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

Quentin Botton, Bernard Fortz and Luis Gouveia

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 nonnegative 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 addition, we consider here a subset of edges that are not subject to failure. We propose 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 study extended formulations for the two variants and we show that considering reliable edges makes the problem more complex. The huge number of variables and constraints linked to our extended formulations is a drawback. Therefore we propose to use Benders decomposition to accelerate the solving process. We develop an exact branch-and-cut algorithm and a fix-and-branch heuristic. From a numerical point of view we show how the original problem (without reliable edges) becomes more complex when reliable edges are added and we conclude with an economical analysis which evaluates the incentive to use reliable edges in network design.

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 $Q$. 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 path is a $L$-path if it contains no more than $L$ arcs. A collection $P_1, P_2, ..., P_k$ of $od$-L-paths is called edge-disjoint if any arc $(i, j)$ and its symmetric arc $(j, i)$ appear in at most one path. This problem 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
requirement for using disjoint paths guarantees the existence of a working path for each commodity after edge-disjoint paths and including a limit on the number of arcs for each path of each commodity. Briefly, the several variants of this problem as well as techniques for solving them. There are several reasons for requiring have been proposed and tested in Botton et al (2011). The survey by Kerivin and Mahjoub (2005) describes decomposition approach to efficiently handle the large number of variables and constraints of the formulation

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 at most 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. Furthermore, to make their use attractive, it also makes sense to integrate economies of scale, that is, if $K$ paths are allowed to pass through a reliable edge $ij$, we want the cost of edge $ij$ to be less than $K$ times the base cost of the same edge. We use an economy of scale factor $f$ defined in the interval $[0,1]$ and we set the cost of a reliable edge as $((K-1) + f)$ times the base cost of edge $ij$. As an example, assume that $K = 2$, $f = 0.2$, the edge $ij$ is reliable and the cost of edge $ij$ assuming it is unreliable is equal to $c_{ij} = 10$. The cost of the reliable edge $ij$ is equal to 12 which is lower than a value equal to $K$ times the normal cost of the edge (that would be equal to 20).

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, 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).

The paper is organized as follows: in Section 2, we first review the model proposed in Botton et al (2011) for the traditional 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 developed. 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, where $V^q = V^q_1 \cup \ldots \cup V^q_{L+1}$ with $V^q_1 = \{o(q)\}$, $V^q_{L+1} = \{d(q)\}$ and $V^q_l = V \setminus \{o(q)\}, \; l = 2,\ldots,L$. Let $v^q_i$ be the copy of $v \in V$ in the $l$-th layer of graph $G^q$. Then, the arcs sets are defined by $A^q = \{(i^q_l,j^q_{l+1}) \mid i^q_l \in E, j^q_l \in V^q_l, j^q_{l+1} \in V^q_{l+1}, l \in \{1,\ldots,L\} \cup \{d(q),d(q)l+1,q\}, l \in \{2,\ldots,L\}\}$, see Fig. 1. In the sequel, an (undirected)
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 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:

\[
\text{(Hop)} \quad \min \sum_{ij \in E} c_{ij} z_{ij}
\]

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

\[
\sum_{j \in \{1, \ldots, L\}} (x_{ij}^{l,q} + x_{ji}^{l,q}) \leq z_{ij}, \quad ij \in E, q \in Q, \quad (2)
\]

\[
z_{ij} \in \{0, 1\}, \quad (i, j) \in A^q, q \in Q, \quad (3)
\]

\[
x_{ij}^{l,q} \geq 0 \text{ and integer}, \quad (i, j, l) \in A^q, q \in Q, \quad (4)
\]

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 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 $x_{il}^{l,q} \leq K$.

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.
\[
\text{(HopR1)} \quad \min \sum_{ij \in E} p_{ij} c_{ij} z_{ij} \\
\text{s.t.} \quad (1), (3), (4), \quad \sum_{k \in \{1, \ldots, L\}} (x_{ij}^k + x_{ji}^k) \leq v_{ij} z_{ij}, \quad ij \in E, q \in Q, \quad (5)
\]

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 \(ij \in E_R\) and the value \(K\) if the edge is reliable \(ij \in E_K\). In the later case the edge may be used by all \(K\) paths of each given commodity. Note also that in this model the \(x_{ij}^k\) variables for reliable edges are no longer restricted to be equal to 0 or 1. 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). 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.

