C = 45

M =

i	v	w	m
2	30	8	2
1	76	16	2
0	48	9	2

$$\max \sum_{i=0}^{n-1} x_i v_i$$

$$\sum_{i=0}^{n-1} x_i w_i \leq C$$

$$x_i \leq m_i, \quad i \in [0, n-1]$$

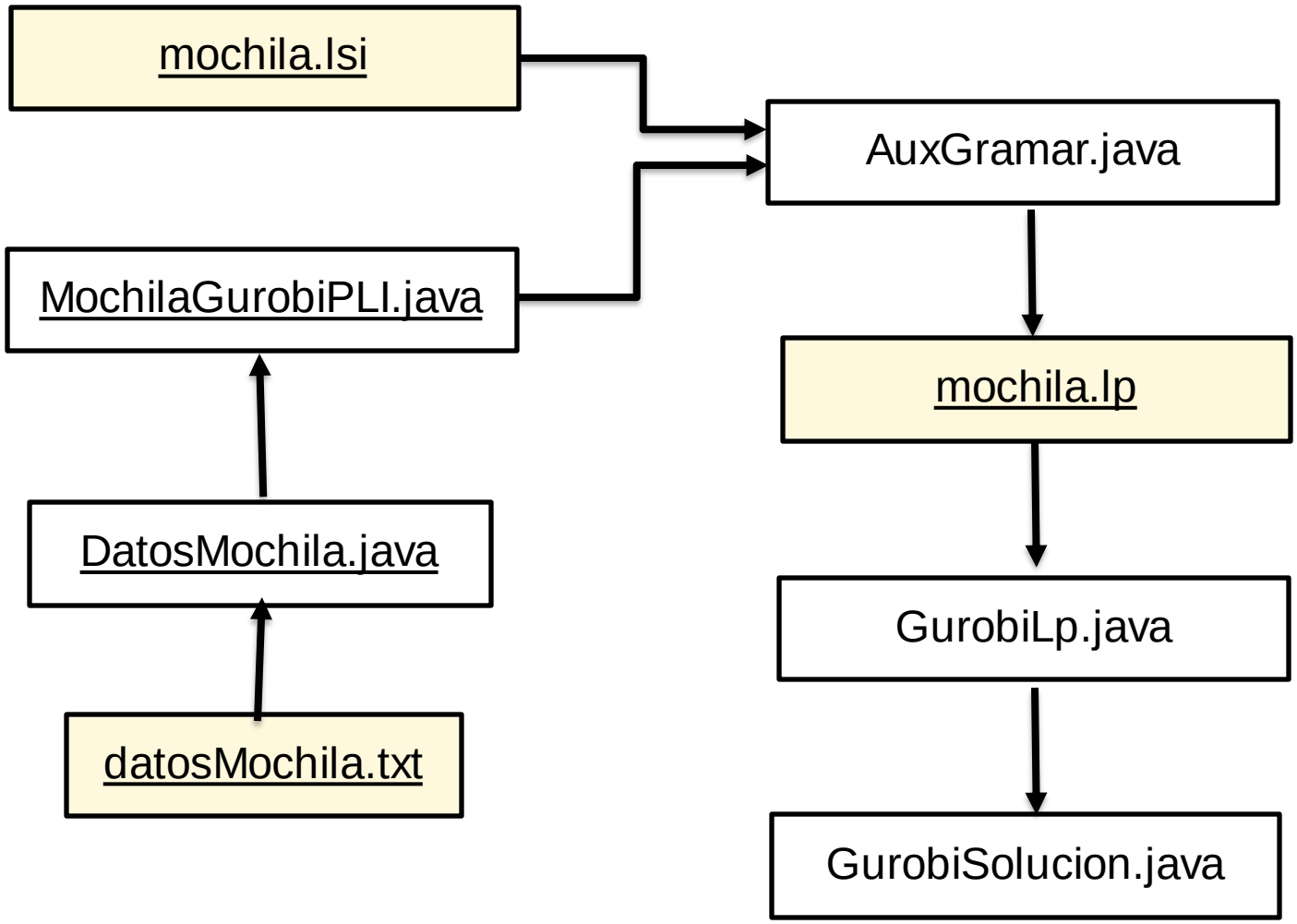
$$int \quad x_i, \quad i \in [0, n-1]$$

max: +48.0 x0 +76.0 x1 +30.0 x2;

