On the hop-constrained survivable network design problem with reliable edges

Quentin Botton[a], Bernard Fortz[b], Luis Gouveia[c]

[a]*Louvain School of Management, Université catholique de Louvain, Louvain-la-Neuve, Belgium*
[b]*Département d'Informatique, Faculté des Sciences, Université Libre de Bruxelles, Bruxelles, Belgium*
[c]*DEIO, CIO, Faculdade de Ciências da Universidade de Lisboa, Lisboa, Portugal*

Abstract

In this paper, we study the hop-constrained survivable network design problem with reliable edges. Given a graph with non-negative edge costs and node pairs Q, the hop-constrained survivable network design problem consists of constructing a minimum cost set of edges so that the induced subgraph contains at least K edge-disjoint paths containing at most L edges between each pair in Q. In addition, we consider here a subset of reliable edges that are not subject to failure. We study two variants: a static problem where the reliability of edges is given, and an upgrading problem where edges can be upgraded to the reliable status at a given cost. We adapt for the two variants an extended formulation proposed in Botton, Fortz, Gouveia, Poss 2011 for the case without reliable edges. As before, we use Benders decomposition to accelerate the solving process. Our computational results indicate that these two variants appear to be more difficult to solve than the original problem (without reliable edges). We conclude with an economical analysis which evaluates the incentive of using reliable edges in the network.

Keywords: Network design, Survivability, Hop constraints, Benders decomposition, Branch-and-cut algorithms

1. Introduction

Given an undirected graph $G = (V, E)$ with a nonnegative cost c_{ij} associated to each edge $ij \in E$, and a set Q of node pairs, we want to find a set of edges of minimum cost such that the induced subgraph contains at least K edge-disjoint L-paths between each pair in Q. A path in G is a L-path if it contains no more than L edges. Given two distinct nodes $o, d \in V$, an od-path is a sequence of node-edges $P = (v_0, e_0, v_1, ..., e_{l-1}, v_l)$, where $l \geq 1$, $v_0, v_1, ..., v_l$ are distinct nodes, $v_0 = o$, $v_l = d$, and $e_i = v_i v_{i+1}$ is an edge connecting v_i and v_{i+1} (for $i = 0, ..., l-1$). A collection $P_1, P_2, ..., P_k$ of od-L-paths is called edge-disjoint if any edge ij appears in at most one path. The problem considered is NP-hard [10] and was first studied by Huygens et al. [9] who only consider $L \leq 4$ and $K = 2$. Recently, an extended formulation combined with a Benders decomposition approach to efficiently handle the large number of variables and constraints of the formulation have been proposed and tested in Botton et al. [1]. They consider instances with $1 \leq K \leq 3$ and $3 \leq L \leq 5$.

The survey by Kerivin and Mahjoub [11] describes several variants of this problem as well as techniques for solving them. There are several reasons for requiring edge-disjoint paths and including a limit on the number of edges for each path of each commodity. Briefly, the requirement for using disjoint paths guarantees the existence

Email addresses: quentin.botton@uclouvain.be (Quentin Botton), bernard.fortz@ulb.ac.be (Bernard Fortz), legouveia@fc.ul.pt (Luis Gouveia)

of a working path for each commodity after $K-1$ edge failures and the hop constraints impose a certain level of quality of service, since in most of the routing technologies, delay is caused at the nodes, and thus, it is usual to measure the delay in a path in terms of its number of intermediate nodes, or equivalently, its number of edges (or hops).

In this paper, we study two different versions of the problem with different degrees of reliability associated to the edges of the underlying graph. In the first variant we assume that the edges in a subset $E_R \subset E$ are more reliable but more costly. Essentially, these edges are not prone to failure and the edge-disjoint property must be guaranteed only for the remaining edges. Thus, for a given commodity, all the K paths can simultaneously use each reliable edge.

This variant arises in the context of multi-layered telecommunication networks, for example, a SONET/SDH layer over an optical network layer. A link of the SONET/SDH layer can be considered as a demand that should be routed through a path in the optical layer. If this path is protected against failures, then the link corresponding to this path in the SONET/SDH layer can be viewed as "reliable." Otherwise the link can fail and hence requires some protection against failures at the SONET/SDH layer (see Pióro and Medhi [13]). Zotkiewicz et al. [14] present a polynomial time algorithm to solve the particular case of this variant where $K = 2$.

In the second variant, we give the network provider the option of choosing whether an edge should be reliable or not. That is, the subset E_R corresponds to edges ij that could be upgraded to more reliable edges at a higher cost.

These two variants are obviously NP-Hard since they contain the original problem without reliable edges as particular case.

The paper is organized as follows: In Section 2, we first review the model proposed in [1] for the original case where all edges are assumed to be unreliable and then we show how to adapt this original model to accommodate the new specifications. In Section 3 we show the Benders reformulation of both problems and in Section 4 a branch-and-cut algorithm is described. Computational results are reported in Section 5, and we provide concluding remarks in Section 6.

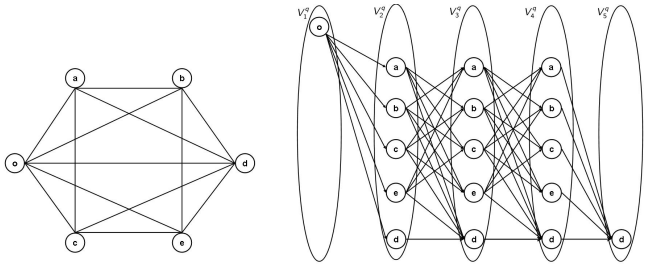
2. Problem definition and formulations

In this section, we start by describing the model for the problem without reliable edges proposed in [1] and then, we show how to modify the model for the two variants previously described.

2.1. Original problem (no reliable edges)

The main idea of the formulation is to model the subproblem associated with each commodity with a directed graph composed of $L+1$ layers as illustrated in Fig. 1.

