INGENIERÍA INDUSTRIAL
UNIVERSIDAD DE CHILE

Auxiliar Tarea 2

Optimización IN34A

Ma. Fernanda Bravo

Software

- Lenguaje de programación: ZIMPL

Descargar versión 2.08

http://zimpl.zib.de/

- Solver: SCIP

Descargar versión 1.1.0 (para Windows o según su SO)

http://scip.zib.de/download.shtml

Instalación

•Descargar, guardar y descomprimir Zimpl en una carpeta de fácil acceso.

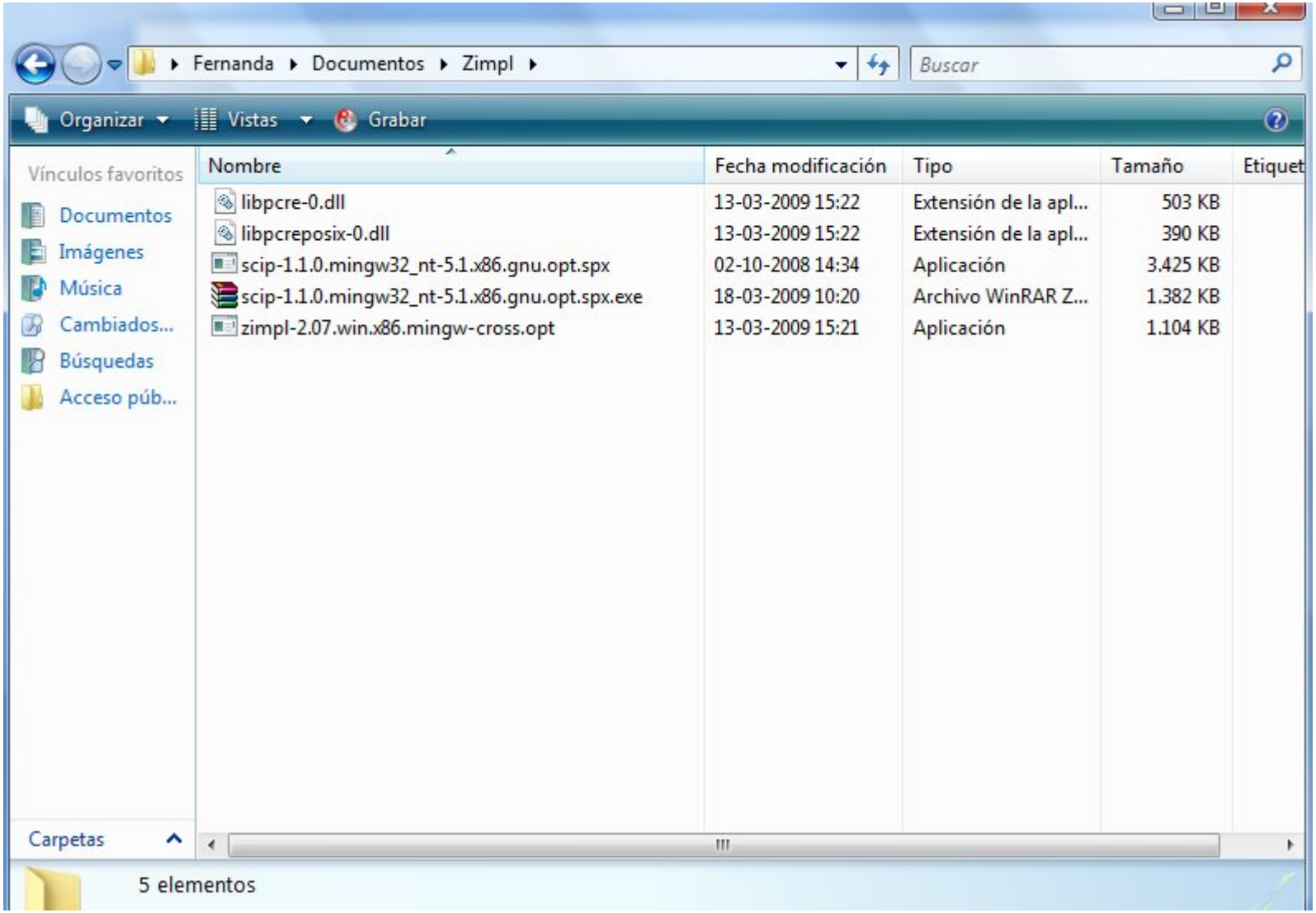
Ej: Documentos\Zimpl

En esta carpeta se almacenarán los modelos a optimizar, que se crean en Bloc de notas/Wordpad con la extensión *nombre.zpl*

Así también los datos de cada modelo, con la extensión *nombre.txt*

•Descargar y guardar SCIP en la carpeta creada para Zimpl, luego descomprimir .

