Herramientas para el diseño y análisis de redes de transporte urbano de pasajeros

Tema 4: Elementos básicos de programación matemática (cont.)

Herramientas de software para programación matemática

- Paquetes que incluyen algoritmos de propósito general para problemas LP, IP y MIP.
- Diferentes niveles de acceso a las funcionalidades: API (Application Programming Interface) invocable desde lenguajes de programación (C, Java), solvers (motores de optimización) que trabajan directamente con la formulación matemática del problema y los datos de la instancia a resolver.
- Solvers: comerciales (CPLEX, GUROBI) y de libre acceso (GLPK).
- Lenguajes de especificación de la formulación matemática: AMPL, MathProg.

GLPK (GNU Linear Programming Kit)

- Kit de programación lineal y entera de GNU.
- Incluye las siguientes funcionalidades:
  - Motor de optimización: algoritmos simplex y de punto interior para LP y branch-and-cut para IP y MIP.
  - API de funciones.
  - Intérprete del lenguaje GNU MathProg.
  - *Solver LP/MIP stand-alone, accesible desde línea de comando (el que usaremos en este curso).*
- Sitio web: https://www.gnu.org/software/glpk/

Descarga, instalación y documentación

Ejemplo para Windows 10. Para otras plataformas, ver documentación en el sitio web de GLPK.

- Instalador: http://winglpk.sourceforge.net/
- Configuración: agregar la carpeta del ejecutable $glpsol.exe$ ($w32$ o $w64$) a la variable PATH.
- Documentación (en carpeta doc de la instalación):
  - GNU Linear Programming Kit (Reference Manual): archivo $glpk.pdf$, Apéndice D, LP/MIP solver stand-alone (invocación del comando, parámetros).
  - Modeling Language GNU MathProg (Language Reference): archivo $gmpl.pdf$, sintaxis y semántica del lenguaje.

Modalidad de trabajo

- Archivo $.mod$ del modelo (conjuntos, parámetros, variables, restricciones y objetivos) y eventualmente instrucciones para su ejecución y despliegue de resultados ($solve$ y $display$).

- Archivo $.dat$ con datos de una instancia particular.

- Ejemplo de ejecución: $glpsol$ -m $problema.mod$ -d $problema.dat$

Algunos ejemplos

- Problema de transporte (diapositiva 11 de la clase 4a).
- Problema de transporte con costos fijos (diapositiva 13 de la clase 4a).
- Camino de costo mínimo en una red (diapositiva 18 de la clase 4a).

Problema de transporte

Se debe enviar mercadería desde m puntos, la cual debe recibirse en n puntos, minimizando los costos de transporte entre origen y destino.

$$\begin{aligned}
\min \quad & \sum_{ij} c_{ij} x_{ij} \\
\text{s.a.} \quad & \sum_{j} x_{ij} = a_i && \forall\ i = 1, 2, \ldots, m, \\
& \sum_{i} x_{ij} = b_j && \forall\ j = 1, 2, \ldots, n, \\
& x_{ij} \geq 0 && \forall\ i = 1, 2, \ldots, m; j = 1, 2, \ldots, n.
\end{aligned}$$

donde a_i es lo disponible en i, b_j es lo que debe llegar a j, c_{ij} es el costo de transporte desde i hasta j y x_{ij} es la variable que indica cuanta mercadería se envía desde i hasta j.

Problema de transporte (modelo)

```
set I; # origenes
set J; # destinos

param a{i in I}; # produccion en origen i
param b{j in J}; # demanda en destino j
param c{i in I, j in J}; # costo de transporte desde i hasta j

var x{i in I, j in J} >= 0; # cantidad enviada de i a j

minimize costo: sum{i in I, j in J} c[i,j] * x[i,j];
# funcion objetivo, costo total de transporte

s.t. oferta{i in I}: sum{j in J} x[i,j] = a[i]; # salida del origen i

s.t. demanda{j in J}: sum{i in I} x[i,j] = b[j]; # llegada al destino j
```

Problema de transporte (modelo, cont.)

```
solve; # ejecutar el modelo

display{i in I, j in J}: x[i,j]; # desplegar la solucion optima
```

Problema de transporte (datos de instancia)

```
data;
set I := Canelones Salto;

set J := Montevideo Maldonado Colonia;

param a := Canelones 350
           Salto 600;

param b := Montevideo 325
           Maldonado 300
           Colonia 325;

param c : Montevideo Maldonado Colonia :=
Canelones 46         155       145
Salto     496        605       404 ;
end;
```

Problema de transporte (resultado de ejecución)

```
x[Canelones,Montevideo].val = 325
x[Canelones,Maldonado].val = 25
x[Canelones,Colonia].val = 0
x[Salto,Montevideo].val = 0
x[Salto,Maldonado].val = 275
x[Salto,Colonia].val = 325
```

Problema de transporte con costos fijos

Al problema anterior se agrega un costo fijo f_{ij} por habilitar la conexión desde i hacia j, lo que se indica mediante la variable binaria y_{ij}.

$$\begin{aligned}
\min \quad & \sum_{ij} (c_{ij} x_{ij} + f_{ij} y_{ij}) \\
\text{s.a.} \quad & \sum_{j} x_{ij} = a_i && \forall\ i = 1, 2, \ldots, m, \\
& \sum_{i} x_{ij} = b_j && \forall\ j = 1, 2, \ldots, n, \\
& x_{ij} \leq M y_{ij} && \forall\ i = 1, 2, \ldots, m; j = 1, 2, \ldots, n, \\
& x_{ij} \geq 0 && \forall\ i = 1, 2, \ldots, m; j = 1, 2, \ldots, n, \\
& y_{ij} \in \{0, 1\} && \forall\ i = 1, 2, \ldots, m; j = 1, 2, \ldots, n.
\end{aligned}$$

donde M es un valor suficientemente grande (puede ser $\sum_i a_i = \sum_j b_j$).

