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

Software Requirements 2

nr.mod 3

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/

NR.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 a lack of recovery NR. 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; // big constant

{string} Nodes = ...; // set of network nodes

tuple arc
{
  string source;
  string destination;
}

{arc} Arcs with source in Nodes, destination in Nodes = ...; // set of network links

tuple demand
{
  string source;
  string destination;
}

{demand} Demands with source in Nodes, destination in Nodes = ...; // set of demands

float Volume[Demands] = ...; // volume for each demand

int Path = ...;

range Paths = 1..Path;

int delta[Arcs][Demands][Paths] = ...; // delta[e][d][p] as in the Pioro and Medhi book: equal to one if candate path p for demand d uses link e

dvar boolean flow[Demands][Paths]; // non-bifurcated routing

dvar float+ flow_summarized[Arcs]; // total flow on a link

int Number_seg = ...;

range Segments = 1..Number_seg;

float Coeff_a[Segments] = ...; // coefficient a in the segment ax+b that is used for linearization of a concave energy profile

float Coeff_b[Segments] = ...; // coefficient b in the segment ax+b that is used for linearization of a concave energy profile

dvar float+ y[Arcs][Segments];

dvar boolean u[Arcs][Segments];

dvar float+ cost_link[Arcs]; // approximated energy usage in a link

minimize sum(a in Arcs) cost_link[a];

subject to{

  forall(d in Demands)
    sum(p in Paths) flow[d][p] == 1;

  forall(a in Arcs)
```

```
62          sum(d in Demands, p in Paths) delta[a][d][p]*flow[d][p]*Volume[d] == flow_summarized[a]
                ↪];
63
64    forall(a in Arcs)
65      flow_summarized[a] == sum(k in Segments) y[a][k];
66
67    forall(a in Arcs,k in Segments)
68      y[a][k] <= BigM*u[a][k];
69
70    forall(a in Arcs,k in Segments)
71      u[a][k] <= BigM*y[a][k];
72
73    forall(a in Arcs)
74      sum(k in Segments) u[a][k] == 1;
75
76    forall(a in Arcs)
77      cost_link[a] ==  sum(k in Segments) (Coeff_a[k]*y[a][k] + Coeff_b[k]*u[a][k]);
78
79  }
```