DM826 – Spring 2012

Modeling and Solving Constrained Optimization Problems

Exercises

Set Variables
SONET Problem

Marco Chiarandini

Department of Mathematics & Computer Science
University of Southern Denmark

[*Partly based on slides by Stefano Gualandi, Politecnico di Milano*]

Sonet **problem**

Optical fiber network design

Sonet problem

Input: weighted undirected demand graph $G = (N, E; d)$, where each node $u \in N$ represents a client and weighted edges $(u, v) \in E$ correspond to traffic demands of a pair of clients.

Two nodes can communicate, only if they join the same ring; nodes may join more than one ring. We must respect:

- maximum number of rings r
- maximum number of clients per ring a
- maximum bandwidth capacity of each ring c

Task: find a topology that minimizes the sum, over all rings, of the number of nodes that join each ring while clients' traffic demands are met.

Sonet **problem**

Sonet problem

A solution of the SONET problem is an assignment of rings to nodes and of capacity to demands such that

1. all demands of each client pairs are satisfied;
2. the ring traffic does not exceed the bandwidth capacity;
3. at most r rings are used;
4. at most a ADMs on each ring;
5. the total number of ADMs used is minimized.

Sonet: **variables**

- Set variable X_i represents the set of nodes assigned to ring i
- Set variable Y_u represents the set of rings assigned to node u
- Integer variable Z_{ie} represents the amount of bandwidth assigned to demand pair e on ring i.

Sonet: **model**

$$
\begin{aligned}
\min \quad & \sum_{i \in R} |X_i| & \\
\text{s.t.} \quad & |Y_u \cap Y_v| \geq 1, & \forall (u,v) \in E, \\
& Z_{ie} \in \{0, d(e)\}, & \forall e \in E, \\
& Z_{i,(u,v)} > 0 \iff i \in (Y_u \cap Y_v), & \forall i \in R, (u,v) \in E, \\
& u \in X_i \Leftrightarrow i \in Y_u, & \forall \in R, u \in N, \\
& |X_i| \leq a, & \forall i \in R \\
& \sum_{e \in E} Z_{ie} \leq c, & \forall i \in R. \\
& X_i \preceq X_j, & \forall i, j \in R : i < j.
\end{aligned}
$$

```
from numpy import *
from gecode import *
Rings = range(4) # upper bound for amount of rings
Nodes = range(5) # amount of clients
demand = array([[0,1,0,1,1],
               [1,0,1,0,0],
               [0,1,0,0,1],
               [1,0,0,0,0],
               [1,0,1,0,0]])
capacity = [3,2,2,3] # capacity in nodes of possible rings

X = map(lambda r: m.setvar(intset(),0,len(Nodes),0,capacity[r]),Rings) #nodes for r
Y = m.setvars(len(Nodes),intset(),0,len(Rings),0,len(Rings)) # rings for u

# at least two nodes in each ring
for r in Rings:
   cardinality(X[r], IRT_NQ, 1) # implied constraint

for (n1,n2) in combinations(Nodes,2):
   IntVar z(intset(),0,4,1,len(Rings));
   if demand[n1,n2]==1:
     rel(Y[n1], SOT_INTER, Y[n2], SRT_SUP, z)

channel(X,Y)

IntVarArray z(len(Rings), )
for r in Rings:
    cardinality(X[r],z[r])


IntVar adm(0,len(Rings)*len(Nodes)
linear(z, IRT_EQ, adm)
```

Once a variable X[i] has been chosen, we first try to include the node that has the most communication with the nodes already placed in ring *i*,

```
while (!and(i in Rings)(X[i].bound())) {
  selectMin (i in Rings: !X[i].bound())(X[i].getCardinalityVariable().getSize()) {
   set{int} S = X[i].getPossibleSet();
   set{int} R = X[i].getRequiredSet();
   Solver<CP> cp = X[i].getSolver();
   selectMax (e in S: !R.contains(e))(sum(n in R)(demand[e,n])) {
     try<cp> cp.requires(nodesInRing[i],e); | cp.excludes(nodesInRing[i],e);
}
```

References