+9.0 x0 +16.0 x1 +8.0 x2 <= 45.0;
+x0 <= 2;
+x1 <= 2;
+x2 <= 2;

int x0 ,x1 ,x2;

3. Detalles de Implementación
Modelo LSI



3. Detalle de Implementación Modelo LSI

head section

Integer getCI()
Integer getN()
Integer getValor(Integer i)
Integer getPeso(Integer i)
Integer getNMU(Integer i)
Integer CI = getCI()
Integer n = getN()

Métodos estáticos

goal section

max sum(getValor(i) x[i], i in 0 .. n)

constraints section

sum(getPeso(i) x[i], i in 0 .. n) <= CI

bounds section

x[i] <= getNMU(i), i in 0 .. n

int

x[i], i in 0 .. n

3. Detalle de Implementación Modelo LSI

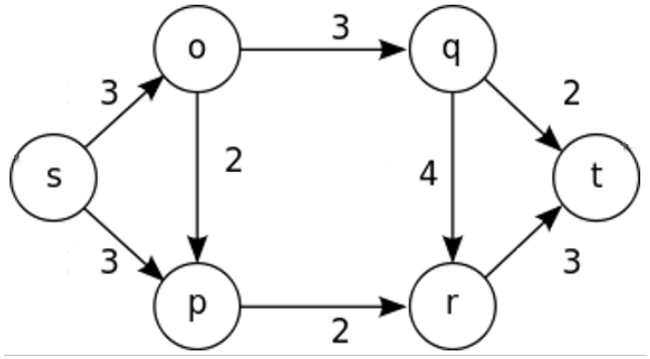
```
Locale.setDefault(new Locale("en", "US"));
DatosMochila.iniDatos("datosMochila.txt");
DatosMochila.capacidadInicial = 78;
MochilaGurobiPLI.CI = DatosMochila.capacidadInicial;
MochilaGurobiPLI.objetos = DatosMochila.getObjetos();
MochilaGurobiPLI.n = objetos.size();
AuxGrammar.generate(MochilaGurobiPLI.class,"mochila.lsi","mochila.lp");
GurobiSolution solution = GurobiLp.gurobi("mochila.lp");
System.out.println(solution.toString((s,d)->d>0.))
```

Contenido

1. Programación Lineal Entera (PLE)
2. Problema de la Mochila
3. Detalles de implementación
4. **Redes de flujo generalizadas**

4. Redes de flujo: generalización del flujo máximo

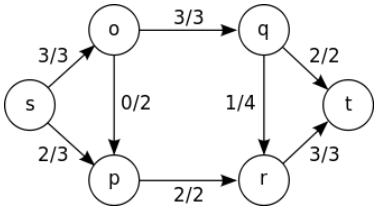
- Cada vértice y cada arista puede tener asociado:
  - un límite inferior,
  - un límite superior y



  - un coste unitario del flujo que pasa por él
- Función objetivo: maximizar o minimizar la suma de los costes del flujo que circula por aristas y vértices
- Podemos añadir la restricción adicional de que el flujo que circula por vértices y/o aristas tenga valor entero.
- Pueden existir más de una fuente y/o más de un sumidero

4. Redes de flujo: Generalización

- A partir de un grafo de flujo, se puede construir un PL (o PLE):
  - Asociar una variable a cada arista para representar el flujo que pasa por ella
  - Asociar una variable a cada vértice para representar el flujo que pasa por él (ó se consume, ó se genera)
  - Restricciones de flujo asociadas a cada vértice (leyes de *Kirchoff*)
  - Restricciones teniendo en cuenta límites inferior y superior de flujo por vértices y aristas
  - Función objetivo: max/min suma costes de flujos en aristas y/o vértices