Problema de transporte con costos fijos (modelo)

```
set I;
set J;

param a{i in I};
param b{j in J};
param c{i in I, j in J};
param f{i in I, j in J};
param M;

var x{i in I, j in J} >= 0;
var y{i in I, j in J} binary;

minimize costo: sum{i in I, j in J} (c[i,j] * x[i,j] + f[i,j] * y[i,j]);
s.t. oferta{i in I}: sum{j in J} x[i,j] = a[i];
s.t. demanda{j in J}: sum{i in I} x[i,j] = b[j];
s.t. activacion{i in I, j in J}: x[i,j] <= M * y[i,j];
```

Problema de transporte con costos fijos (datos)

```
data;

set I := Canelones Salto;

set J := Montevideo Maldonado Colonia;

param a := Canelones 350
           Salto 600;

param b := Montevideo 325
           Maldonado 300
           Colonia 325;
```

Problema de transporte con costos fijos (datos, cont.)

```
param c : Montevideo Maldonado Colonia :=
Canelones 46         155       145
Salto     496        605       404 ;

param f : Montevideo Maldonado Colonia :=
Canelones 4600         15500       14500
Salto     49600        60500       40400 ;

param M := 950;

end;
```

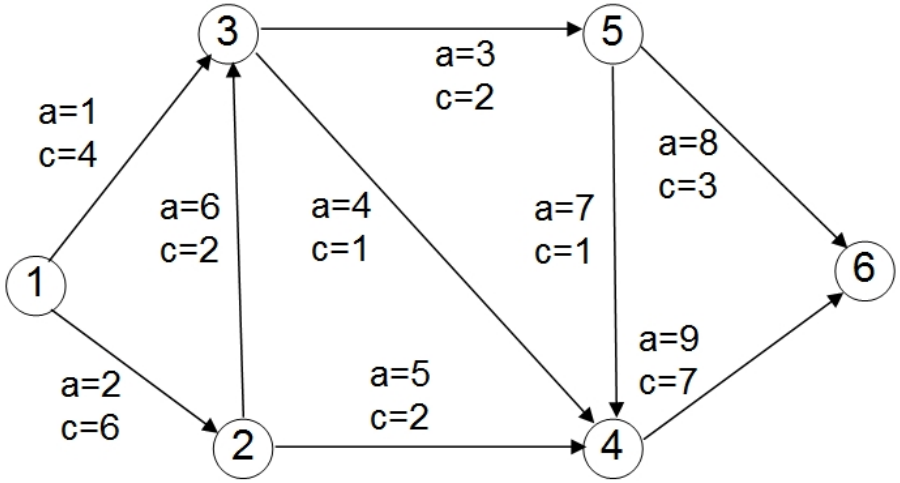
Problema de transporte con costos fijos (resultado de ejecución)

```
x[Canelones,Montevideo].val = 50
x[Canelones,Maldonado].val = 300
x[Canelones,Colonia].val = 0
x[Salto,Montevideo].val = 275
x[Salto,Maldonado].val = 0
x[Salto,Colonia].val = 325
```

```
y[Canelones,Montevideo].val = 1
y[Canelones,Maldonado].val = 1
y[Canelones,Colonia].val = 0
y[Salto,Montevideo].val = 1
y[Salto,Maldonado].val = 0
y[Salto,Colonia].val = 1
```

Camino de costo mínimo en una red

- Formulación en términos de arcos y para un solo destino (Tema 5).
- Caso de prueba (las etiquetas de los arcos indican el identificador (a) y el costo (c), el origen es el nodo 1, el destino es el nodo 6):



Camino de costo mínimo en una red (modelo)

```
set N; # nodos del grafo
set A; # arcos del grafo

set salientes{N} within A; # arcos salientes de un nodo
set entrantes{N} within A; # arcos entrantes a un nodo

param c{a in A} >= 0; # costo del arco a
param t{n in N}; # flujo requerido en el nodo n

var x {a in A} >= 0;

minimize costo: sum {a in A} (c[a] * x[a]);

s.t. conservacion_flujo {n in N}: sum {a in salientes[n]} x[a] -
                                  sum {a in entrantes[n]} x[a] = t[n];
```

Camino de costo mínimo en una red (datos)

```
data;

set N := 1 2 3 4 5 6;

set A := 1 2 3 4 5 6 7 8 9;

set salientes[1] := 1 2;
set salientes[2] := 5 6;
set salientes[3] := 3 4;
set salientes[4] := 9;
set salientes[5] := 7 8;
set salientes[6] := ;

set entrantes[1] := ;
set entrantes[2] := 2;
```

```
set entrantes[3] := 1 6;
set entrantes[4] := 4 5 7;
set entrantes[5] := 3;
set entrantes[6] := 8 9;

param:    c :=
1 4
2 6
3 2
4 1
5 2
6 2
7 1
8 3
9 7
;

param:    t :=
```

```
1 1
2 0
3 0
4 0
5 0
6 -1
;

end;
```

Solución óptima: camino 1-3-8 con valor óptimo 9.