4OR manuscript No.
(will be inserted by the editor)

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

Quentin Botton · Bernard Fortz · Luis Gouveia

Received: date / Accepted: date

Abstract In this paper, we study the hop-constrained survivable network design problem with reliable edges. Given a graph with non-negative edge weights and node pairs Q, the hop-constrained survivable network design problem consists of constructing a minimum weight 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 this paper, we propose and study the hop-constrained survivable network design problem with so-called *reliable* edges where in addition, we consider a subset of 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 et al (2011) for the case without reliable edges. As before, due to the huge number of variables and constraints included in the extended formulations, we use Benders decomposition to accelerate the solving process. We develop an exact branch-and-cut algorithm and a fix-and-branch heuristic. 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

Mathematics Subject Classification (2000) 90C27 · 68M10 · 90C57 · 49M27 · 90C10

1 Introduction

Given a graph $G = (V,E)$ with nonnegative edge weights c_{ij} and a set Q of node pairs, we want to find a set of edges of minimum weight 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 arcs. Given two distinct nodes o (the origin vertex of demand) and d (the destination vertex of demand) of V, we say that an od-path is a sequence of node-arcs $P = (v_0,(v_0,v_1),v_1,...,(v_{l-1},v_l),v_l)$, where $l \geq 1$, $v_0,v_1,...,v_l$ are distinct vertices, $v_0 = o$, $v_l = d$, and (v_{i-1},v_i) is an arc connecting v_{i-1} and v_i (for $i = 1,...,l$). A collection $P_1,P_2,...,P_k$ of od-L-paths is called

Q. Botton
Louvain School of Management, Université catholique de Louvain, Louvain-la-Neuve, Belgium
e-mail: quentin.botton@uclouvain.be

B. Fortz
Départment d'Informatique, Faculté des Sciences, Université Libre de Bruxelles, Bruxelles, Belgium
e-mail: bernard.fortz@ulb.ac.be

L. Gouveia
DEIO, CIO, Faculdade de Ciências da Universidade de Lisboa, Lisboa, Portugal
e-mail: legouveia@fc.ul.pt

edge-disjoint if any arc (i,j) and its symmetric arc (j,i) appear in at most one path. This problem is NP-hard (Itaí et al 1982) and was first studied by Huygens et al (2007) which 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 (2011). They consider instances with $1 \leq K \leq 3$ and $3 \leq L \leq 5$.

The survey by Kerivin and Mahjoub (2005) 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 arcs for each path of each commodity. Briefly, the requirement for using disjoint paths guarantees the existence 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 arcs (or hops).

The problem previously described is uniform in the sense that all commodities have the same priority. That is, for each commodity we associate the same number, K, of disjoint paths, as well as the same limit, L, for the number of arcs in the paths. However, it is not difficult to motivate variations of the problem where some commodities are more important than others (e.g., the traffic sent from a to b is more important than the traffic from c to d) and thus we may associate a parameter $K = k_1$ to some commodities and a parameter $K = k_2$ to others with $(k_1 \neq k_2)$. In a similar way, the parameter L may differ from commodity to commodity (Bley 1997).

In this paper, we study two different versions of the problem where the lack of uniformity is motivated by different degrees of reliability associated to the edges of the underlying graph. In the first variant we assume that a subset of edges, E_R, of the network represented by the graph $G = (V,E)$, is composed of more reliable but more costly edges. 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 ij of the network. We assume that each edge ij has a base cost c_{ij} that is proportional to the length of the wire needed to connect the two nodes i and j. Since reliable edges can be used K times by the paths in any given commodity, it makes sense to assume that their cost must be higher than the cost of a similar and unreliable edge.

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 correspond to edges ij that could be upgraded to more reliable edges provided that a higher cost (as defined above) is incurred. Clearly, the first variant is a special case of the second one and, for a given instance, the optimal solution to this variant cannot cost more than the optimal solution to the previous variant neither to the optimal solution of the original problem (with no reliable edges).

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 Botton et al (2011) 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.