From the original undirected graph $G = (V, E)$, we create a directed layered graph $G^q = (V^q, A^q)$ for each commodity q, where $V^q = V_1^q \cup \ldots \cup V_{L+1}^q$ with $V_1^q = \{o(q)\}$, $V_{L+1}^q = \{d(q)\}$ and $V_l^q = V \backslash \{o(q)\}$, $l = 2, \ldots, L$. Let v_l^q be the copy of $v \in V$ in the *l-th* layer of graph G^q. Then, the arc sets are defined by $A^q = \{(i_l^q, j_{l+1}^q) \mid ij \in E, i_l^q \in V_l^q, j_{l+1}^q \in V_{l+1}^q, l \in \{1, \ldots, L\}\} \cup \{d(q)^l, d(q)^{l+1}, l \in \{2, \ldots, L\}\}$, see Fig. 1. In the sequel, an (undirected) edge in E



(a) (b)

Figure 1: A basic network (a) and its alternative layered representation (b) when $L = 4$.

with endpoints i and j is denoted ij while a (directed) arc between $i_l^q \in V_l^q$ and $j_{l+1}^q \in V_{l+1}^q$ is denoted by (i, j, l) (the commodity q is omitted in the notation as it is often clear from the context).

Note that each path between $o(q)$ and $d(q)$ in the layered graph G^q is composed of exactly L arcs (hops), which correspond to a maximum of L edges (hops) in the original graph (the horizontal-loop arcs at the bottom of Figure 1 allow paths with less than L arcs). This transformation was first proposed by Gouveia [7] and is motivated by the fact that in the layered graph any path is feasible with respect to the hop-constraints. A set of K-paths satisfying the hop limit can be obtained by sending K units of flow in the layered graph, leading to the following extended formulation:

$$\text{(Hop)} \qquad \min \quad \sum_{ij \in E} c_{ij} z_{ij}$$

$$\text{s.t.} \quad \sum_{j:(j,i,l-1) \in A^q} x_{ji}^{l-1,q} - \sum_{j:(i,j,l) \in A^q} x_{ij}^{l,q} = \begin{cases} -K & if \quad (i = o(q)) \; and \; (l = 1) \\ K & if \quad (i = d(q)) \; and \; (l = L+1) \\ 0 & else \end{cases}, \quad i \in V^q, l \in \{2, \ldots, L+1\}, \; q \in Q, \qquad (1)$$

$$\sum_{l \in \{1,\ldots,L\}} \left(x_{ij}^{l,q} + x_{ji}^{l,q} \right) \leq z_{ij}, \qquad ij \in E, \; q \in Q, \qquad (2)$$

$$z_{ij} \in \{0, 1\}, \qquad ij \in E, \qquad (3)$$

$$x_{ij}^{l,q} \geq 0 \text{ and integer}, \qquad (i, j, l) \in A^q, \; q \in Q. \qquad (4)$$

Each binary variable z_{ij} indicates whether edge $ij \in E$ is in the solution and each variable $x_{ij}^{l,q}$ describes the amount of flow through arc (i, j, l) for commodity q in layered graph G^q (constraints (2) together with (3) imply that $x_{ij}^{l,q} \leq 1$ for $i \neq j$, and thus, these variables may be interpreted as indicating whether arc (i, j) is used in position l in one of the paths of commodity q). Constraints (1) are the flow conservation constraints at every node of the layered graph and guarantee the existence of K paths from $o(q)$ to $d(q)$. Constraints (2) guarantee edge-disjointness of the paths.

As discussed in [1], for $K = 1$ and any L, for any K and $L = 2, 3$ and for $L = 4$ and $K = 2$, the integrality on x can be dropped (x will be integral as soon as z is). However, in the general case, x can be fractional even if z is integer.

2.2. The static problem

Let us first consider the version of the problem where the reliable status of an edge is given a priori. We denote by E_R the set of reliable edges and by $E_{\overline{R}}$ the set of remaining edges. The formulation (HopR1) for this variant uses the same variables and is very similar to the extended formulation (Hop) for the original problem, and thus we only specify the main differences.

To make meaningful comparisons between the formulations, we assume the cost of a reliable edge is $p_{ij} > 1$ times the cost of the unreliable version of that edge used in (Hop), i.e. a reliable edge $ij \in E_R$ has cost $p_{ij}c_{ij}$.

$$\text{(HopR1)} \qquad \min \quad \sum_{ij \in E_R} p_{ij}c_{ij}z_{ij} + \sum_{ij \in E_{\overline{R}}} c_{ij}z_{ij}$$

$$\text{s.t.} \quad (1), (3), (4),$$

$$\sum_{l \in \{1,\dots,L\}} (x_{ij}^{l,q} + x_{ji}^{l,q}) \leq Kz_{ij}, \qquad ij \in E_R,\ q \in Q, \tag{5}$$

$$\sum_{l \in \{1,\dots,L\}} (x_{ij}^{l,q} + x_{ji}^{l,q}) \leq z_{ij}, \qquad ij \in E_{\overline{R}},\ q \in Q, \tag{6}$$

Note that in this model the $x_{ij}^{l,q}$ variables for reliable edges are not necessarily binary (while (2) implicitly force an upper bound of 1 in model Hop). This might explain why this model produces LP bounds that are significantly worse than the one provided by the previous model (for the problem with unreliable edges).

2.3. The upgrading problem

In this version of the problem, all edges $ij \in E$ are considered as normal edges. The subset of reliable edges ($E_R \subset E$) contains the edges that can be upgraded to be reliable, and $E_{\overline{R}} := E \setminus E_R$ are edges that cannot be upgraded.

To formulate this variant, we introduce new variables y_{ij} for edges in E_R to indicate whether they are upgraded to and used as reliable edges or not.

