Optimization/simulation-based risk mitigation in resilient green communication networks Code for optimization procedures

Piotr Chołda and Piotr Jaglarz

To reproduce the steps of the algorithm presented in the paper, save attached files to a common directory and provide required tools.

CONTENTS

The authors are with the AGH University of Science and Technology, Department of Telecommunications, Krakow, Poland. E-mail: {piotr.cholda}@agh.edu.pl.

SOFTWARE REQUIREMENTS

- CPLEX (including OPL Interpreter): http://www-01.ibm.com/software/commerce/optimization/cplex-optimizer/

DL.MOD

This CPLEX script presents the optimization problem of optimal assignment of optical flows if energy profiles in links are concave. The recovery option used by all the demands is based on dedicated link (local) protection 1:1 DL. The energy profiles are approximated with linear segments.

```
// ********************************************/
// OPL 12.6.0.0 Model
// Author: Piotr Cholda, AGH University of Science and Technology
// piotr.cholda@agh.edu.pl
// Creation Date: 2 June 2015
// ********************************************/

float BigM = 100000;

{string} Nodes = ...;

tuple arc
{
  string source;
  string destination;
}

{arc} Arcs with source in Nodes, destination in Nodes = ...;

tuple demand
{
  string source;
  string destination;
}

{demand} Demands with source in Nodes, destination in Nodes = ...;

float Volume[Demands] = ...;

int Path = ...;

range Paths = 1..Path;

int delta[Arcs][Demands][Paths] = ...;

int gamma[Arcs][Arcs][Paths] = ...; //gamma[e][a][p] = 1 if segment p protecting capacity of link a uses link e

dvar boolean flow[Demands][Paths]; // single flow

dvar boolean backup_link_flow_which[Arcs][Paths]; // single backup segment for each link, = 1 if segment p protects capacity of link a

dvar float+ backup_link_flow[Arcs][Paths];

dvar float+ flows_working[Arcs];

dvar float+ flows_backup[Arcs];

dvar float+ flow_summarized[Arcs];

int Number_seg = ...;

range Segments = 1..Number_seg;

float Coeff_a[Segments] = ...;

float Coeff_b[Segments] = ...;

dvar float+ y[Arcs][Segments];

dvar boolean u[Arcs][Segments];

dvar float+ cost_link[Arcs];
```

```
63
64  minimize sum(a in Arcs) cost_link[a];
65
66  subject to{
67
68    forall(d in Demands)
69      sum(p in Paths) flow[d][p] == 1;
70
71    forall(a in Arcs)
72         sum(d in Demands, p in Paths) delta[a][d][p]*flow[d][p]*Volume[d] == flows_working[a];
73
74    forall(a in Arcs)
75      sum(p in Paths) backup_link_flow[a][p] == flows_working[a];
76
77    forall(a in Arcs)
78      sum(p in Paths) backup_link_flow_which[a][p] == 1;
79
80    forall(a in Arcs, p in Paths)
81      backup_link_flow[a][p] <= BigM*backup_link_flow_which[a][p];
82
83    forall(a in Arcs)
84         sum(e in Arcs, p in Paths) gamma[a][e][p]*backup_link_flow[e][p] == flows_backup[a];
85
86    forall(a in Arcs)
87         flows_working[a] + flows_backup[a] == flow_summarized[a];
88
89    forall(a in Arcs)
90      flow_summarized[a] == sum(k in Segments) y[a][k];
91
92    forall(a in Arcs,k in Segments)
93      y[a][k] <= BigM*u[a][k];
94
95    forall(a in Arcs,k in Segments)
96      u[a][k] <= BigM*y[a][k];
97
98    forall(a in Arcs)
99      sum(k in Segments) u[a][k] == 1;
100
101   forall(a in Arcs)
102     cost_link[a] ==  sum(k in Segments) (Coeff_a[k]*y[a][k] + Coeff_b[k]*u[a][k]);
103
104 }
```