Instalación

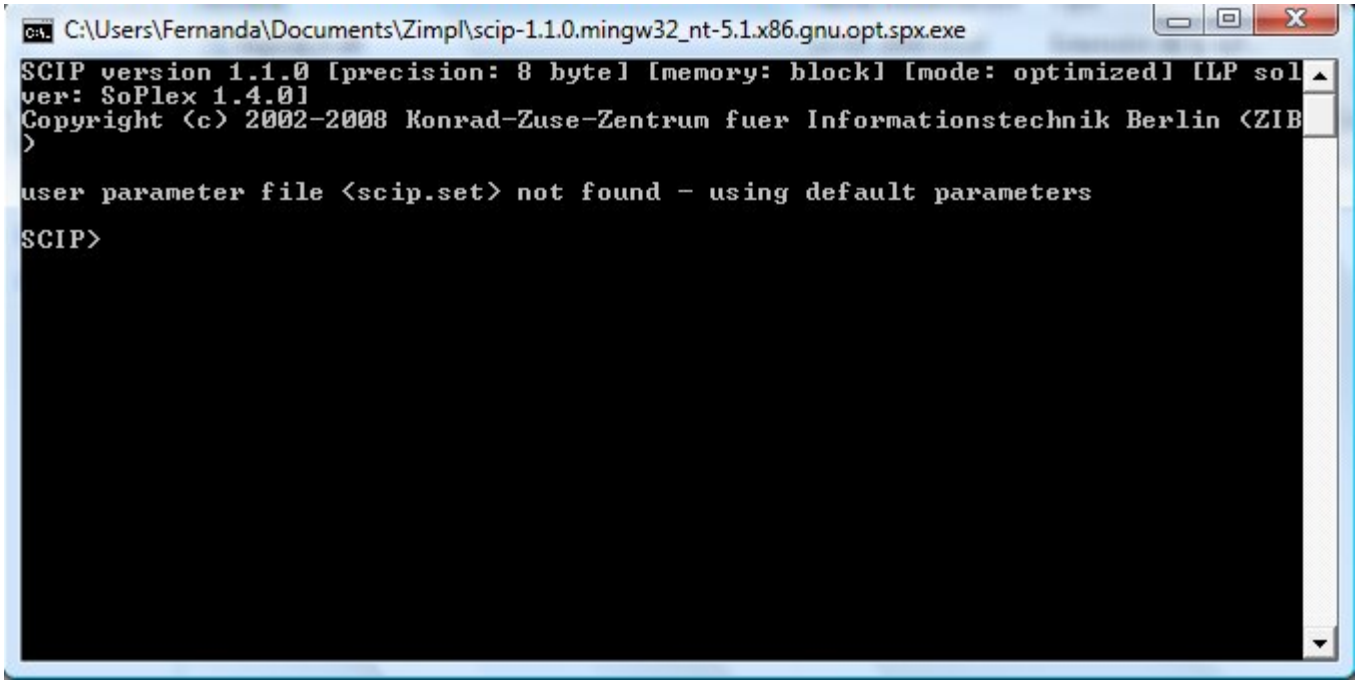


Ejecutar

Abrir SCIP mediante el siguiente archivo

scip-1.1.0.mingw32_nt-5.1.x86.gnu.opt.spx

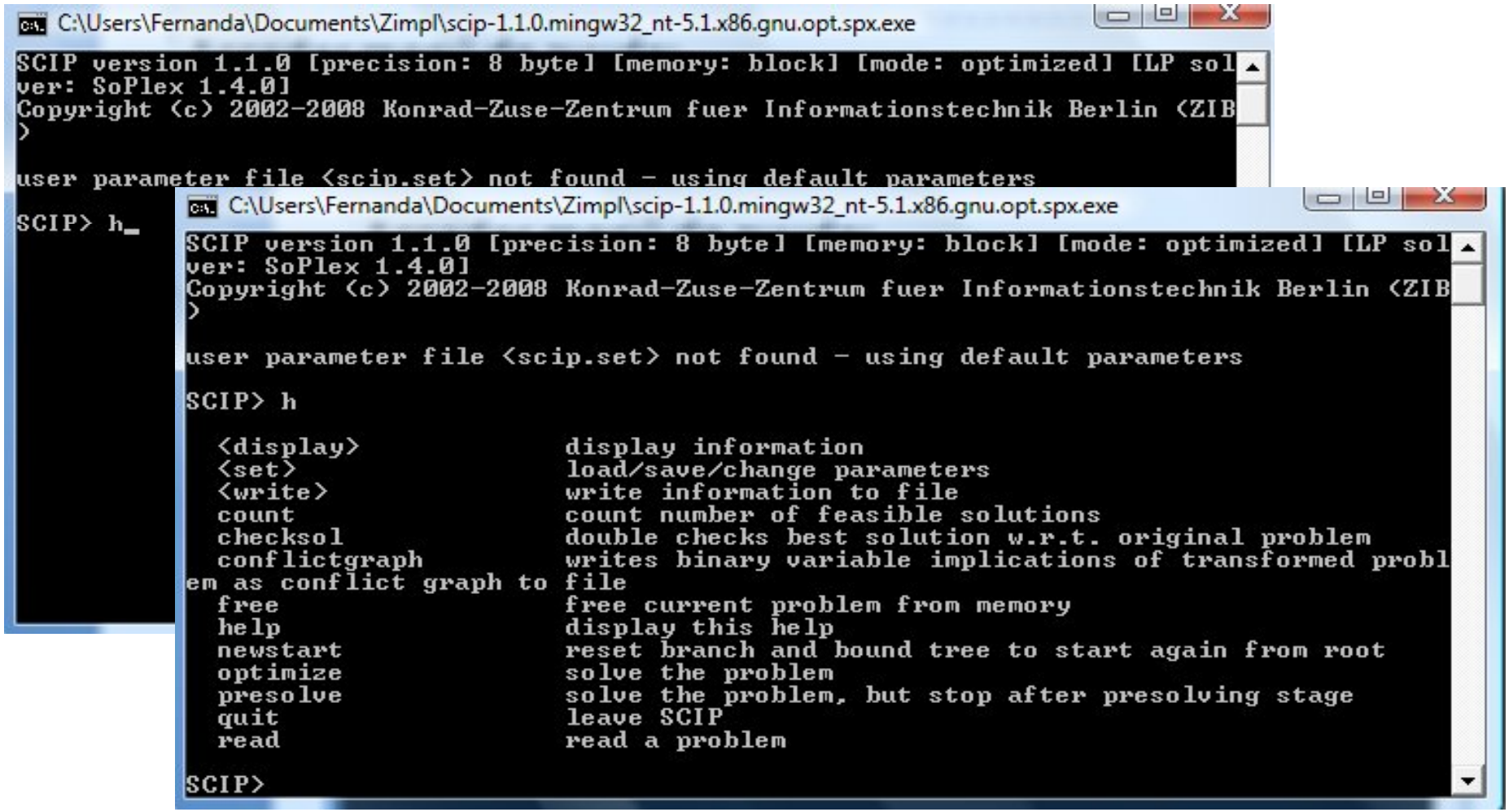
Se abrirá la siguiente consola DOS , a través de la cual daremos los comandos para resolver cada problema.



Ejecutar

Acceder menú de ayuda:

SCIP > h Enter



Resolver

Ejemplo Knapsack

$$Max \sum_i v_i x_i$$

$$st \; w_i x_i \leq W$$

$$x_i \in \{0,1\}$$

Escribimos el modelo knapsack.zpl y guardar en carpeta Zimpl

```
set Items := { 1, 2, 3, 4, 5, 6};
param v[Items] := <1> 1, <2> 2, <3> 3, <4> 5, <5> 7, <6> 11;
param w[Items] := <1> 13, <2> 17, <3> 19, <4> 23, <5> 29, <6> 31;
param W := 80;

var x[Items] binary;

maximize value: sum <i> in {1..6} do v[i]*x[i];

subto capacity: sum <i> in {1..6} do w[i]*x[i] <= W;
```