As before, the base cost c_{ij} is multiplied by p_{ij} when the edge is upgraded to a reliable status.

$$\text{(HopR2)} \qquad \min \quad \sum_{ij \in E_R} p_{ij}c_{ij}y_{ij} + \sum_{ij \in E} c_{ij}z_{ij}$$

$$\text{s.t.} \quad (1), (4)$$

$$\sum_{l \in \{1,\dots,L\}} (x_{ij}^{l,q} + x_{ji}^{l,q}) \leq Ky_{ij} + z_{ij}, \qquad ij \in E_R,\ q \in Q, \tag{7}$$

$$\sum_{l \in \{1,\dots,L\}} (x_{ij}^{l,q} + x_{ji}^{l,q}) \leq z_{ij}, \qquad ij \in E_{\overline{R}},\ q \in Q, \tag{8}$$

$$y_{ij} + z_{ij} \leq 1, \qquad ij \in E_R, \tag{9}$$

$$y_{ij} \in \{0, 1\}, \qquad ij \in E_R. \tag{10}$$

The main differences with the previous models are constraints (9) which guarantee that if an edge in E_R is used, the two scenarios are mutually exclusive. The edge-disjointness constraints (7) are also modified.

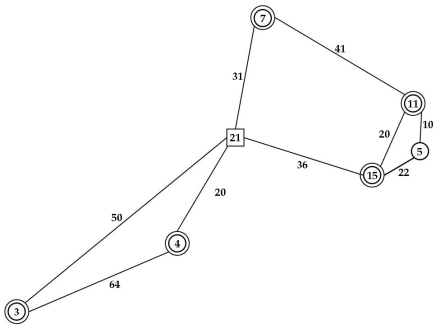
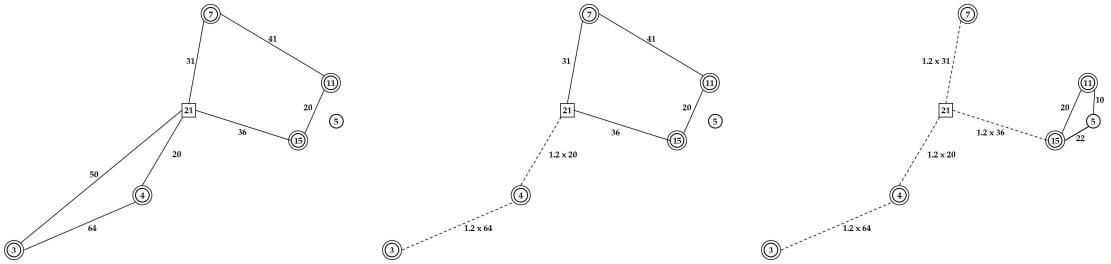


Figure 2: Network with 1 source, 5 destinations.



(a) Total Cost=262 (b) Total Cost=212 (c) Total Cost=216.4

Figure 3: Optimal topologies respectively for models Hop (a), HopR2 (b) and HopR1 (c).

For the same reasons as for the (Hop) model of Section 2.1, variables x must be defined as integer in the static and the upgrading models.

Constraints (7) and (9) are typical of the so-called multiple-choice model that often arises in network design problems with modular costs (see. e.g, [3, 4]). Note also that (HopR1) is a particular case of (HopR2) and by fixing variables in (HopR2), constraints (9) become redundant and constraints (7) become equivalent to constraints (5) of (HopR1). However, we presented both models as (HopR1) has a structure very similar to (Hop) — the only difference lies in different right hand sides of (2) and (5), while for (HopR2) the model is more complex due to the addition of the choice variables and the need for additional constraints (9).

2.4. Example

Fig. 2 shows a graph G composed of 7 nodes and 9 edges. The base cost c_{ij} is indicated next to each edge. All commodities have the same origin node, the square node 21, and the five destination nodes (the double-circle nodes) are nodes 3, 4, 7, 11 and 15. We consider parameters $L = 3$ and $K = 2$. Fig. 3(a) represents the optimal solution for the original problem where no reliable edges exist. The associated cost is equal to 262.

Assume now that a subset of the edges are reliable: $E_R = \{3-4, 3-21, 4-21, 7-21, 15-21\}$. For this example, we set multiplicative factor for the reliable edges to $p_{ij} = 1.2$ for all $ij \in E_R$. Fig. 3(c) shows the optimal topology for the static problem, while Fig. 3(b) presents the optimal solution of the upgrading problem. Here the 5 edges that were previously considered as reliable can either be used as normal or as reliable edges. We see that edges $4-21$ and $3-4$ are upgraded and edges $7-21$ and $15-21$ are not.

This solution shows a positive effect of allowing the provider to choose which edges should be used as reliable

or not since the optimal cost for the upgrading problem (212) is lower than the optimal solution of the original problem (262) and also lower than the optimal solution of the static problem (216.4). A deeper analysis of the cost effect linked to the utilization or not of reliable edges is presented later.

3. Benders decomposition

The two models are extended formulations of large scale (with $O(L.|Q|.|E|)$ integer variables and $O(L.|Q|.|V|)$ constraints) and difficult to solve even with current solvers. To overcome this difficulty, we propose to use alternative formulations by projecting out the flow variables $x_{ij}^{l,q}$ in a Benders decomposition approach. The remaining problem called master problem is only dependent on design variables y and z. When we project out the flow variables, the subproblems become independent linear programs for each commodity, thus reducing significantly the size of the linear programs to solve.

To apply this standard Benders decomposition approach, we introduce new formulations (HopR1') and (HopR2'), where we relax the integrality restrictions on x variables in (HopR1) and (HopR2), replacing (4) by

$$x_{ij}^{lq} \geq 0, \qquad (i,j,l) \in A^q, q \in Q. \tag{11}$$

As discussed above and in [1], for $K = 1$ with any L, for $L = 2, 3$ with any $K \geq 2$ and for $L = 4$ and $K = 2$, the integrality on x can be dropped (x will be integral as soon as y and z are). For higher values of K, the existence of a feasible solution with x integer for (HopR1) can be checked by solving the following integer program for each commodity:

$$\begin{aligned}
\min \quad & \sum_{ij \in E} e_{ij}^q & & \\
\text{s.t.} \quad & \sum_{j:(j,i,l-1) \in A^q} x_{ji}^{l-1,q} - \sum_{j:(i,j,l) \in A^q} x_{ij}^{l,q} & = & \begin{cases} -K & if \quad (i = o(q)) \; and \; (l = 1) \\ K & if \quad (i = d(q)) \; and \; (l = L+1) \\ 0 & else \end{cases}, \\
& & & i \in V^q, l \in \{2, \ldots, L+1\}, \; q \in Q, \\
& \sum_{l \in \{1,\ldots,L\}} \left(x_{ij}^{lq} + x_{ji}^{lq} \right) \leq K \overline{z}_{ij}, & & ij \in E_R, \\
& \sum_{l \in \{1,\ldots,L\}} \left(x_{ij}^{lq} + x_{ji}^{lq} \right) \leq \overline{z}_{ij} + e_{ij}^q (1 - \overline{z}_{ij}), & & ij \in E_{\overline{R}}, \\
& x_{ij}^{lq} \quad \text{integer}, & & (i,j,l) \in A^q, \\
& e_{ij}^q \in \{0,1\} & & ij \in E,
\end{aligned}$$

where $\overline{z}$ is the current solution to the master problem. Let $\overline{e}^q$ denote the optimal value of e^q, and $\overline{e}_{\overline{z}} = \max_{q \in Q} \{\sum_{ij \in E} \overline{e}_{ij}^q\}$. If $\overline{e}_{\overline{z}} \neq 0$, we can add the combinatorial Benders cuts

$$\sum_{ij \in E^0(\overline{z})} z_{ij} \geq \overline{e}_{\overline{z}}, \tag{12}$$

with $E^0(\overline{z}) = \{ij \in E \text{ s.t. } \overline{z}_{ij} = 0\}$, to move away from the current solution. Note that these cuts can be seen as a reinforced feasibility cut typically used in logic-based Benders decomposition, see Hooker [8] and Codato and Fischetti [2] among others. A similar approach is used for (HopR2).

The master problems associated to (HopR1') and (HopR2') are slightly different because design variables y and z have different meanings. On the other hand, the structure of the subproblems is similar. The goal of subproblems is to ensure that, for a given design vector $\overline{z}$ and a given commodity, the problem of finding K edge-disjoint paths with maximum L hops is feasible. Except for small details this problem is similar for both models.

3.1. Dual subproblems

Given a commodity $q \in Q$, let us introduce a dual variable π_i^l, associated with node $i \in V$ and layer l, for each constraint (1) and a dual variable σ_{ij} for each constraint (5) or (7). Defining $o := o(q)$ and $d := d(q)$, and adding the normalizing constraints $\pi_d^{L+1} \leq 1$ and $\pi_o^1 = 0$ to normalize the dual cone we get the dual subproblem $SP(q, \overline{y}, \overline{z})$

$$
\begin{aligned}
SP(q,\overline{y},\overline{z}) \qquad \max \quad & objective\ function && && (13)\\
\text{s.t.} \quad & \pi_i^2 - \pi_o^1 - \sigma_{oi} \leq 0, && oi \in E,\\
& \pi_i^{l+1} - \pi_j^l - \sigma_{ij} \leq 0, && ij \in E, i, j \notin \{o,d\}, l \in \{2,\ldots,(L-1)\},\\
& \pi_j^{l+1} - \pi_i^l - \sigma_{ij} \leq 0, && ij \in E, i, j \notin \{o,d\}, l \in \{2,\ldots,(L-1)\},\\
& \pi_d^{l+1} - \pi_i^l - \sigma_{id} \leq 0, && id \in E, l \in \{2,\ldots,L\},\\
& \pi_d^{l+1} - \pi_d^l \leq 0, && l \in \{2,\ldots,L\},\\
& \pi_o^1 = 0,\\
& \pi_d^{L+1} \leq 1,\\
& \sigma_{ij} \geq 0, && ij \in E.
\end{aligned}
$$

The subproblems associated to the two variants of the problem differ in their objective function. For (HopR1'), (13) becomes

$$\max \qquad K\pi_d^{L+1} - \sum_{ij \in E_R} K\overline{z}_{ij}\sigma_{ij} - \sum_{ij \in E_{\overline{R}}} \overline{z}_{ij}\sigma_{ij} \tag{14}$$

while for (HopR2'), we replace (13) by

$$\max \qquad K\pi_d^{L+1} - \sum_{ij \in E_R} K\overline{y}_{ij}\sigma_{ij} - \sum_{ij \in E} \overline{z}_{ij}\sigma_{ij} \tag{15}$$

For both problems and for a particular commodity $q \in Q$, given a vector $(\overline{y}, \overline{z})$ defining a graph, if it is possible to find a feasible flow in terms of continuous x variables leading to K edge-disjoint paths composed of maximum L hops, then the value of the objective function of $SP(q, \overline{y}, \overline{z})$ is equal to 0. Otherwise, a new Benders cut is added to the master problem. Depending on the nature of the problem the Benders cuts are slightly different. For (HopR1'), Benders cuts are:

$$K\overline{\pi}_{d(q)}^{L+1} - \sum_{ij \in E_R} \overline{\sigma}_{ij} K z_{ij} - \sum_{ij \in E_{\overline{R}}} \overline{\sigma}_{ij} z_{ij} \leq 0 \tag{16}$$

and for (HopR2'), Benders cuts are:

$$K\overline{\pi}_{d(q)}^{L+1} - \sum_{ij \in E_R} \overline{\sigma}_{ij} K y_{ij} - \sum_{ij \in E} \overline{\sigma}_{ij} z_{ij} \leq 0 \tag{17}$$

3.2. Master problems

Given a set $\mathcal{R}^q$ of extreme points of the dual subproblem $SP(q, \overline{y}, \overline{z})$ described in the previous subsection, the master problems associated with the two versions of the problem are given below.