4. Redes de flujo: Generalización

$$\min/\max \sum_{i \in V} w_i x_i + \sum_{j \in E} w_j y_j$$

$$x_i = \sum_{k \in Out(i)} y_k, \quad i \in F$$

$$\sum_{k \in In(i)} y_k = x_i, \quad i \in V - F$$

$$\sum_{k \in In(i)} y_k = \sum_{k \in Out(i)} y_k, \quad i \in I = V - F \cup S$$

$$b_i^d \leq x_i \leq b_i^u, \quad i \in F \cup S$$

$$b_j^d \leq y_j \leq b_j^u, \quad j \in E$$

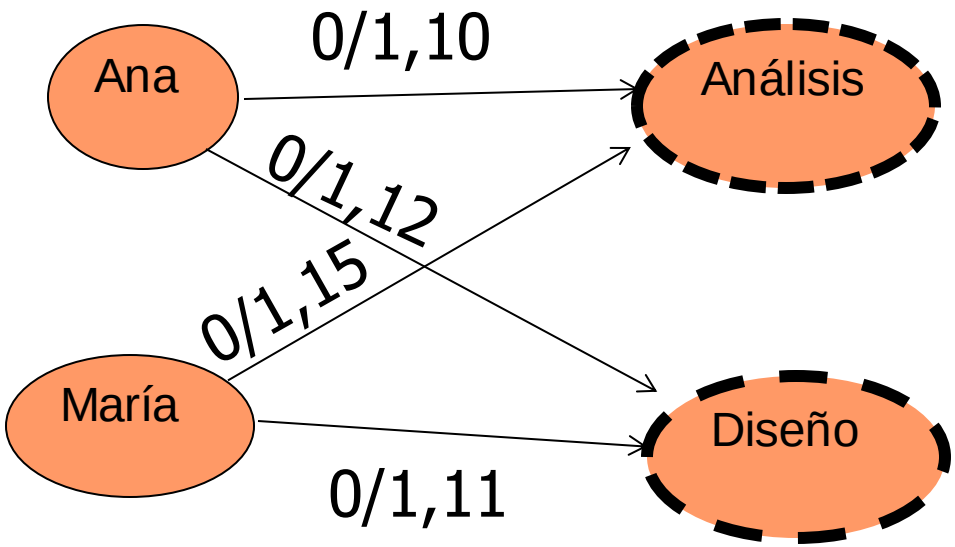
$$b_i^u \leq \sum_{k \in In(i)} y_k \leq b_i^u, \quad i \in I = V - F \cup S$$

- x_i, w_i, b_i: flujo, coste unitario, y límites de flujo por vértices
- y_j, w_j, b_j: flujo, coste unitario y límites de flujo por aristas
- V es el conjunto de vértices
- F el conjunto de vértices fuente
- S el conjunto de vértices sumidero
- I el conjunto de vértices intermedios
- E el conjunto de aristas

4. Redes de flujo: Generalización. Problemas

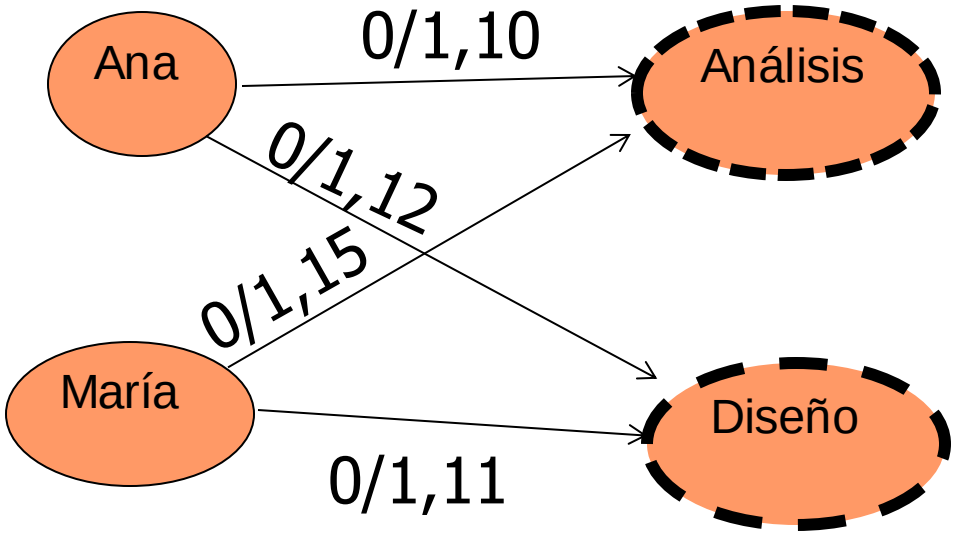
Vértices: id,tipo,min,max,coste
Aristas: *origen,destino,min,max,coste*
Tipos: *Source,Intermediate,Sink*
Por defecto: [min=0,max=inf,coste=0]