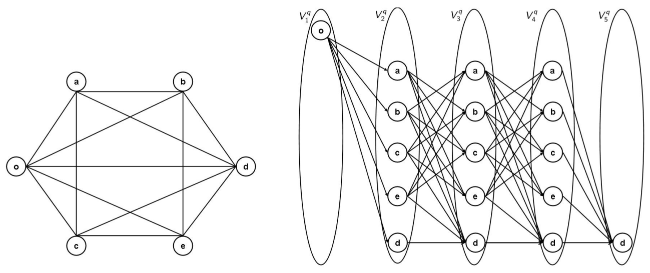
2 Problem definition and formulations

In this section, we start by describing the model for the problem without reliable edges proposed in Botton et al (2011) 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 non-directed 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 $V_l^q = V \setminus \{o(q)\}$, $l = 2,\ldots,L$. Let v_l^q be the copy of $v \in V$ in the *l-th* layer of graph G^q. Then, the arcs 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)



(a) (b)

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

edge in E 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 corresponds 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 (1998) 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:

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

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

$$i \in V^q, l \in \{2,\dots,L+1\}, q \in Q, \qquad (1)$$

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

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

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

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 $x_{ij}^{l,q} \le 1$ for $i \ne j$, 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 network flow 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. Note that (1) implies that $x_{ii}^{l,q} \le K$.

The flow variables are required to be integer in the previous model. To see why notice that, even for a single commodity, $L \ge 4$ and for a fixed vector z, the constraints (2) link together the flow variables of copies of the same original arc in different layers. Hence, the resulting formulation is more constrained than a straighforward flow model and the total unimodularity of the matrix is lost. Therefore x can be fractional even if z is integer. Note also that if constraints (1) and (2) would lead to an integer polyhedron (for a fixed binary z), it would also imply that we can decide in polynomial time if a graph contains K disjoint L-path. This would contradict the complexity result of Itaí et al (1982) unless $P = NP$. An example of fractional x for an integer z is described in Botton et al (2011) (Section 4 and Figure 3).

On the other hand, as discussed in Botton et al (2011), 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).

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. 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.

$$
\begin{aligned}
\text{(HopR1)} \qquad & \min && \sum_{ij\in E} p_{ij}c_{ij}z_{ij} && \\
& \text{s.t.} && (1),(3),(4), && \\
& && \sum_{l\in\{1,\dots,L\}} (x_{ij}^{l,q}+x_{ji}^{l,q}) \le v_{ij}z_{ij}, && ij\in E, q\in Q, \qquad (5)
\end{aligned}
$$

The difference in constraints (5) is the inclusion of the coefficient v_{ij} (which can be viewed as the "capacity" of the edge ij) and which takes the value 1 if the edge ij is a non-reliable edge ($ij \in E_{\overline{R}}$) and the value K if the edge is reliable ($ij \in E_R$). In the later case the edge may be used by all K paths of each given commodity. Note also that in this model the $x_{ij}^{l,q}$ variables for reliable edges are not necessarily binary (while (2) implicitely 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). In our numerical experiments, the new parameter p_{ij} for each edge in the cost function is equal to 1 for normal (unreliable) edges and to $((K-1)+f)$ for reliable edges, with $0<f<1$.

2.3 The upgrading problem

In this version of the problem, all edges ij are considered as normal edges ($E_{\overline{R}} = E$). The subset of reliable edges ($E_R \subset E_{\overline{R}}$) contains the edges that can be upgraded to be reliable.

To formulate this variant, we duplicate variables z_{ij} for edges in E_R by adding a new index s. The new binary variables z_{ij}^s indicate whether edge ij is used in the scenario s in the optimal topology, that is $z_{ij}^1 = 1$, if edge ij is included in the solution as a normal edge and $z_{ij}^2 = 1$ if edge ij is used in the solution as a reliable edge.