$$\begin{aligned}
\text{(BR-HopR1')} \qquad \min \quad & \sum_{ij \in E_R} p_{ij} c_{ij} z_{ij} + \sum_{ij \in E_{\overline{R}}} c_{ij} z_{ij} & & \\
\text{s.t.} \quad & K\overline{\pi}_{d(q)}^{L+1} - \sum_{ij \in E_R} \overline{\sigma}_{ij} K z_{ij} - \sum_{ij \in E_{\overline{R}}} \overline{\sigma}_{ij} z_{ij} \leq 0, & (\overline{\pi}, \overline{\sigma}) \in \bigcup_{q \in Q} \mathcal{R}^q, & \\
& z_{ij} \in \{0, 1\}, & ij \in E. &
\end{aligned}$$

$$\begin{aligned}
\text{(BR-HopR2')} \qquad \min \quad & \sum_{ij \in E_R} p_{ij} c_{ij} y_{ij} + \sum_{ij \in E} c_{ij} z_{ij} & & \\
\text{s.t.} \quad & K\overline{\pi}_{d(q)}^{L+1} - \sum_{ij \in E_R} \overline{\sigma}_{ij} K y_{ij} - \sum_{ij \in E} \overline{\sigma}_{ij} z_{ij} \leq 0, & (\overline{\pi}, \overline{\sigma}) \in \bigcup_{q \in Q} \mathcal{R}^q, & \\
& y_{ij} + z_{ij} \leq 1, & ij \in E_R, & \\
& y_{ij} \in \{0, 1\}, & ij \in E_R, & \\
& z_{ij} \in \{0, 1\}, & ij \in E. &
\end{aligned}$$

4. Branch-and-cut algorithm

The Benders reformulation is useful here because the three extended formulations (Hop, HopR1 and HopR2) are composed with two groups of variables with a large number of x variables. A first natural approach consists in solving this kind of huge extended models with a commercial solver like CPLEX. But the large number of variables and constraints can have a negative effect on the performance (CPU time and memory usage) of this kind of techniques.

A second approach consists in using the Benders decomposition approach with cutting plane techniques. The main idea is to solve, in an alternative manner, at each iteration, the master problem and the subproblems for each commodity $q \in Q$. At each iteration, a new branch-and-bound tree is explored to solve the master problem. Then the solution of the master problem in terms of design variables z gives new parameters $\overline{z}$ for the different subproblems. For each commodity $q \in Q$ a new subproblem is generated and solved to optimality using the new parameters from the master problem. Depending on the optimal solution of each subproblem, a new cut is possibly generated and added to the master problem for the next iteration. This process stops when the optimal solution of the master problem's branch-and-bound tree is a vector of y, z variables for which no more Benders cut is added. At this stage, it is ensured that there exists a vector x feasible for the linear relaxation of our models. A weak cut (12) can then be generated if no integral feasible x can be found.

This cutting plane approach can converge slowly because a new branch-and-bound tree has to be explored at each iteration (this was also observed in [5] for a multi-layer network design problem). Therefore, we proposed in [1] to embed Benders cuts in a branch-and-cut algorithm. We have already deeply investigated this algorithmic aspect in [1] and we propose in this paper to use the fastest branch-and-cut algorithm we have developed to test models (HopR1) and (HopR2).

Algorithm 1: Hybrid branch-and-cut algorithm: `bc-n`.

begin `/* Initialization */`
  $T = \{o\}$ where o has no branching constraints;
  $UB = +\infty$;
while T *is nonempty* **do**
  select a node $o' \in T$;
  $T \leftarrow T \backslash \{o'\}$; `/* withdraw node` o' `from the tree */`
  solve o';
  let $\overline{z}$ be an optimal solution;
  let $\overline{w}$ be the optimal cost;
  if $\overline{w} < UB$ **then**
    if $\overline{z} \in \{0,1\}^{|E|}$ ***or*** $depth(o') \leq n$ **then**
      foreach $q \in Q$ **do** compute $s_q = SP(q, \overline{y}, \overline{z})$;
      if $s_q > 0$ **then** add (16) or (17) to (MP);
    if $\overline{z} \in \{0,1\}^{|E|}$ ***and*** $s_q \leq 0$ *for each* $q \in Q$ **then**
      foreach $q \in Q$ **do** compute $f_q = FP(q, \overline{y}, \overline{z})$ (see [1]);
      if $f_q > 0$ *for some* $q \in Q$ **then** add combinatorial Benders cuts (12) to (MP);
      else
        $UB \leftarrow \overline{w}$; `/* define a new upper bound */`
        $z^* \leftarrow \overline{z}$; `/* save current incumbent */`
    if $s_q > 0$ ***or*** $f_q > 0$ *for some* $q \in Q$ **then**
      $T \leftarrow T \cup \{o'\}$; `/* put node` o' `back in the tree */`
    else if $\overline{z} \notin \{0,1\}^{|E|}$ **then**
      branch, resulting in nodes o^* and o^{**};
      $T \leftarrow T \cup \{o^*, o^{**}\}$; `/* add children to the tree */`
return z^*

Our best performing branch-and-cut algorithm (called bc-5-heur) adds Benders cuts only at "strategic points" of the exploration tree: when the optimal solution of the relaxation at a node is integer, and when the depth of the node is less or equal to a given parameter n. This algorithm adds a lot of Benders cuts at the beginning of the exploration. In [1], we performed a tuning stage to evaluate the best value for the parameter on a set of complicated instances and $n = 5$ gave the best results. We use the same value for this parameter in this paper.

Furthermore, the algorithm is improved with a good upper bound found by a simple fix-and-branch heuristic, also proposed in [1]. The linear relaxation at the root node of the branch-and-bound tree usually has a large number of y and z variables equal to 0. The heuristic consists in fixing those variables to 0 before starting the branch-and-bound process. The small size of this problem allows to obtain its optimal solution quickly. In this paper, we experience our branch-and-cut algorithm with the heuristic for both models (bc-5-heurR1 and bc-5-heurR2).

Table 1: Instances description.

Name	$\|N\|$	$\|E\|$	$\|Q\|$	Rooted demands?
TC-5	21	210	5	true
TC-10	21	210	10	true
pdh	11	34	24	false
di-yuan	11	42	22	false

5. Computational experiments

The objective of our experiments is twofold. First we want to study the behavior of the formulations for solving the proposed variants when the proportion of reliable edges changes. More precisely, we want to examine the gap between the linear programming relaxation value and the optimal solution value as well as the CPU time needed to reach optimality. We also compare our branch-and-cut algorithms (bc-5-heurR1 and bc-5-heurR2) with the alternative of using the formulations within CPLEX. The second objective of this section is to examine more closely the optimal solutions obtained for the upgrading variant in contrast to the optimal solutions obtained for the original problem and for the static variant in order to evaluate the economical impact of allowing to upgrade edges for a network manager.