Persona Tarea	Ana	María
Análisis	10	15
Diseño	12	11




```
#VERTEX#
Ana,Source,0.,1.,0.
Maria,Source,0.,1.,0.
Analisis,Sink,1.,1.,0.
Diseño,Sink,1.,1.,0.
#EDGE#
Ana,Analisis,0.,1.,10
Ana,Diseño,0.,1.,12
Maria,Analisis,0.,1.,15
Maria,Diseño,0.,1.,11
```

4. Redes de flujo: Generalización. Problemas



Persona Tarea	Ana	María
Análisis	10	15
Diseño	12	11

```
min: +10.0*e0+12.0*e1+
                15.0*e2+11.0*e3;

v0 <= +1.0;    e0 <= +1.0;
v1 <= +1.0;    e1 <= +1.0;
v2 = +1.0;     e2 <= +1.0;
v3 = +1.0;     e3 <= +1.0;

v0 = e0+e1;
v1 = e2+e3;
e0+e2 = v2;
e1+e3 = v3;

int v0,v1,v2,v3,e0,e1,e2,e3;
```

4. Redes de flujo: Generalización. Problemas

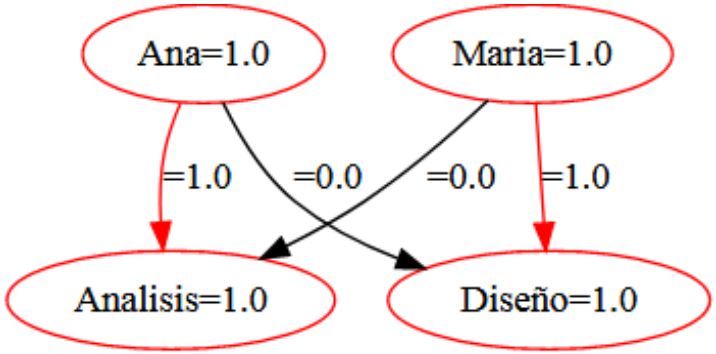
```
Locale.setDefault(new Locale("en", "US"));
FlowGraph fg = FlowGraph.newGraph("redFlujoTareas.txt");

FlowData.graph = g;
AuxGrammar.generate(FlowData.class,"flow.lsi",
                                                    "tareas_flow.lp");
GurobiSolution solution = GurobiLp.gurobi("tareas_flow.lp");
System.out.println(solution.toString((s,d)->d>0.));

FlowGraphSolution fs = FlowGraphSolution.of(g,solution);
fs.toDotIndex("redFlujoTareas.gv");

System.out.println(fs.goal);
System.out.println(fs.flowVertices);
System.out.println(fs.flowEdges);
```

Persona Tarea	Ana	María
Análisis	10	15
Diseño	12	11




```
#VERTEX#
Ana,Source,0.,1.,0.
Maria,Source,0.,1.,0.
Analisis,Sink,1.,1.,0.
Diseño,Sink,1.,1.,0.
#EDGE#
Ana,Analisis,0.,1.,10
Ana,Diseño,0.,1.,12
Maria,Analisis,0.,1.,15
Maria,Diseño,0.,1.,11
```

Programación Lineal Entera

ESCUELA TÉCNICA SUPERIOR DE
INGENIERÍA INFORMÁTICA
Departamento de Lenguajes y Sistemas Informáticos
Curso 2020-2021