Universidad Politécnica de Cataluña

Facultad de Matemáticas y Estadística

Ejercicio de modelización

Flujos en redes - Práctica I

Autor:
Eduardo Peña - Laia Pomar

Octubre 2019

1. Problema de flujo máximo

Sea $G = (N, A)$ el grafo definido por el conjunto de nodos $N = \{1,...,n\}$ y un conjunto de arcos $A \subseteq N \text{x} N$. En este grafo cada arco tiene asociado una cierta capacidad $b_{i,j}$. El fin de este problema es maximizar el flujo que se puede trasladar a partir de un cierto nodo origen (1) y un cierto nodo destino (n), teniendo en consideración que el flujo que puede circular por cada arco (i, j) está acotado por $b_{i,j}$.

1.1. Formulación matemática del problema

A continuación se ha formulado el problema como un problema de programación lineal (PL). Nuestra formulación tiene como objetivo maximizar v, que representa la cantidad de flujo total del problema, y utilizaremos las variables x_{ij} como la cantidad de flujo que circula por el arco (i, j).

$$
\begin{cases}
\text{máx} \quad v \\
s.a. \displaystyle\sum_{\{k:(i,k)\in A\}} x_{ik} - \sum_{\{k:(k,j)\in A\}} x_{kj} = 0, \quad \forall i \in N \setminus \{s,t\} \\
\displaystyle\sum_{\{i:(i,n)\in A\}} x_{in} = v \\
\displaystyle\sum_{\{j:(1,j)\in A\}} x_{1j} = v \\
x_{ij} \leq b_{ij} \quad \forall (i,j) \in A \\
x_{ij} \geq 0 \quad \forall (i,j) \in A, \ v \geq 0
\end{cases}
$$

1.2. Modelo AMPL

Para encontrar una solución a este problema será necesario la utilización y la implementación de AMPL, y por esa razón, se ha trasladado la formulación de este problema a este lenguaje de programación, culminando en el resultado siguiente:

```
#Problema de flujo maximo

#Numero de nodos
param n;
set NODOS:=1..n;
set ARCOS within {NODOS,NODOS};
param capacidad{ARCOS}>=0;
var x{(i,j) in ARCOS}>=0, <=capacidad[i,j], integer;
var v;

#Funcion objetivo
maximize funcio: v;

#Restricciones
subject to restriccion_flujo {k in NODOS: k != 1 and k != n}:
  (sum{(k,j) in ARCOS} x[k,j]-sum{(i,k)in ARCOS}x[i,k]) = 0;
subject to restriccion_salida :
  sum{(i,n) in ARCOS} x[i,n] = v;
subject to resticcion_entrada :
  sum {(1,i) in ARCOS} x[1,i] = v;
```

2. Problema de caminos mínimos

Sea $G = (N,A)$ el grafo definido por el conjunto de nodos $N = \{1,...,n\}$ y un conjunto de arcos $A \subseteq N \text{x} N$. En este grafo, cada arco estará relacionado con un cierto coste $c_{i,j}$. El objetivo es encontrar de entre todas las rutas que unen el nodo origen (1) y el nodo destino (n), la que tenga la suma mínima de costes de los arcos que configuran dicha ruta.

2.1. Formulación matemática del problema

A continuación se ha vuelto a formular el problema como un problema de programación lineal (PL). En este caso queremos minimizar la función objetivo, que consiste en el sumatorio de el coste asociada a cada arco $x_{i,j}$ por el que se pase. $x_{i,j}$ es una variable binaria, que es 1 si el camino trazado del nodo 1 al n pasa por este arco y 0 alternativamente.

$$
\begin{cases}
\text{mín} \displaystyle\sum_{\{(i,j)\in A\}} c_{ij}x_{ij} \\
s.a. \displaystyle\sum_{\{k:(i,k)\in A\}} x_{ik} - \sum_{\{k:(k,j)\in A\}} x_{kj} = 0, \quad \forall k \in N \setminus \{1,n\} \\
\displaystyle\sum_{\{j:(1,j)\in A\}} x_{1j} = 1 \\
\displaystyle\sum_{\{i:(i,n)\in A\}} x_{in} = 1 \\
x_{ij} \in \{0,1\} \quad \forall (i,j)
\end{cases}
$$