2.3 The upgrading problem

In this version of the problem, all edges \(ij\) are considered as normal edges \((E_R = E)\), with an associated base cost \(c_{ij}\). The subset of reliable edges \((E_R \subset E_R)\) contains the edges that can be upgraded to be reliable, with total cost \((K - 1) + f\) \(c_{ij}\).

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 \(z_{ij}^s = 1\), if edge \(ij\) is included in the solution as a normal edge and \(z_{ij}^s = 1\) 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 cost is also multiplied by a coefficient depending on the scenario: \(p^1 = 1\) and \(p^2 = (K - 1) + f\).

\[
\text{(HopR2)} \quad \min \sum_{ij \in E} \sum_{s=1}^{2} p^s c_{ij} z_{ij}^s \\
\text{s.t.} \quad (1), (4), \quad \sum_{k \in \{1, \ldots, L\}} (x_{ij}^k + x_{ji}^k) \leq \sum_{s=1}^{2} v^s z_{ij}^s, \quad ij \in E, q \in Q, \quad (6)
\]

\[
\sum_{s=1}^{2} z_{ij}^s \leq 1, \quad ij \in E, \quad (7)
\]

\[
z_{ij}^s = 0, \quad ij \in E \setminus E_R, \quad (8)
\]

\[
z_{ij}^s \in \{0, 1\}, \quad ij \in E, s \in \{1, 2\}. \quad (9)
\]

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.

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 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.

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 $x_{ij}^q$ in a Benders decomposition approach. The remaining problem called master problem is only dependent of 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

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

As discussed in Botton et al (2011), for $K = 3$, 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 $z$ 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’

\[
\begin{align*}
& \text{(BR-HopR1')} \quad \min & \sum_{ij \in E} p_{ij} c_{ij} z_{ij} \\
& \text{s.t.} & \pi_{d(q)}^{L+1} - \sum_{ij \in E} \sigma_{ij} \psi_{ij} z_{ij} \leq 0, (\pi, \sigma) \in \bigcup_{q \in Q} \mathcal{F}^q, \\
& & \sum_{s=1}^{2} \zeta_{ij}^{s} \leq 1, \quad \pi_{L+1} \leq 1, \quad \sigma_{ij} \geq 0, \\
& & z_{ij} \in \{0,1\}, \quad i \in E,
\end{align*}
\]

3.2 Master problem of HopR2’

\[
\begin{align*}
& \text{(BR-HopR2')} \quad \min & \sum_{ij \in E} \sum_{s=1}^{2} p_{ij} c_{ij} z_{ij}^{s} \\
& \text{s.t.} & \pi_{d(q)}^{L+1} - \sum_{ij \in E} \sum_{s=1}^{2} \sigma_{ij} \psi_{ij} z_{ij}^{s} \leq 0, (\pi, \sigma) \in \bigcup_{q \in Q} \mathcal{F}^q, \\
& & \sum_{s=1}^{2} \zeta_{ij}^{s} \leq 1, \quad \pi_{L+1} \leq 1, \quad \sigma_{ij} \geq 0, \\
& & z_{ij} \in \{0,1\}, \quad i \in E,
\end{align*}
\]

For both master problems $\mathcal{F}^q$ contains the extreme points of the feasibility polyhedron for the dual subproblem $\text{SP}(q, \pi)$ described next.

3.3 Dual subproblems

Given a commodity $q \in Q$, let us introduce a dual variable $\pi_{ij}$, associated with node $i \in V$ and layer $l$, for each constraint (1) and a dual variable $\sigma_{ij}$ for each constraint (5) or (6). Defining $o := o(q)$ and $d := d(q)$, and adding the normalizing constraints $\pi_{l+1}^{o} \leq 1$ and $\pi_{l+1}^{d} = 0$ to normalize the dual cone we get the dual subproblem $\text{SP}(q, \pi)$

\[
\begin{align*}
& \text{SP}(q, \pi) \quad \max & \text{objective function} \quad (11) \\
& \text{s.t.} & \pi_{l}^{2} - \pi_{l}^{1} - \sigma_{oi} \leq 0, & o \in E, \quad i \in E, \\
& & \pi_{l}^{j} - \pi_{l}^{j-1} - \sigma_{ij} \leq 0, & ij \in E, i, j \notin \{o,d\}, l \in \{2,\ldots,(L-1)\}, \\
& & \pi_{l}^{j+1} - \pi_{l}^{j} - \sigma_{ij} \leq 0, & ij \in E, i, j \notin \{o,d\}, l \in \{2,\ldots,(L-1)\}, \\
& & \pi_{l}^{d} - \pi_{l}^{d-1} - \sigma_{id} \leq 0, & id \in E, l \in \{2,\ldots,L\}, \\
& & \pi_{l}^{d+1} - \pi_{l}^{d} \leq 0, & l \in \{2,\ldots,L\}, \\
& & \pi_{l}^{d} = 0, & i \in E, \\
& & \pi_{l}^{d+1} \leq 1, & i \in E, \\
& & \sigma_{ij} \geq 0, & i \in E.
\end{align*}
\]

