Consegna Laboratorio 8

PEPE SVEVA - 1743997
SCOTTI FRANCESCO - 1758391
POGGI MATTIA- 1762074

Descrizione Problema:

Un'azienda di spedizioni deve effettuare delle consegne a clienti dislocati in alcune città del nord e del centro italia, partendo e ritornando al magazzino localizzato nella città di Firenze. I clienti devono essere serviti tutti ed il furgone utilizzato per le consegne ha un costo di 0.50 € per chilometro percorso. L'azienda vuole determinare il tour di costo minimo che dovrà effettuare il furgone per servire tutti i clienti.

	Genova	Milano	Ferrara	Bologna	Firenze	Ancona	Perugia	Pescara	Roma
Genova	0	140.3	277.8	249	230.5	450	366.9	591.8	482.3
Milano	140.3	0	254.3	212.3	298.8	417.5	453.3	576.3	575.2
Ferrara	277.8	254.3	0	46.3	146.8	246.6	301.1	405.1	423.1
Bologna	249	212.3	46.3	0	102.1	210.7	256.4	369.4	378.4
Firenze	230.5	298.8	146.8	102.1	0	310	156.7	459.2	278.7
Ancona	450	417.5	246.6	210.7	310	0	135.5	169.3	314.2
Perugia	366.9	453.3	301.1	256.4	156.7	135.5	0	257.1	172.5
Pescara	591.8	576.3	405.1	369.4	459.2	169.3	257.1	0	207.9
Roma	482.3	575.2	423.1	378.4	278.7	314.2	172.5	207.9	0

1 Formula matematicamente il problema di determinare il tour di costo minimo delle città riportate nella tabella sovrastante

Variabili Decisionali:

$$y_{ij} = \begin{cases} 1 \\ 0 \end{cases} \qquad i = 1, ..., 9 \; j = 1, ..., 9$$

Variabile indicatrice di quali archi appartengono al tour
$y_{ij} = 1$ se l'arco (i,j) è nel tour
$y_{ij} = 0$ altrimenti

Formulazione Modello

$$\begin{cases} \min \sum_{(i,j)\in E} c_{ij} y_{ij} \\ \sum_{(i,j)\in A(i)} y_{ij} \leq 2 & \forall i \in N \\ \sum_{(i,j)\in A(S)} y_{ij} \leq |S| - 1 & \forall S \subset \{1, 2, ...n\} \\ \sum_{(i,j)\in E} y_{ij} = n \\ y_{ij} \in \{0, 1\} & \forall (i, j) \in E \end{cases}$$

n=9, che sono i nodi del mio problema, quindi le città.
m sono gli archi e E è l'insieme degli archi.
A(i) è l'insieme degli spigoli che hanno un estremo nel vertice i.
A(S) è l'insieme degli spigoli formati dai nodi nell'insieme S.

c_{ij} $\rightarrow$ corrisponde al costo, cioè 0.5*distanza km dell'arco (i,j)

$\sum_{(i,j)\in A(i)} y_{ij} \leq 2 \quad \forall i \in N$ $\rightarrow$ **vincolo che identifica che ogni nodo deve avere grado al più pari a 2**

$\sum_{(i,j)\in A(S)} y_{ij} \leq |S| - 1 \quad \forall S \subset \{1, 2, ...n\}$ $\rightarrow$ **Non ci devono essere cicli formati da 2,3,..n-1 nodi**

$\sum_{(i,j)\in E} y_{ij} = n$ $\rightarrow$ **In totale dobbiamo andare a visitare n spigoli**

Il problema che stiamo considerando corrisponde a quello del commesso viaggiatore *simmetrico*, quindi di un grafo `non` orientato.

2 Utilizzando l'esempio presente in Cplex Studio denominato "TravelingSalesmanProblem", implementa il modello in Opl scrivendo il file .mod ed il file .dat, aggiungendo una funzione di pre-processing per il calcolo dei costi associati agli archi ed inserendo una funzione main che generi il modello e lo risolva iterativamente eseguendo la funzione di post-processing per l'aggiunta dei vincoli di eliminazione dei subtours

lab8.dat :

```
n = 9;
dist=[140.3
277.8
249
230.5
450
366.9
591.8
482.3
254.3
212.3
298.8
417.5
453.3
576.3
575.2
46.3
146.8
246.6
301.1
405.1
423.1
102.1
210.7
256.4
369.4
378.4
310
156.7
459.2
278.7
135.5
169.3
314.2
257.1
172.5
207.9
];

subtours={};
```