The parameter v^s is the capacity of an edge in scenario s: $v^1 = 1$ and $v^2 = K$. The base cost c_{ij} is also multiplied by a coefficient depending on the scenario. In our numerical experiments, we chose $p^1 = 1$ and $p^2 = (K-1)+f$, with $0<f<1$.

$$
\begin{aligned}
\text{(HopR2)} \qquad & \min && \sum_{ij\in E}\sum_{s=1}^{2} p^s c_{ij} z_{ij}^s && \\
& \text{s.t.} && (1),(4) && \\
& && \sum_{l\in\{1,\dots,L\}} (x_{ij}^{l,q}+x_{ji}^{l,q}) \le \sum_{s=1}^{2} v^s z_{ij}^s, && ij\in E, q\in Q, \qquad (6)\\
& && \sum_{s=1}^{2} z_{ij}^s \le 1, && ij\in E, \qquad (7)\\
& && z_{ij}^2 = 0, && ij\in E\setminus E_R, \qquad (8)\\
& && z_{ij}^s \in \{0,1\}, && ij\in E, s\in\{1,2\}. \qquad (9)
\end{aligned}
$$

The main differences with the previous models are constraints (7) which guarantee that if an edge in E_R is used, the two scenarios are mutually exclusive. The edge-disjointness constraints (6) are also modified, and constraints (8) ensure only edges in E_R can be upgraded.

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 (6) and (7) are typical of the so-called multiple-choice model that often arises in network design problems with modular costs (see. e.g, Croxton et al (2003)). Note also that as mentioned before, the HopR1 is a particular case of HopR2 and by fixing variables in HopR2, constraints (7) become redundant and constraints (6) 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 (7).

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.

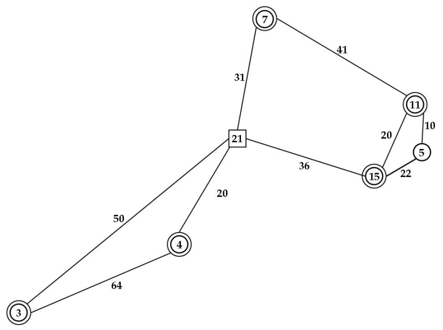
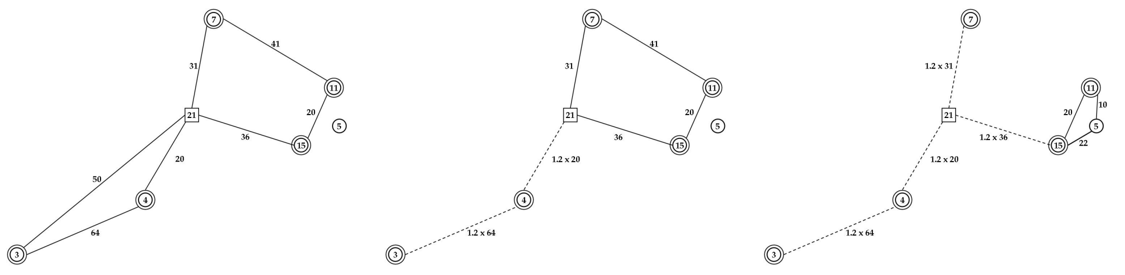


Fig. 2 Network with 1 source, 5 destinations.

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 the scale economy factor f to 0.2. 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.



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

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

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 extended models are of very large scale 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 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.

But we can only project out continuous variables, so we introduce new formulations HopR1' and HopR2', where we relax the integrality restrictions on x variables in HopR1 and HopR2, replacing (4) by

$$x_{ij}^{lq} \geq 0, \qquad (i,j,l) \in A^q, q \in Q. \tag{10}$$

As discussed above and in Botton et al (2011), 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 z is). For higher values of K, a feasibility subproblem can be solved and a weak cut can be generated if the problem turns out to be infeasible.

The master problems associated to HopR1' and HopR2' are slightly different because design variables 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 $\bar{z}$ and a given commodity, the problem of finding K edge-disjoint paths with maximum L hops is feasible. And except for small details this problem is similar for both models.

3.1 Master problem for HopR1'

$$\begin{array}{lll}
\text{(BR-HopR1')} & \min & \displaystyle\sum_{ij\in E} p_{ij} c_{ij} z_{ij} \\
 & \text{s.t.} & K\pi_{d(q)}^{L+1} - \displaystyle\sum_{ij\in E} \overline{\sigma}_{ij} v_{ij} z_{ij} \leq 0, \ (\overline{\pi}, \overline{\sigma}) \in \bigcup_{q\in Q} \mathscr{R}^q, \\
 & & z_{ij} \in \{0,1\}, \qquad ij \in E.
\end{array}$$