5.1. Implementation details

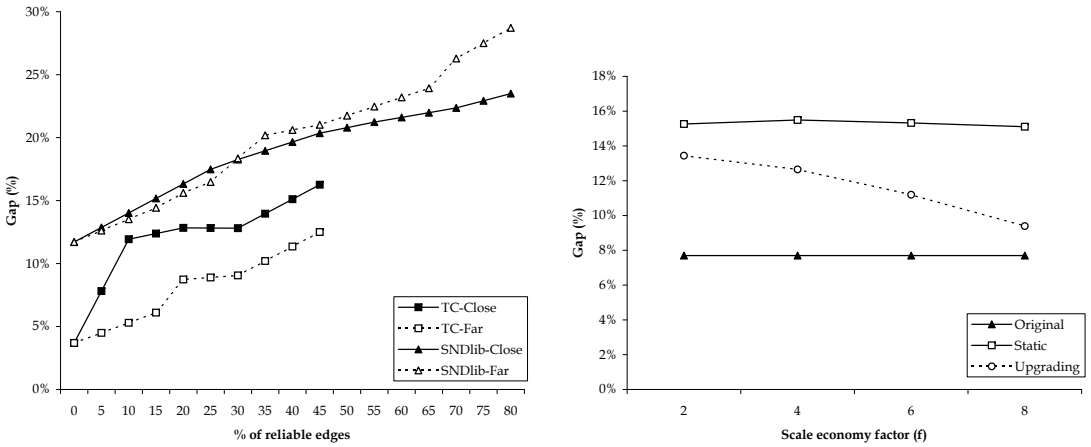
All models have been coded in JAVA using CPLEX 12 MIP solver and run on a DELL Latitude D820 with a processor Intel Core Duo 2 T7200 of 2GHz and 2.5 GB of RAM memory. We allow CPLEX to store the branch-and-bound tree in a file, setting parameter *IntParam.NodeFileInd* to 2, to avoid from running out of memory.

5.2. Instance details

For our computational experiments we used two different test sets. One set denoted by TC, is taken from a class of complete graphs that has been widely used in the network design literature, e.g, [1, 6]. For each instance, the cost matrix is given by the integer parts of the Euclidean distance between the coordinates of the 21 nodes, randomly placed among the integer points of a grid (100×100) and all point-to-point demands have one of their extremities in common (which we call rooted demands in Table 1). The second set contains two instances that are based on sparse networks from SNDlib [12]: pdh and di-yuan. Table 1 reports the characteristics of all the instances. We solved the instances with L ranging from 3 to 5 and K equal to 2 and 3. We set a time limit of 3600 seconds for all instances and algorithms.

We have also evaluated the models in terms of three other parameters: the scale economy parameter, the proportion of reliable edges and the geographical distribution of reliable edges. The multiplicative factor for the cost of a reliable edge ij is given by $p_{ij} = ((K-1)+f)$, where $0 < f < 1$. This allows to model different economy of scale scenarios for the use of reliable edges in both versions. In this computational study we have considered $f \in \{0.2, 0.4, 0.6, 0.8\}$ to evaluate how sensitive the optimal design is to changes in the scale economy parameter.

For the geographical distribution of reliable edges, we have evaluated two different strategies, that we call Close and Far. To define Close we consider two parameters cg and c. The parameter cg defines one of the nodes



(a) (b)

Figure 4: Evolution of the gap (%) in function of the percentage of reliable edges (a) and depending on the scale economy factor f (b) for all instances tested.

of the network which is considered as a center of gravity. The parameter c indicates that we create a subset of $(c + 1)$ nodes C composed of cg and of c additional nodes which are the closest nodes from the center of gravity node cg. Then, the set of reliable edges is given by

$$E_R = \{ij \in E : i \in C \text{ or } j \in C\}.$$

We have performed tests with $c \in \{0, 1, 2, 3, 4\}$ for TC instances and $c \in \{0, 1, 2, 3\}$ for SNDlib instances. The Far strategy is also defined with two parameters cg and b. As before, the parameter cg defines a center of gravity and the parameter b indicates that we create a subset F of b nodes which are the farthest nodes from the center of gravity node cg. Then,

$$E_R = \{ij \in E : i \in F \text{ and } j \in F\}.$$

Note that the Close strategy sets as reliable, those edges which are close to a center of gravity and the Far strategy favors edges far from the center of gravity. The set F is a clique and contains all edges ij belonging to the farthest clique (composed of b nodes) from the center of gravity cg. We have performed tests with $b \in \{0, 8, 10, 12, 14\}$ for TC instances and $b \in \{0, 4, 5, 6, 7\}$ for SNDlib instances to keep a proportion of reliable edges similar for the two kind of instances.

5.3. Evaluation of the models

The two variants studied in this paper can be solved with the hop-indexed model developed in [1] with simple modifications. The first evaluation criterion is given by the average gap (expressed in %) which represents the relative difference between the optimal linear programming relaxation solution and the optimal integer solution, i.e. $\frac{(IP^* - LR^*)}{IP^*}$. The results depicted on Fig. 4(a) clearly show that the gap increases with the proportion of reliable edges.

These results are interesting because:

i) without any reliable edges, the extended hop-indexed model performs reasonably well, e.g., for $K = 2, 3$ the average gap decreases to reach approximately the threshold of 5% for the TC test set as shown in [1].

Table 2: Arithmetic average of CPU time (sec.).

	Original	**Static**	**Upgrading**
Extended formulation	41.10	122.80	111.44
Branch-and-cut algorithm + heuristic	2.96	6.26	14.42

Table 3: Savings obtained by the utilization of reliable edges.

	Upgrading < Original	Upgrading < Static
(%) of all the instances	42.76	58.21
Arithmetic average of savings realized (%)	3.95	7.05
Maximum savings realized (%)	24.29	24.70

ii) in the limit, the problem with all edges being reliable should become easier since, for each commodity, all paths should share the same set of edges.