lab8.mod

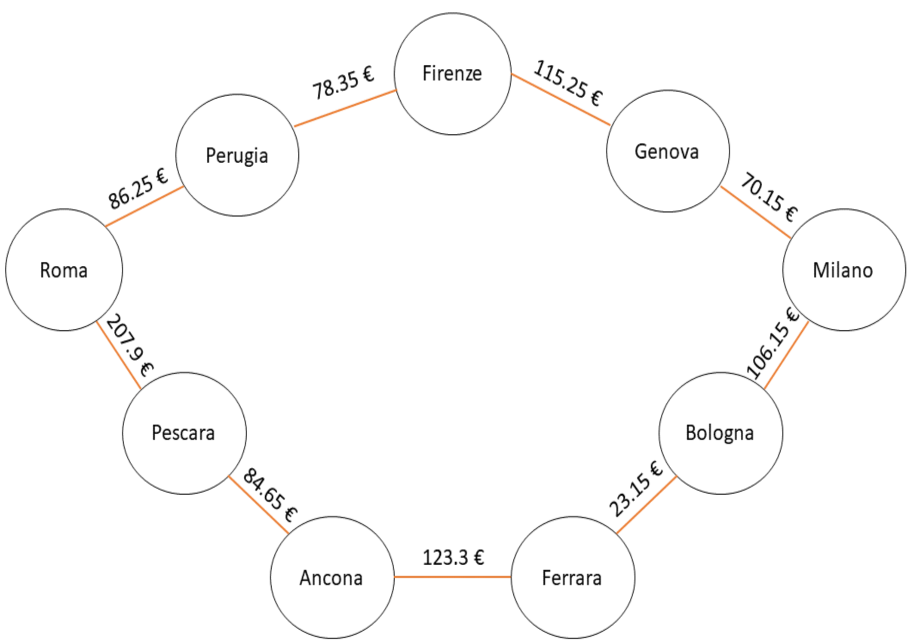
```
/** Cities **/
int       n        = ...;
range     Cities   = 1..n;

/** Edges -- sparse set **/
tuple        edge        {int i; int j;}
```

```
setof(edge) Edges       = {<i,j> | ordered i,j in Cities};
float        dist[Edges] = ...;
float      p[Edges];

/** Decision variables **/
dvar boolean x[Edges];

tuple Subtour { int size; int subtour[Cities]; }
{Subtour} subtours = ...;

/**PRE-PROCESSING**/
execute{
  for (var i in Edges)
  (p[i]=0.5*dist[i]);
}


/** Objective **/
minimize sum (<i,j> in Edges) p[<i,j>]*x[<i,j>];
subject to {

  /** Each city is linked with two other cities **/
  forall (j in Cities)
    sum (<i,j> in Edges) x[<i,j>]
    + sum (<j,k> in Edges) x[<j,k>] == 2;

  /** Subtour elimination constraints **/
  forall (s in subtours)
    sum (i in Cities : s.subtour[i] != 0)
      x[<minl(i, s.subtour[i]), maxl(i, s.subtour[i])>]
      <= s.size-1;

};

/** POST-PROCESSING to find the subtours **/

/** Solution information **/
int thisSubtour[Cities];
int newSubtourSize;
int newSubtour[Cities];

/** Auxiliary information **/
int visited[i in Cities] = 0;
setof(int) adj[j in Cities] =
{i | <i,j> in Edges : x[<i,j>] == 1} union
{k | <j,k> in Edges : x[<j,k>] == 1};

execute {

  newSubtourSize = n;
  for (var i in Cities) { // Find an unexplored node
    if (visited[i]==1) continue;
    var start = i;
    var node = i;
    var thisSubtourSize = 0;
    for (var j in Cities)
      thisSubtour[j] = 0;
```

```
    while (node!=start || thisSubtourSize==0) {
      visited[node] = 1;
      var succ = start;
      for (i in adj[node])
        if (visited[i] == 0) {
          succ = i;
          break;
        }

      thisSubtour[node] = succ;
      node = succ;
      ++thisSubtourSize;
    }

    writeln("Found subtour of size : ", thisSubtourSize);
    if (thisSubtourSize < newSubtourSize) {
      for (i in Cities)
        newSubtour[i] = thisSubtour[i];
        newSubtourSize = thisSubtourSize;
    }
  }
  if (newSubtourSize != n){
    writeln("Best subtour of size ", newSubtourSize);
  }
  var ofile=new IloOplOutputFile("Lab8.txt");
  ofile.writeln("Objective=", cplex.getObjValue());

  for (var e in Edges){
    ofile.writeln("scelto", e, "=", x[e]);
  }
  ofile.close();
}


main {
  var opl = thisOplModel
  var mod = opl.modelDefinition;
  var dat = opl.dataElements;

  var status = 0;
  var it =0;
  while (1) {
    var cplex1 = new IloCplex();
    opl = new IloOplModel(mod,cplex1);
    opl.addDataSource(dat);
    opl.generate();
    it++;
    writeln("Iteration ",it, " with ", opl.subtours.size, " subtours.");
    if (!cplex1.solve()) {
      writeln("ERROR: could not solve");
      status = 1;
      opl.end();
      break;
    }
    opl.postProcess();
    writeln("Current solution : ", cplex1.getObjValue());
```

```
      if (opl.newSubtourSize == opl.n) {
        opl.end();
        cplex1.end();
        break; // not found
      }

      dat.subtours.add(opl.newSubtourSize, opl.newSubtour);
      opl.end();
      cplex1.end();
    }

    status;
  }
```

3 Rappresenta graficamente, con un programma a scelta, il tour ottimo riportandone il costo associato.



La soluzione della funzione obiettivo: 791.2 €

Risultati.txt

```
Objective=791.2
arco 12=1 costo 70.15
arco 13=0 costo 138.9
arco 14=0 costo 124.5
arco 15=1 costo 115.25
arco 16=0 costo 225
arco 17=0 costo 183.45
arco 18=0 costo 295.9
arco 19=0 costo 241.15
arco 23=0 costo 127.15
arco 24=1 costo 106.15
arco 25=0 costo 149.4
arco 26=0 costo 208.75
arco 27=0 costo 226.65
arco 28=0 costo 288.15
arco 29=0 costo 287.6
arco 34=1 costo 23.15
arco 35=0 costo 73.4
arco 36=1 costo 123.3
arco 37=0 costo 150.55
arco 38=0 costo 202.55
arco 39=0 costo 211.55
arco 45=0 costo 51.05
arco 46=0 costo 105.35
arco 47=0 costo 128.2
arco 48=0 costo 184.7
arco 49=0 costo 189.2
arco 56=0 costo 155
arco 57=1 costo 78.35
arco 58=0 costo 229.6
arco 59=0 costo 139.35
arco 67=0 costo 67.75
arco 68=1 costo 84.65
arco 69=0 costo 157.1
arco 78=0 costo 128.55
arco 79=1 costo 86.25
arco 89=1 costo 103.95
```