The subproblems associated to the two variants of the problem differ in there objective function. For problem HopR1’, (11) becomes:
On the hop-constrained survivable network design problem with reliable edges

\[
\max \quad K \pi_d^{L+1} - \sum_{i,j \in E} v_{ij} x_{ij} \sigma_{ij} \quad (12)
\]

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

\[
\max \quad K \pi_d^{L+1} - \sum_{i,j \in E} \sum_{s=1}^{2} s^s x_{ij} \sigma_{ij} \quad (13)
\]

For both problems and for a particular commodity \( q \in Q \), given a vector \( \pi \) 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 \( SP(q, \pi) \) 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:

\[
K \pi_d^{L+1} - \sum_{i,j \in E} \pi_{ij} v_{ij} x_{ij} \leq 0 \quad (14)
\]

and for problem HopR2’, Benders cuts are:

\[
K \pi_d^{L+1} - \sum_{i,j \in E} \sum_{s=1}^{2} \pi_{ij} s^s x_{ij} \sigma_{ij} \leq 0 \quad (15)
\]

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 \( \pi \) 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.

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

begin
    /* Initialization */
    T = {o} where o has no branching constraints;
    UB = +∞;
end

while T is nonempty do
    select a node o′ ∈ T;
    T ← T \ {o′};
    /* withdraw node o′ from the tree */
    solve o′;
    let z be an optimal solution;
    let w be the optimal cost;
    if w < UB then
        if z ∈ {0, 1} | E | or depth(o′) ≤ n then
            foreach q ∈ Q do compute s_q = SP(q, z);
            if s_q > 0 then add (14) or (15) to (MP);
        end
        if z ∈ {0, 1} | E | and s_q ≤ 0 for each q ∈ Q then
            foreach q ∈ Q do compute f_q = FP(q, z) (see Botton et al (2011));
            if f_q > 0 for some q ∈ Q then add weak cuts (see Botton et al (2011)) to (MP);
            else
                UB ← z;
                /* define a new upper bound */
                z∗ ← z;
                /* save current incumbent */
            end
            if s_q > 0 or f_q > 0 for some q ∈ Q then
                branch, resulting in nodes o∗ and o∗∗;
                T ← T ∪ {o∗, o∗∗};
                /* add children to the tree */
            else if z ∉ {0, 1} | E | then
                branch, resulting in nodes o∗ and o∗∗;
                T ← T ∪ {o∗, o∗∗};
                /* add children to the tree */
            end
        end
    end
    else
        UB ← w;
        /* define a new upper bound */
        z∗ ← z;
        /* save current incumbent */
    end

return z∗

Furthermore, the algorithm is improved with a good upper 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 11 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 \((100 \times 100)\) 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 five instances that are based on sparse networks from SNDlib Orlowski et al (2010): pdh, di-yuan, dfw-gwin, polska, nobel-us. 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.

| Name   | \(|N|\) | \(|E|\) | \(|Q|\) | \# of instances | Rooted demands? |
|--------|--------|--------|--------|-----------------|-----------------|
| TC-5   | 21     | 210    | 5      | 5               | true            |
| TC-10  | 21     | 210    | 10     | 5               | true            |
| pdh    | 11     | 34     | 27     | 1               | false           |
| di-yuan| 11     | 42     | 48     | 1               | false           |
| dfw-gwin| 11   | 47     | 9      | 1               | false           |
| polska | 12     | 18     | 17     | 1               | false           |
| nobel-us| 14    | 21     | 33     | 1               | 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 \(( (K - 1) + f ) \times \text{base cost} \), where \( f \) is a fractional number between 0 and 1. 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 \( c_g \) and \( c \). The parameter \( c_g \) 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 \( c_g \) and \( c \) additional nodes which are the closest nodes from the center of gravity node \( c_g \). 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 \( c \in \{0, 1, 2, 3\} \). The Far strategy is also defined with two parameters \( c_g \) and \( b \). As before, the parameter \( c_g \) 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 \( c_g \). 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 \( c_g \).