Table 2 clearly shows that the original problem without reliable edges is the easiest problem since it takes less time to reach optimality for the associated extended formulation and for the branch-and-cut algorithm. The two new variants (static and upgrading) are harder to solve as the CPU time increases considerably. The average time needed for the two extended formulations (HopR1) and (HopR2) to reach optimality are close (around 110 seconds). It is not the case for the branch-and-cut algorithms since on average the static version takes less time than the upgrading version (6.26 seconds versus 14.42 seconds respectively). For both variants, the associated branch-and-cut algorithms again outperform the standard extended formulations as in [1] for the original model and problem. The improvement is really significant and in some cases, the branch-and-cut algorithms is 40 times faster than using the formulation within CPLEX. Note also that only 1 instance out of 912 is not solved to optimality within the 3600 seconds time limit for the static version with a remaining gap to close of 7%. However, this instance is easily solved to optimality by the proposed branch-and-cut bc-5-heurR1 (in 70.67 seconds). Table 10 shows the good performance of the heuristic. For the static variant, the value given by the heuristic is the optimal solution in around 45% of the case, otherwise the gap ($\frac{(HEUR^* - IP^*)}{IP^*}$) between the value of the heuristic and the optimal solution is around 3.34%. Note that the heuristic takes on average 1.64 seconds to find the upper bound which is clearly better than the 122.80 seconds and the 6.26 seconds respectively needed by the extended formulation and the branch-and-cut algorithm. For the upgrading variant the conclusions about the heuristic are equivalent.

5.4. Economic advantages of upgrading edges

One of the advantages of having a pool of reliable edges is to simplify the task of the managers since the edge-disjoint property is not required for edges in the pool. It is clear that if the set of potential reliable edges becomes smaller then finding the optimal solution becomes less difficult. Our aim is now to measure the savings the utilization of reliable edges can bring.

Table 3 shows that in approximately 43% of the instances, the upgrading variant leads to a cheaper optimal design compared to the one obtained by the original problem with an average saving of 3.95% of the total design cost when reliable edges are not included. Moreover in approximately 60% of the instances, the upgrading variant leads to a cheaper optimal design compared to the one of the static variant with an average saving of 7.05% of the

Table 4: Arithmetic average of gap (%) depending on the number of commodities.

Instance		**Problems**		
Name	$\|Q\|$	Original	Static	Upgrading
TC	5	1.61	6.61	5.82
	10	5.77	14.85	10.84
pdh	27	11.84	20.77	15.78
di-yuan	48	13.62	19.97	14.98
Average		7.70	15.30	11.67

Table 5: Arithmetic average of gap (%) depending on the location of the reliable edges set.

Geographical distribution	**Problems**	
	Static	Upgrading
Close	16.20	12.41
Far	14.49	11.00

Table 6: Arithmetic average of gap (%) depending on the number of edge-disjoint paths K.

K	**Problems**		
	Original	Static	Upgrading
2	10.21	16.39	14.15
3	5.19	14.20	9.19

Table 7: Arithmetic average of gap (%) depending on the hop limit L.

L	**Problems**		
	Original	Static	Upgrading
3	9.86	16.65	13.25
4	7.89	15.61	11.87
5	5.35	13.63	9.89

Table 8: Impact on the arithmetic average of gap of the scale economy factor f.

			Scale economy factor f			
	Network	**K**	**0.2**	**0.4**	**0.6**	**0.8**
Static	TC	2	10.61	10.98	10.36	9.81
	TC	3	11.88	11.32	10.70	10.18
	SNDlib	2	22.00	22.98	23.44	23.63
	SNDlib	3	17.46	17.66	17.84	17.95
Average Static			15.26	15.49	15.32	15.11
Upgrading	TC	2	10.38	10.21	8.57	6.19
	TC	3	9.96	8.64	7.26	5.45
	SNDlib	2	21.31	21.13	19.72	18.05
	SNDlib	3	12.83	11.35	9.97	8.67
Average Upgrading			13.44	12.65	11.20	9.39

Table 9: Impact on the arithmetic average of IP CPU Time of the scale economy factor f.

			Scale economy factor f			
	Network	**K**	**0.2**	**0.4**	**0.6**	**0.8**
Static	TC	2	8.17	6.92	6.83	5.89
	TC	3	10.43	8.83	8.17	7.61
	SNDlib	2	6.74	7.49	7.45	7.47
	SNDlib	3	1.71	1.72	1.70	1.69
Average Static			6.89	6.33	6.11	5.72
Upgrading	TC	2	20.57	19.79	11.29	7.12
	TC	3	12.78	8.16	6.18	5.20
	SNDlib	2	34.11	42.19	29.08	24.04
	SNDlib	3	3.46	3.48	3.13	2.88
Average Upgrading			17.67	18.17	12.23	9.62

Table 10: Arithmetic average of heuristic CPU time (sec.) and quality of the heuristic.

	HEUR*=IP*	**Heuristic CPU Time (sec.)**	**Heuristic gap (%)**
Static	415/912 (45.50%)	1.64	3.34
Upgrading	561/912 (61.51%)	3.87	1.19

total design cost. This means that the improvements proposed by the upgrading variant in terms of cost are not huge, indeed only small percentages of the total design cost can be saved. Nevertheless, although the percentage is small, it could represent huge amounts of savings for managers. The question for managers now is to know if the increase in the time needed to reach optimality can be balanced by the savings in design cost. With a simple extended model as the one we have given, the answer appears to be negative, but with a fast branch-and-cut algorithm as the one developed in this paper it could be worth to integrate the use of reliable edges in these problems.

Tables 4, 5, 6 and 7 report the same results (average gap at the root node) related with the number of commodities, the geographical distribution of the reliable edges set, the number of required edge-disjoint paths and the hop limit, respectively. The results again illustrate the increase of the gap between the original problem and the extensions with reliable edges.