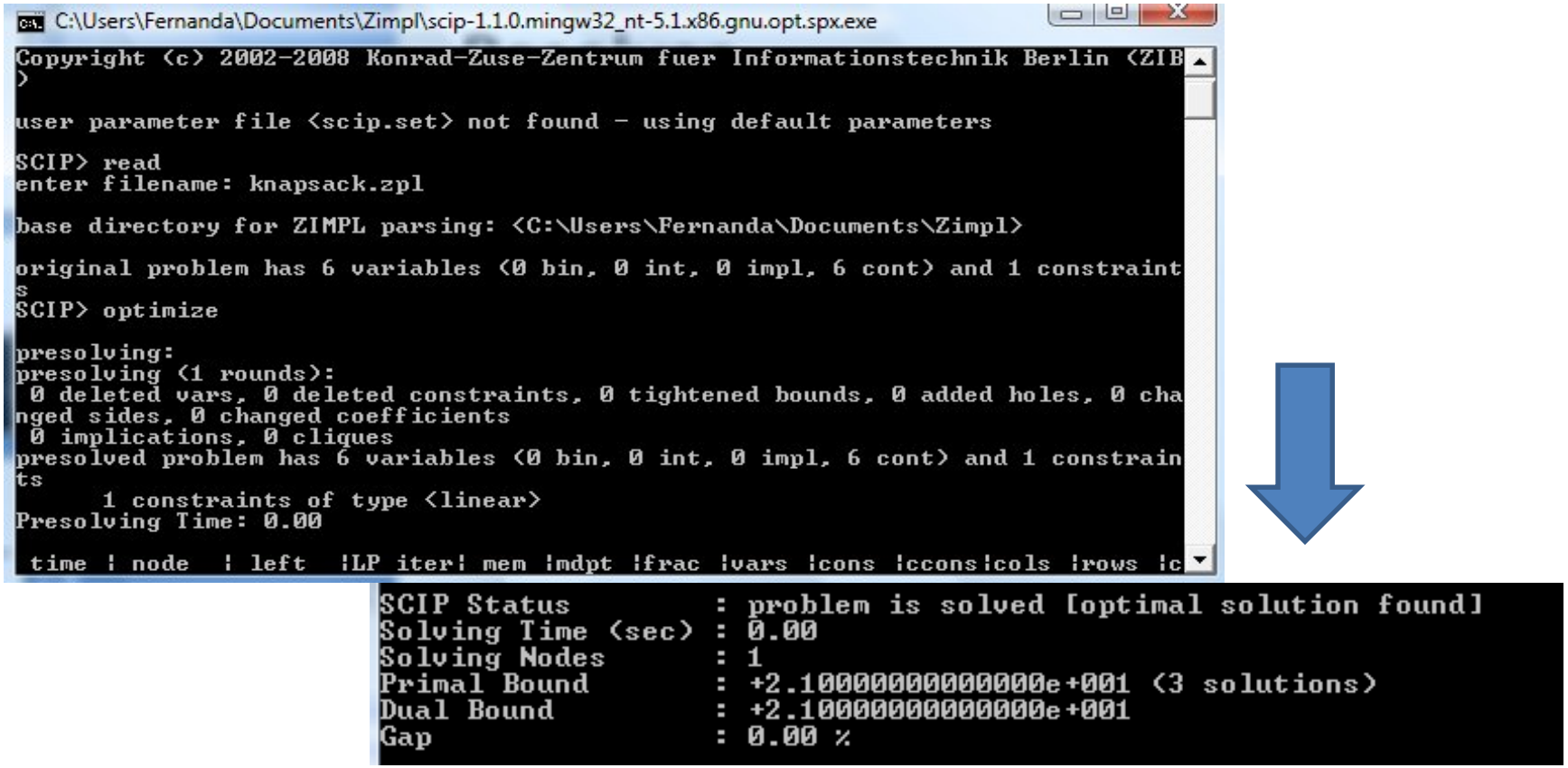
Resolver

Ejecutar SCIP

SCIP> read

Enter filename: knapsack.zpl

SCIP> optimize



Ver Solución

SCIP> display

SCIP\display> solution

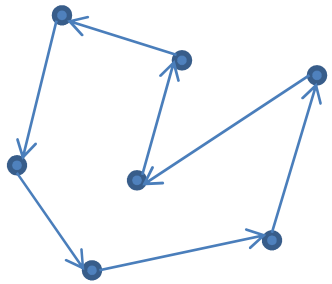
Ejemplo: TSP

$$Min \sum_{i,j \in V} d_{ij} x_{ij}$$

$$st \sum_{v,i \in V} x_{vi} + \sum_{j,v \in V} x_{jv} = 2 \quad \forall v \in V$$