5.3 Evaluation of the models

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 root gaps) related to 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.

### Table 2: Arithmetic average of gap (%) depending on the number of commodities.

<table>
<thead>
<tr>
<th>Instance</th>
<th>Q</th>
<th>Original</th>
<th>Static</th>
<th>Upgrading</th>
</tr>
</thead>
<tbody>
<tr>
<td>TC</td>
<td>5</td>
<td>3.22</td>
<td>12.54</td>
<td>9.27</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>6.80</td>
<td>18.84</td>
<td>13.30</td>
</tr>
<tr>
<td>pdh</td>
<td>27</td>
<td>12.54</td>
<td>27.91</td>
<td>18.70</td>
</tr>
<tr>
<td>di-yuan</td>
<td>48</td>
<td>13.62</td>
<td>28.04</td>
<td>18.49</td>
</tr>
<tr>
<td>dfn-gwin</td>
<td>9</td>
<td>9.90</td>
<td>21.17</td>
<td>14.36</td>
</tr>
<tr>
<td>polska</td>
<td>11</td>
<td>3.18</td>
<td>9.90</td>
<td>6.43</td>
</tr>
<tr>
<td>Average</td>
<td></td>
<td>6.39</td>
<td>17.92</td>
<td>12.56</td>
</tr>
</tbody>
</table>

### Table 3: Arithmetic average of gap (%) depending on the localization of the reliable edges set.

<table>
<thead>
<tr>
<th>Geographical distribution</th>
<th>Problems</th>
<th>Static</th>
<th>Upgrading</th>
</tr>
</thead>
<tbody>
<tr>
<td>Close</td>
<td></td>
<td>19.53</td>
<td>13.41</td>
</tr>
<tr>
<td>Far</td>
<td></td>
<td>16.33</td>
<td>11.72</td>
</tr>
</tbody>
</table>

### Table 4: Arithmetic average of gap (%) depending on the number of edge-disjoint paths $K$.

<table>
<thead>
<tr>
<th>K</th>
<th>Problems</th>
<th>Original</th>
<th>Static</th>
<th>Upgrading</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td></td>
<td>7.63</td>
<td>17.36</td>
<td>14.17</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>4.94</td>
<td>17.98</td>
<td>10.78</td>
</tr>
</tbody>
</table>

### Table 5: Arithmetic average of gap (%) depending on the hop limit $L$.

<table>
<thead>
<tr>
<th>L</th>
<th>Problems</th>
<th>Original</th>
<th>Static</th>
<th>Upgrading</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td></td>
<td>7.48</td>
<td>18.93</td>
<td>13.37</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>6.58</td>
<td>17.76</td>
<td>12.57</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td>5.19</td>
<td>17.10</td>
<td>11.77</td>
</tr>
</tbody>
</table>

