

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
sl.mod	3

SOFTWARE REQUIREMENTS

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

SL.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 shared backup link (local) protection SL. The energy profiles are approximated with linear segments.

```

1 // *****/
2 // OPL 12.6.0.0 Model
3 // Author: Piotr Cholda, AGH University of Science and Technology
4 // piotr.cholda@agh.edu.pl
5 // Creation Date: 2 June 2015
6 // *****/
7
8 float BigM = 100000;
9
10 {string} Nodes = ...;
11
12 tuple arc
13 {
14     string source;
15     string destination;
16 }
17
18 {arc} Arcs with source in Nodes, destination in Nodes = ...;
19
20 tuple demand
21 {
22     string source;
23     string destination;
24 }
25
26 {demand} Demands with source in Nodes, destination in Nodes = ...;
27
28 float Volume[Demands] = ...;
29
30 int Path = ...;
31
32 range Paths = 1..Path;
33
34 int delta[Arcs][Demands][Paths] = ...;
35
36 int gamma[Arcs][Arcs][Paths] = ...;
37
38 dvar boolean flow[Demands][Paths]; // single flow
39
40 dvar boolean backup_link_flow_which[Arcs][Paths]; // single backup tunnel
41
42 dvar float+ backup_link_flow[Arcs][Paths];
43
44 dvar float+ flows_working[Arcs];
45
46 dvar float+ flows_backup[Arcs];
47
48 dvar float+ flow_summarized[Arcs];
49
50 int Number_seg = ...;
51
52 range Segments = 1..Number_seg;
53
54 float Coeff_a[Segments] = ...;
55
56 float Coeff_b[Segments] = ...;
57
58 dvar float+ y[Arcs][Segments];
59
60 dvar boolean u[Arcs][Segments];
61
62 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, e in Arcs)
84         sum(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 }

```