$$\sum_{i,j \in P} x_{ij} \leq Card(P) - 1 \quad \forall \; 2 \leq Card(P) \leq Card(V) - 2$$

$$x_{ij} \in \{0,1\}$$



Ejemplo: TSP

tsp.zpl

```
set V := {read "tsp.txt" as "<1s>" comment "#"};
set E := {<i,j> in V*V with i < j};
set P[ ] := powerset(V);
set K := indexset(P);

param px[V]:= read  "tsp.txt" as "<1s> 2n" comment "#";
param py[V]:= read  "tsp.txt" as "<1s> 3n" comment "#";

defnumb dist(a,b):= sqrt((px[a]-px[b])^2 + (py[a]-py[b])^2);

var x[E] binary;

minimize
cost: sum <i,j> in E : dist(i,j) * x [i, j];

subto two_connected: forall <v> in V do
(sum <v,j> in E : x [v,j]) + (sum <i,v> in E : x [i,v] ) == 2;

subto no_subtour: forall  <k> in K with card (P[k]) > 2 and card (P[k]) < card (V) - 2 do
sum <i,j> in E with <i> in P[k] and <j> in P[k] : x [i,j] <= card(P[k]) - 1;
```

→ Leer ciudades

→ Generar arcos únicos entre par de ciudades

→ Todos los subconjuntos de ciudades

→ Conjunto de índices para cada subconjunto

→ Leer coordenadas

→ Función distancia entre dos ciudades

→ Variable de decisión, 1 si se utiliza el arco <i,j>

→ Función Objetivo

→ Cada ciudad esta conectada por 2 arcos

→ Eliminar subtours

Ejemplo: TSP

tsp.txt

```
# City X Y
Berlin 5251 1340
rankfurt 5011 864
Leipzig 5133 1237
Heidelberg 4941 867
Karlsruhe 4901 840
Hamburg 5356 998
Bayreuth 4993 1159
Trier 4974 668
Hannover 5237 972
```

Ejemplo: TSP

Solución

```
presolved problem has 36 variables (36 bin, 0 int, 0 impl, 0 cont) and 429 const
raints
    336 constraints of type <knapsack>
      9 constraints of type <linear>
     84 constraints of type <logicor>
Presolving Time: 0.00
SCIP Status        : problem is solved [optimal solution found]
Solving Time (sec) : 0.02
Solving Nodes      : 1
Primal Bound       : +1.77810038634643e+003 (1 solutions)
Dual Bound         : +1.77810038634643e+003
Gap                : 0.00 %
```

```
SCIP/display> solution

objective value:                 1778.10038634643
x$Berlin$Leipzig                                1   (obj:156.630137585332)
x$Berlin$Hamburg                                1   (obj:357.755503102328)
x$Heidelberg$Karlsruhe                          1   (obj:48.2597140480546)
x$Karlsruhe$Trier                               1   (obj:186.850207385488)
x$Hamburg$Hannover                              1   (obj:121.807224744676)
x$Bayreuth$Leipzig                              1   (obj:160.262285020525)
x$Bayreuth$Heidelberg                           1   (obj:296.593998590666)
x$Trier$rankfurt                                1   (obj:199.461775786741)
x$Hannover$rankfurt                             1   (obj:250.479540082618)
```

Berlin-Leipzig-Bayreuth-Heidelberg-Karlsruhe-Trier-Frankfurt-Hannover-Hamburg-Berlin

Ejemplo: Flujo Máximo

En este problema se desea encontrar la cantidad máxima de flujo que puede circular en la red desde el nodo de salida s al de entrada e.

