Universidad de Buenos Aires
Facultad De Ingeniería
Año 2022 - 2do Cuatrimestre

Modelos y Optimización I (71.14)

TRABAJO PRÁCTICO Nº
TEMA:
FECHA:

INTEGRANTES:

Paredes Ramirez, Luis José - #104851
<lparedesr@fi.uba.ar>

Índice

1. Enunciado

2. Impresión del problema

Lo primero que notamos al correr el código es la cantidad de tiempo que toma en correr. Despues de dejarlo correr durante 52 minutos todavia no terminó de ejecutarse, por lo que decidimos abortar la ejecucion y analizar hasta donde llego.

Dado que el problema se clasifica de NP-Hard el código podria tardar dias en ejecutarse (suponiendo que la maquina es rápida)

Se trata de un problema de coloreo estandar.

2.1. Análisis del código

Hago un análisis de lo que ejecuta el codigo llevandolo a ecuaciones matematicas

```
int n = ...;
int limiteColores = n;
int e = ...;
range nodos = 1 .. n;
range colores = 1 .. limiteColores;
range aristas = 1 .. e;

tuple peso {
  int i;
  int w;
}

peso weights[nodos] = ...;

tuple edge {
  int i;
  int j;
}

edge edges[aristas] = ...;


dvar boolean x[nodos, colores];
dvar int pesoColor[colores];

minimize
  sum(k in colores) pesoColor[k];

subject to {
  forall ( i in nodos )
    todo_coloreado:
```

```
      sum ( k in colores ) x[i,k] == 1;

  forall ( i in nodos )
    forall ( k in colores )
      peso_color:
      pesoColor[k] >= weights[i].w * x[i,k];

  forall ( e in aristas )
    incompatibles:
        forall ( k in colores )
          x[edges[e].i,k]+x[edges[e].j,k]<=1;
        /*
  forall ( i in 2 .. limiteColores )
    simetria:
      pesoColor[i-1]>= pesoColor[i];
      */
}

main {
  var mod = thisOplModel.modelDefinition;
  var dat = thisOplModel.dataElements;
  var cplex1 = new IloCplex();
  var opl = new IloOplModel(mod, cplex1);
  opl.addDataSource(dat);
  opl.generate();

  if (cplex1.solve()) {
    writeln("solution: ", cplex1.getObjValue(), " /size: ", dat.n, " /time: ",
        cplex1.getCplexTime());

        for ( i in opl.nodos )
          for ( k in opl.colores ){
            if (opl.x[i][k] == 1)
            {
              writeln("Nodo ", i, ": ", k);
            }
          }
/*
    for (i in opl.cities) {
      if (i == 1)
        writeln("Ciudad ", i, ": ", -1);
      else
        writeln("Ciudad ", i, ": ", opl.u[i]);
    }*/
    opl.end()
    cplex1.end()
```

```
    }
}
```

2.2. Variables

- n: Número de nodos (int)
- limiteColores: Cantidad máxima de colores a utilizar (int)
- e: Número de Aristas (int)
- nodos: rango de nodos (1,2, ... , n)
- colores: rango de colores (1,2, ... ,limColores)
- aristas: rango de aristas (1,2, ... , e)
- wights:
- peso:
- edge: Vector de Aristas (arista 1, arista 2, ... , arista e)
- X:
- pesoColor: peso que tiene cada color (vector)

2.3. Funcional

```
    minimize
  sum(k in colores) pesoColor[k];
```

$$\text{Min } Z \leftarrow \sum_{k \in colores} pesoColor_k$$

Se quiere minimizar el peso total de los colores.

Cada color va a tener un peso propio en orden creciente (color 1 tiene peso 1, color 2 peso 2, y asi sucesivamente) por lo que el modelo tratara de pintar con el menor número de colores posible.

2.4. Restricciones

3. Objetivo

4. Hipótesis y supuestos

5. Definición de variables

6. Modelo de programación lineal

7. Resolución gráfica

8. Resolución por software

9. Informe de la solución óptima

10. Anexo