3.2 Master problem of HopR2'

$$\begin{array}{lll}
\text{(BR-HopR2')} & \min & \displaystyle\sum_{ij\in E} \sum_{s=1}^{2} p^s c_{ij} z_{ij}^s \\
 & \text{s.t.} & K\pi_{d(q)}^{L+1} - \displaystyle\sum_{ij\in E} \sum_{s=1}^{2} \overline{\sigma}_{ij} v^s z_{ij}^s \leq 0, \ (\overline{\pi}, \overline{\sigma}) \in \bigcup_{q\in Q} \mathscr{R}^q, \\
 & & \sum_{s=1}^{2} z_{ij}^s \leq 1, \qquad ij \in E, \\
 & & z_{ij}^s \in \{0,1\}, \qquad ij \in E, s \in \{1,2\}.
\end{array}$$

For both master problems $\mathscr{R}^q$ contains the extreme points of the feasibility polyhedron for the dual subproblem $SP(q,\bar{z})$ described next.

3.3 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 (6). 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 $SP(q,\bar{z})$

$$
\begin{array}{llll}
\mathrm{SP}(q,\overline{Z}) & \max & \mathit{objective\ function} & (11)\\
 & \text{s.t.} & \pi_i^2-\pi_o^1-\sigma_{oi}\le 0, & oi\in E,\\
 & & \pi_i^{l+1}-\pi_j^l-\sigma_{ij}\le 0, & ij\in E, i,j\notin\{o,d\}, l\in\{2,\dots,(L-1)\},\\
 & & \pi_j^{l+1}-\pi_i^l-\sigma_{ij}\le 0, & ij\in E, i,j\notin\{o,d\}, l\in\{2,\dots,(L-1)\},\\
 & & \pi_d^{l+1}-\pi_i^l-\sigma_{id}\le 0, & id\in E, l\in\{2,\dots,L\},\\
 & & \pi_d^{l+1}-\pi_d^l\le 0, & l\in\{2,\dots,L\},\\
 & & \pi_o^1=0, & \\
 & & \pi_d^{L+1}\le 1, & \\
 & & \sigma_{ij}\ge 0, & ij\in E.
\end{array}
$$

The subproblems associated to the two variants of the problem differ in their objective function. For problem HopR1', (11) becomes:

$$\max \quad K\pi_d^{L+1}-\sum_{ij\in E} v_{ij}\bar{z}_{ij}\sigma_{ij} \tag{12}$$

while for problem HopR2', we replace (11) by:

$$\max \quad K\pi_d^{L+1}-\sum_{ij\in E}\sum_{s=1}^{2} v^s\bar{z}_{ij}^s\sigma_{ij} \tag{13}$$

For both problems and for a particular commodity $q\in Q$, given a vector $\bar{z}$ 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 $SP(q,\bar{z})$ 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 problem HopR1', Benders cuts are:

$$K\overline{\pi}_{d(q)}^{L+1}-\sum_{ij\in E}\overline{\sigma}_{ij}v_{ij}z_{ij}\le 0 \tag{14}$$

and for problem HopR2', Benders cuts are:

$$K\overline{\pi}_{d(q)}^{L+1}-\sum_{ij\in E}\sum_{s=1}^{2}\overline{\sigma}_{ij}v^s z_{ij}^s\le 0 \tag{15}$$

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 among with one (x variables) is large. A first natural approach consists in solving this kind of huge extended models with a commercial solver like CPLEX. But the big 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 $\bar{z}$ 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 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. Using the strategy described in Botton et al (2011), a weak cut 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. Therefore, we proposed in Botton et al (2011) to embed Benders cuts in a branch-and-cut algorithm. We have already deeply investigated this algorithmic aspect in Botton et al (2011) 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 $s_q = SP(q,\overline{z})$;
            if $s_q > 0$ then add (14) or (15) 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 $f_q = FP(q,\overline{z})$ (see Botton et al (2011));
            if $f_q > 0$ for some $q \in Q$ then add weak cuts (see Botton et al (2011)) 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 Botton et al (2011), 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 Botton et al (2011). The linear relaxation at the root node of the branch-and-bound tree usually has a big number of 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).

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 solutions obtained by the upgrading variant in contrast to the solutions obtained by the original problem and 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 $G = (V,E)$, reported in Botton et al (2011). 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 Orlowski et al (2010): 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.