2.2. Modelo AMPL

Una vez más, se ha traducido el modelo de la formulación anterior al lenguaje AMPL, para su posterior utilización con el objetivo de encontrar una solución al problema asignado. El resultado es el siguiente:

```
#Problema de caminos minimos

#Numero de nodos
param n;
set NODOS:= 1..n;
set ARCOS within{NODOS,NODOS};
param coste{ARCOS}>=0;
var x{(i,j) in ARCOS}>=0,binary;

#Funcion objetivo
minimize total_coste:
sum{(i,j) in ARCOS} coste[i,j]*x[i,j];

#Restricciones
subject to restriccion_nodos{k in NODOS: k != 1 and k != n}:
  sum{(k,j) in ARCOS} x[k,j] - sum{(i,k) in ARCOS} x[i,k] = 0;
subject to restriccion_entrada:
  sum{(1,j) in ARCOS} x[1,j] = 1;
subject to restriccion_salida:
  sum{(i,n) in ARCOS} x[i,n] = 1;
```

3. Datos

Los archivos de datos que nos han sido asignados son el 1.DAT y el 5.DAT, que contienen una matriz de incidencias nodos - arcos, un vector b de capacidades máximas y un vector c de costes. La información de estos archivos dan lugar a los siguientes grafos dirigidos (Nótese que cada arista tiene una tupla que, tiene como primer componente su capacidad asociada y, como segundo, su coste):

Problema de flujos en redes

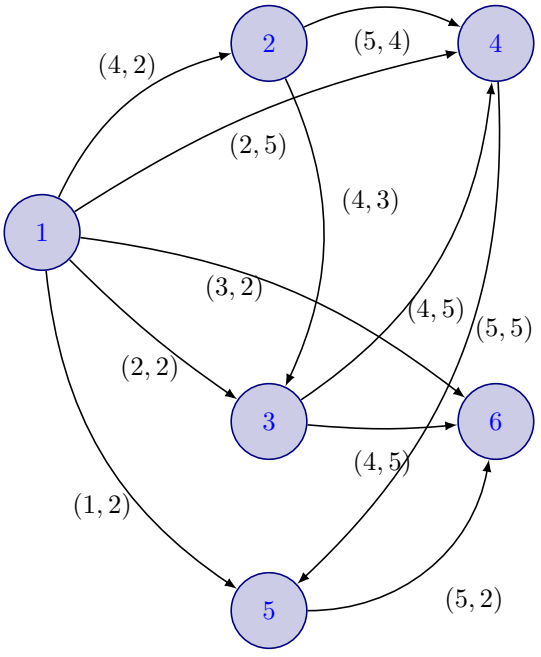
Matriz de incidencias nodos arcos

1 1 1 1 1 0 0 0 0 0 0
0-1 0 0 0 1 1 0 0 0 0
0 0-1 0 0-1 0 1 1 0 0
0 0 0-1 0 0-1 0-1 1 0
-1 0 0 0 0 0 0 0 0-1 1
0 0 0 0-1 0 0-1 0 0-1