Maximizar F

s.a.
$$\sum_{k\in D(j)} x_{jk} - \sum_{i\in A(j)} x_{ij} = \begin{cases} F & \text{si} \quad j = s, \\ 0 & \text{si} \quad j \neq s, e, \\ -F & \text{si} \quad j = e, \end{cases}$$

$$0 \leq x_{ij} \leq q_{ij} \qquad \forall i, j = 1, 2, \ldots, n$$

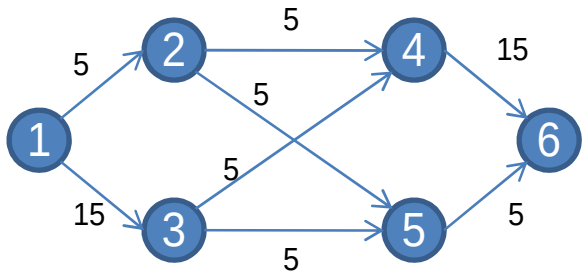
A(j) y D(j) son los conjuntos de nodos que son orígenes y destinos de arcos de entrada y salida:

$$A(j) = \{i \mid i \in N; (i, j) \in L\}$$

$$D(j) = \{k \mid k \in N; (j, k) \in L\}$$

Ejemplo: Flujo Máximo

flujomaximo.zpl




```
param N := read "mfex.net" as "3n" comment "c" use 1;
param A := read "mfex.net" as "4n" comment "c" use 1;

set NODES := {1..N};
param S := read "mfex.net" as "2n" comment "c,a,p" use 1;
param T := read "mfex.net" as "2n" comment "c,a,p" skip 1 use 1;

set ARCS := { read "mfex.net" as "<2n,3n>" comment "c,n,p"};
param Max[ARCS] := read "mfex.net" as "<2n,3n> 4n" comment "c,n,p";

var Flow[ARCS] >=0;         # Flujo
var F >=0;     # Flujo MAXIMO

#El objetivo es maximizar
maximize FLOW:    F;

# Respetar capacidades
subto CAP:  forall <i,j> in ARCS do Flow[i,j]<=Max[i,j];

# Continuidad de Flujo
subto CONT:  forall <i> in NODES with <i> in NODES\{S,T} do sum <i,k> in ARCS : Flow[i,k] - sum <j,i> in ARCS : Flow[j,i] == 0;

# Continuidad de Flujo
subto BEG:  sum <S,k> in ARCS : Flow[S,k] - sum <j,S> in ARCS : Flow[j,S] == F;

# Continuidad de Flujo
subto END:  sum <T,k> in ARCS : Flow[T,k] - sum <j,T> in ARCS : Flow[j,T] == -F;
```

Ejemplo: Flujo Máximo

flujomaximo.net

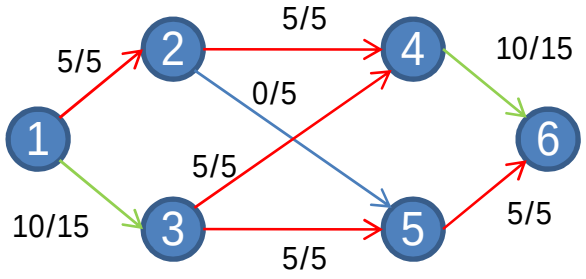
```
c This is a simple example file to demonstrate the DIMACS
c input file format for maximum flow problems. The solution
c vector is [5,10,5,0,5,5,10,5] with cost at 15.
c Problem line  max/min nodes arcs
p max 6 8
c source n ID Which (s/t)
n 1 s
c sink
n 6 t
c Arc descriptor lines (from, to, capacity)
a 1 2 5
a 1 3 15
a 2 4 5
a 2 5 5
a 3 4 5
a 3 5 5
a 4 6 15
a 5 6 5
c
c End of file
```

Ejemplo: Flujo Máximo

Solución

```
SCIP Status        : problem is solved [optimal solution found]
Solving Time (sec) : 0.00
Solving Nodes      : 1
Primal Bound       : +1.50000000000000e+001 (2 solutions)
Dual Bound         : +1.50000000000000e+001
Gap                : 0.00 %
```

```
objective value:                                 15
Flow#1#2                                          5   (obj:0)
Flow#1#3                                         10   (obj:0)
Flow#2#4                                          5   (obj:0)
Flow#3#4                                          5   (obj:0)
Flow#3#5                                          5   (obj:0)
Flow#4#6                                         10   (obj:0)
Flow#5#6                                          5   (obj:0)
F                                                15   (obj:1)
```



Dudas y/o Comentarios

Ma. Fernanda Bravo
mariabra@ing.uchile.cl