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

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 cost of a reliable edge ij is given by $((K-1)+f)$ times the base cost, where f is a fractional number between 0 and 1. This allows to model different economy of scale scenarios for the use of reliable edges. 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 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

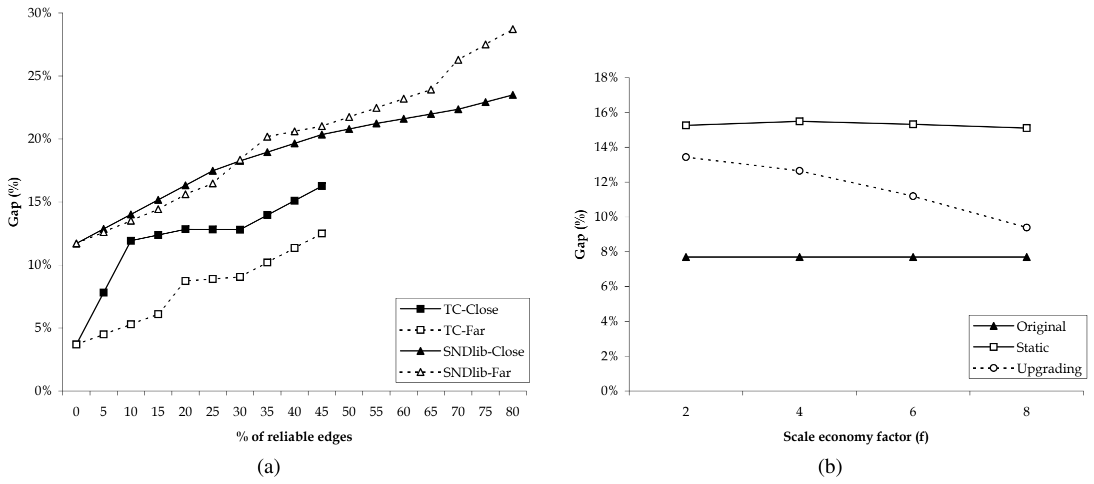


Fig. 4 Evolution of the gap (%) in function of the percentage of reliable edges (a) and depending on the scale economy factor f (b).

The two variants studied in this paper can be solved with the hop-indexed model developed in Botton et al (2011) with simple modifications. The first evaluation criterion is given by the 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 Botton et al (2011).
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 arcs.

Tables 2, 3, 4 and 5 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 3 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

Table 2 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 3 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 4 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 5 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

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.

Another evaluation criteria is related to the scale economy factor f. On Table 6 we show the average gap (%) obtained for the static and upgrading models and on Table 7 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.

Table 6 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 7 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 8 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 8 shows clearly 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

Table 9 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

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 Botton et al (2011) 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 9 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 10 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

Table 10 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 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.

6 Conclusion

In this paper we have proposed 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 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 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 Botton et al (2011) and 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 easily modeled, by creating z 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 an “Actions de Recherche Concertées” project of the “Communauté française de Belgique” (ARC 06/11-346).

The work of Quentin Botton and Bernard Fortz is also supported by a FRFC program “Groupe de Programmation Mathémathique” (2.4547.08).

The work of Luis Gouveia is supported by the grant MATH-LVT-Lisboa-152.

References

Bley A (1997) Node-disjoint length-restricted paths. Diploma thesis, TU Berlin

Botton Q, Fortz B, Gouveia L, Poss M (2011) Benders decomposition for the hop-constrained survivable network design problem, DOI http://dx.doi.org/10.1287/ijoc.1110.0472, to appear in *Informs Journal on Computing*

Croxton KL, Gendron B, Magnanti TL (2003) A comparison of mixed-integer programming models for nonconvex piecewise linear cost minimization problems. Management Science 49(9):1268–1273

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

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

Itaí A, Perl Y, Shiloach Y (1982) The complexity of finding maximum disjoint paths with length constraints. Networks 2:277–286

Kerivin H, Mahjoub A (2005) Design of survivable networks: A survey. Networks 46(1):1–21

Orlowski S, Wessäly R, Pióro M, Tomaszewski A (2010) Sndlib 1.0survivable network design library. Networks 55(3):276–286