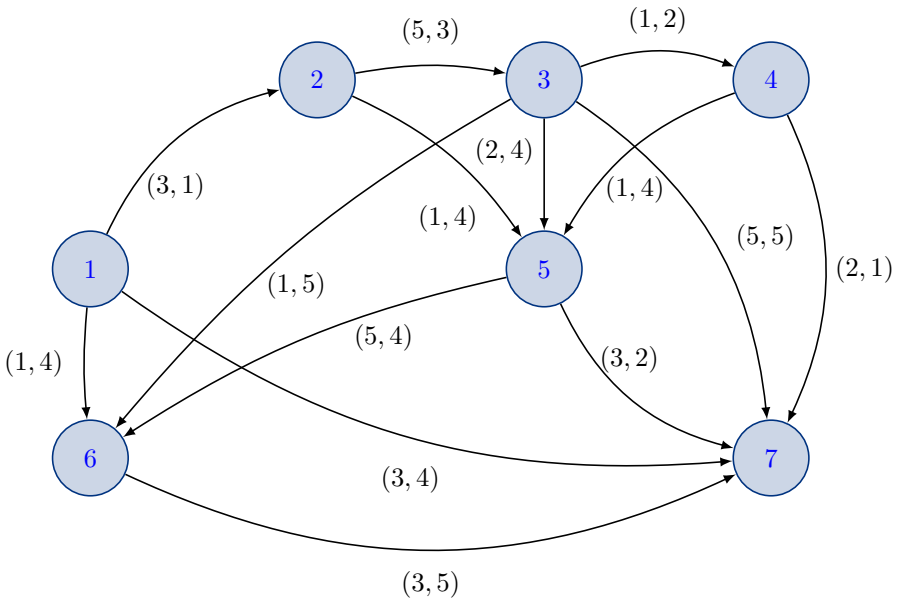
b = (1, 4, 2, 2, 3, 4, 5, 4, 4, 5, 5)
c = (2, 2, 2, 5, 2, 3, 4, 5, 5, 5, 2)

Ejemplo de archivo de datos asignados a cada alumno, en este caso es el 1.DAT.

3.1. Grafo Datos 1



3.2. Grafo Datos 5



4. Programa en C++

Para solucionar esta práctica, se ha decidido hacer un programa escrito en C++ que se encontrará en el anexo de este informe. El Programa al ejecutarse solicita el nombre del fichero de datos que se desee introducir, lee dicho fichero y crea (con el nombre que haya indicado el usuario cuando se pide en la terminal) los archivos .dat y .run con los datos necesarios para ejecutar el modelo. Seguidamente ejecuta el archivo .run llamando al ejecutable de ampl des del programa, y en la terminal se imprimen los resultados del problema.

Nota: Para el buen funcionamiento del programa, deben estar guardados en la misma carpeta: El modelo de los dos problemas, el fichero de datos sin modificar establecido para cada alumno, y el ejecutable de ampl.

4.1. Ejemplos de archivo .dat y .run

Estos son unos ejemplos significativos de los ficheros .dat y .run que crea nuestro programa para su posterior utilización.

```
reset;
#Problema flujo
model fluxmax.mod;
data flujo5.dat;
option solver cplex;
solve;
display x;
reset;
#Problema caminos
model caminsminims.mod;
data caminos5.dat;
option solver cplex;
solve;
display x;
```

```
param n:=6;
param: ARCOS: capacidad :=
1 5 1
1 2 4
1 3 2
1 4 2
1 6 3
2 3 4
2 4 5
3 6 4
3 4 4
4 5 5
5 6 5
```

```
param n:=6;
param: ARCOS: coste :=
1 5 2
1 2 2
1 3 2
1 4 5
1 6 2
2 3 3
2 4 4
3 6 5
3 4 5
4 5 5
5 6 2
```

Ejemplos de .run y .dat del problema 1 y 2 respectivamente.

5. Resultados

La solución a los problemas provista pro nuestro programa utilizando el solver Cplex son las dispuestas a continuación. Además de eso, el cuadro consecuente, representa como actuaría el programa en la terminal utilizando el archivo de datos 1.DAT.

```
Introduzca nombre del fichero:
> 1.DAT
Introduzca nombre del fichero P.flujo máximo
> flujo1.dat
Introduzca nombre del fichero P.caminos mínimos
> caminos1.dat
CPLEX 12.9.0.0: optimal integer solution; objective 12
0 MIP simplex iterations
0 branch-and-bound nodes
x :=
1 2   4
1 3   2
1 4   2
1 5   1
1 6   3
2 3   4
2 4   0
3 4   2
3 6   4
4 5   4
5 6   5
;
CPLEX 12.9.0.0: optimal integer solution; objective 2
0 MIP simplex iterations
0 branch-and-bound nodes
x :=
1 2   0
1 3   0
1 4   0
1 5   0
1 6   1
2 3   0
2 4   0
3 4   0
3 6   0
4 5   0
5 6  0
;
```