Results reported in Table 5 show that the Close instances lead to worse gaps than the Far instances. These results are easily explained since the reliable edge clique defined by the Far strategy is far from the center and thus it is unlikely that many reliable edges will be used in the optimal design. Thus, the instances become "closer" to the problem without reliable edges which, as pointed out before, appears easier to solve. On the other hand, if the reliable edges are close to the center, then the probability to use a reliable edge in the optimal design increases and, based on previous remarks, the problem becomes more difficult to solve.

Table 6 shows that the LP gaps become smaller for instances with more edge disjoint paths (K = 3 versus 2) and Table 7 shows that the gaps become smaller when more edges are allowed in the paths.

Another evaluation criteria is related to the scale economy factor f. In Table 8 we show the average gap (%) obtained for the static and upgrading models and on Table 9 the average IP CPU time (in seconds) for $K = 2, 3$ and different values of the scale economy factor f. We show on Fig. 4(b) that for the upgrading problem, as the factor f increases, the average gap decreases as well as the average IP CPU time. This observation is not a surprise because the economical incentive to use reliable edges decreases when f increases. Thus if it is less profitable to use reliable edges, it is easier to find the optimal topology because the problem becomes closer to the original problem without reliable edges.

6. Conclusion

In this paper we have proposed and studied two variants to integrate the concept of reliable edges in the survivable network design problem with hop constraints. We have shown how to adapt models previously developed for the basic problem. We have also adapted the branch-and-cut algorithm developed for the original problem and have tested its performance on a test set. One conclusion of this testing phase is that the linear programming gaps increase when the proportion of reliable edges increases. We can also conclude that for the upgrading problem, when the scale economy factor associated to the reliable edges increases, the linear programming gap decreases because the upgrading problem becomes closer to the original problem without reliable edges. Finally, the use of reliable edges leads to reasonable savings and this could motivate managers to consider the two types of edges when designing a network.

We conclude by pointing out interesting research topics that are, in a certain way, motivated by the present work and could be worth investigating:

i) This work is a direct follow up to the previous study [1] that explains the incorporation of the hop constraints. However, it is clear that incorporating reliable edges makes sense in problems dealing only with survivability considerations. Furthermore, it would be interesting to compare the behavior of the different models with and without hop constraints.

ii) A further generalization of the second variant studied here is to measure the number of paths of a given commodity through a reliable edge. Consider, for instance, a case with $K = 3$ (this generalization only makes sense for $K > 2$) and we want to distinguish the situations: one path traverses edge ij, two paths traverse edge ij and three paths traverse edge ij. Economies of scale should be considered but we would like to set the cost of using two paths less than the cost of using three paths. This could be easily modeled, by creating new variables with adequate coefficients, for each possible scenario. Finally, knapsack-like constraints imposing a maximum slack for each commodity, or for all commodities could be used as well.

iii) Finally, model (HopR1) appears to be more difficult to solve than the original model (Hop). We have pointed out that the reason for this behavior might be explained by the fact that many path variables are no longer binary. It would be worth to see how the given model can be improved for this simple but apparently elusive variation.

Acknowledgements

The work of Quentin Botton is supported by "Concours Bourses de Voyage" of the "Communauté française de Belgique" and Marie-Curie Research Training Network "ADONET". The work of Bernard Fortz is supported by the Interuniversity Attraction Poles Programme initiated by the Belgian Science Policy Office. The work of Luis Gouveia is supported by National Funding from FCT - Funda�still;o para a Ciłncia e a Tecnologia, under the project: PEst-OE/MAT/UI0152.

References

[1] Q. Botton, B. Fortz, L. Gouveia, and M. Poss. Benders decomposition for the hop-constrained survivable network design problem. *INFORMS Journal on Computing*, Articles in Advance (http://dx.doi.org/10.1287/ijoc.1110.0472):1–14, 2011.

[2] G. Codato and M. Fischetti. Combinatorial Benders' cuts for mixed-integer linear programming. *Oper. Res.*, 54(4):756–766, 2006.

[3] K.L. Croxton, B. Gendron, and T.L. Magnanti. A comparison of mixed-integer programming models for nonconvex piecewise linear cost minimization problems. *Management Science*, 49(9):1268–1273, 2003.

[4] K.L. Croxton, B. Gendron, and T.L. Magnanti. Variable disaggregation in network flow problems with piecewise linear costs. *Operations Research*, 55(1):146–157, 2007.

[5] B. Fortz and M. Poss. An improved Benders decomposition applied to a multi-layer network design problem. *Oper. Res. Lett.*, 37(5):359–364, 2009.

[6] L. Gouveia. Using variable redefinition for computing lower bounds for minimum spanning and steiner trees with hop constraints. *INFORMS J. Comput.*, 10:180–188, 1998.

[7] L. Gouveia. Using variable redefinition for computing lower bounds for minimum spanning and steiner trees with hop constraints. *INFORMS Journal on Computing*, 10(2):180–188, 1998.

[8] J.N. Hooker. *Logic-based methods for optimization: combining optimization and constraint satisfaction*. Wiley-Interscience, 2000.

[9] D. Huygens, M. Labbé, A.R. Mahjoub, and P. Pesneau. The two-edge connected hop-constrained network design problem: Valid inequalities and branch-and-cut. *Networks*, 49(1):116–133, 2007.

[10] A. Itaí, Y. Perl, and Y. Shiloach. The complexity of finding maximum disjoint paths with length constraints. *Networks*, 2:277–286, 1982.

[11] H. Kerivin and A.R. Mahjoub. Design of survivable networks: A survey. *Networks*, 46(1):1–21, 2005.

[12] S. Orlowski, R. Wessäly, M. Pióro, and A. Tomaszewski. SNDlib 1.0 Survivable Network Design Library. *Networks*, 55(3):276–286, 2010. ISSN 1097-0037.

[13] M. Pióro and D. Medhi. *Routing, Flow, and Capacity Design in Communication and Computer Networks*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2004. ISBN 0125571895.

[14] M. Zotkiewicz, W. Ben-Ameur, and M. Pióro. Failure disjoint paths. *Electronic Notes in Discrete Mathematics*, 36:1105–1112, 2010.