6. Anexo

El programa en C++ creado para la práctica.

```
#include<iostream>
#include<fstream>
#include<iomanip>
#include<string>
#include<vector>
using namespace std;
ifstream myfile;
ofstream fileout;

string flujo;
string caminos;
vector<pair<int,int>> VP;
vector<int> c;
vector<int> b;
int arcos, nodos;

//Lee el archivo de datos inciales
void modif_data() {
  VP = vector<pair<int,int>> (0);
  string s;
  for(int i = 0; i < 5; ++i ){
    getline(myfile,s);
  }
  int h;
  for(h = 1; s.size() > 1; ++h){
    int n = s.size();
    int k = 0;
    pair<int,int> a;
    for(int i = 0; i < n; ++i){
      if(s[i] == '1'){
        if (h == 1)VP.push_back(a);
        VP[k].first = h;
        ++k;
      }
      else if(s[i] == '-'){
        if (h == 1)VP.push_back(a);
        VP[k].second = h;
        ++k;
        ++i;
      }else if(s[i]== '0') {
        if(h == 1)VP.push_back(a);
        ++k;
      }
    }
  getline(myfile,s);
  }
  nodos = h-1;
  getline(myfile,s);
  arcos = VP.size();
  b = vector<int> (arcos);
  int i = 6;
  for(int j = 0; j < arcos; ++j){
    b[j] = s[i] - '0';
    i += 3;
  }
  getline(myfile,s);
  c = vector<int> (arcos);
  i = 6;
  for(int j = 0; j < arcos; ++j){
    c[j] = s[i] - '0';
    i += 3;
  }

}

```

```
66 //Genera archivos de datos compatibles con nuestro modelo de AMPL
67 void generar_datos(){
68   fileout.open(flujo);
69   fileout << "param n:=" << nodos << ";" << endl;
70   fileout << "param: ARCOS: capacidad :=" << endl;
71   for(int i = 0; i < arcos ; ++i){
72     fileout << VP[i].first << " " << VP[i].second << " " << b[i] << endl;
73   }
74   fileout.close();
75   fileout.open(caminos);
76   fileout << "param n:=" << nodos << ";" << endl;
77   fileout << "param: ARCOS: coste :=" << endl;
78   for(int i = 0; i < arcos ; ++i){
79     fileout << VP[i].first << " " << VP[i].second << " " << c[i] << endl;
80   }
81   fileout.close();
82 }
83
84 //Genera un archivo run que usa los ficheros de datos generados
85 void generar_run(){
86   fileout.open("Practica1.run");
87   fileout << "reset;" << endl;
88   fileout << "#Problema flujo" << endl;
89   fileout << "model fluxmax.mod;" << endl;
90   fileout << "data " << flujo << ";" << endl;
91   fileout << "option solver cplex;" << endl << "solve;" << endl;
92   fileout << "display x;" << endl;
93   fileout << "reset;" << endl;
94   fileout << "#Problema caminos" << endl;
95   fileout << "model caminsminims.mod;" << endl;
96   fileout << "data " << caminos << ";" << endl;
97   fileout << "option solver cplex;" << endl << "solve;" << endl;
98   fileout << "display x;" << endl;
99   fileout.close();
100 }
101
102 int main(){
103   string nombre;
104   cout << "Introduzca nombre del fichero:" << endl;
105   //Se refiere al nombre del archivo que contiene los datos del problema
106   while (cin >> nombre){
107     myfile.open (nombre);
108     modif_data();
109     cout << endl;
110     myfile.close();
111     cout << "Introduzca nombre del fichero P.flujo mximo" << endl;
112     cin >> flujo;
113     cout << "Introduzca nombre del fichero P.caminos mnimos" << endl;
114     cin >> caminos;
115     generar_datos();
116     generar_run();
117
118     //Ejecuta el archivo run des de la terminal de AMPL
119     system("./ampl Practica1.run");
120     //NOTA: para que esto funcione es necesario que los ficheros empleados y
121     //el ejecutable de AMPL esten en el mismo directorio
122
123     cout << "Introduzca nombre del fichero:" << endl;
124   }
125 }
```