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

---

**Fig. 4** Evolution of the gap (%) in function of the percentage of reliable edges (a) and depending on the scale economy factor $f$ (b)
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 models HopR1 and HopR2 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 problem HopR2, 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>
<thead>
<tr>
<th>Network</th>
<th>( K )</th>
<th>0.2</th>
<th>0.4</th>
<th>0.6</th>
<th>0.8</th>
</tr>
</thead>
<tbody>
<tr>
<td>HopR1</td>
<td>TC</td>
<td>15.79</td>
<td>15.78</td>
<td>15.32</td>
<td>14.83</td>
</tr>
<tr>
<td></td>
<td>TC</td>
<td>16.53</td>
<td>16.15</td>
<td>15.79</td>
<td>15.31</td>
</tr>
<tr>
<td></td>
<td>SNDlib</td>
<td>23.72</td>
<td>23.97</td>
<td>23.82</td>
<td>23.46</td>
</tr>
<tr>
<td></td>
<td>SNDlib</td>
<td>25.52</td>
<td>25.51</td>
<td>25.34</td>
<td>25.11</td>
</tr>
<tr>
<td>Average</td>
<td>HopR1</td>
<td>18.28</td>
<td>18.17</td>
<td>17.82</td>
<td>17.39</td>
</tr>
<tr>
<td>HopR2</td>
<td>TC</td>
<td>15.03</td>
<td>13.50</td>
<td>10.93</td>
<td>8.05</td>
</tr>
<tr>
<td></td>
<td>TC</td>
<td>13.60</td>
<td>11.73</td>
<td>9.79</td>
<td>7.64</td>
</tr>
<tr>
<td></td>
<td>SNDlib</td>
<td>22.36</td>
<td>20.91</td>
<td>18.91</td>
<td>16.66</td>
</tr>
<tr>
<td></td>
<td>SNDlib</td>
<td>14.16</td>
<td>12.06</td>
<td>10.06</td>
<td>8.15</td>
</tr>
<tr>
<td>Average</td>
<td>HopR2</td>
<td>15.53</td>
<td>13.83</td>
<td>11.64</td>
<td>9.23</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Network</th>
<th>( K )</th>
<th>0.2</th>
<th>0.4</th>
<th>0.6</th>
<th>0.8</th>
</tr>
</thead>
<tbody>
<tr>
<td>HopR1</td>
<td>TC</td>
<td>29.58</td>
<td>23.32</td>
<td>19.02</td>
<td>16.67</td>
</tr>
<tr>
<td></td>
<td>TC</td>
<td>26.41</td>
<td>23.83</td>
<td>23.19</td>
<td>20.94</td>
</tr>
<tr>
<td></td>
<td>SNDlib</td>
<td>8.55</td>
<td>7.12</td>
<td>5.82</td>
<td>4.38</td>
</tr>
<tr>
<td></td>
<td>SNDlib</td>
<td>3.11</td>
<td>2.95</td>
<td>2.44</td>
<td>2.17</td>
</tr>
<tr>
<td>Average</td>
<td>HopR1</td>
<td>22.46</td>
<td>18.94</td>
<td>16.85</td>
<td>14.89</td>
</tr>
<tr>
<td>HopR2</td>
<td>TC</td>
<td>76.96</td>
<td>66.98</td>
<td>62.70</td>
<td>54.79</td>
</tr>
<tr>
<td></td>
<td>TC</td>
<td>63.62</td>
<td>56.73</td>
<td>53.84</td>
<td>45.39</td>
</tr>
<tr>
<td></td>
<td>SNDlib</td>
<td>84.59</td>
<td>62.13</td>
<td>35.62</td>
<td>27.79</td>
</tr>
<tr>
<td></td>
<td>SNDlib</td>
<td>1.34</td>
<td>1.13</td>
<td>1.03</td>
<td>0.96</td>
</tr>
<tr>
<td>Average</td>
<td>HopR2</td>
<td>65.43</td>
<td>55.68</td>
<td>48.94</td>
<td>41.64</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Formulations</th>
<th>CPU time (sec.)</th>
<th>Not solved to optimality</th>
<th>Final gap (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hop</td>
<td>214.67</td>
<td>0 ( \backslash ) 18</td>
<td>0</td>
</tr>
<tr>
<td>HopR1</td>
<td>262.41</td>
<td>0 ( \backslash ) 660</td>
<td>0</td>
</tr>
<tr>
<td>HopR2</td>
<td>496.53</td>
<td>22 ( \backslash ) 660</td>
<td>5.31</td>
</tr>
<tr>
<td>bc-5-heur</td>
<td>5.73</td>
<td>0 ( \backslash ) 18</td>
<td>0</td>
</tr>
<tr>
<td>bc-5-heurR1</td>
<td>4.95</td>
<td>0 ( \backslash ) 660</td>
<td>0</td>
</tr>
<tr>
<td>bc-5-heurR2</td>
<td>31.96</td>
<td>0 ( \backslash ) 660</td>
<td>0</td>
</tr>
</tbody>
</table>

CPU times were analysed only on the SNDlib test set. Table 8 shows that in general, the original and the static problems take similar amounts of time to reach optimality for both the associated extended formulations and the branch-and-cut algorithms. In the first variant, the associated branch-and-cut algorithms bc-5-heurR1 again outperforms 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. On the other hand, the second variant seems more difficult. The proposed HopR2 formulation is more difficult to solve both in the context of using it directly within CPLEX and using the branch-and-cut algorithm. Note also that 22 instances out of 660 are not solved to optimality within the 3600 seconds time limit and an average gap of approximately 5% remains to be closed. However, these 22 instances are easily solved to optimality by the proposed branch-and-cut bc-5-heurR2.

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 9 Savings obtained by the utilization of reliable edges.

<table>
<thead>
<tr>
<th></th>
<th>Upgrading &lt; Original</th>
<th>Upgrading &lt; Static</th>
</tr>
</thead>
<tbody>
<tr>
<td>(%) of all the instances</td>
<td>47.67</td>
<td>83.23</td>
</tr>
<tr>
<td>Arithmetic average of savings realized (%)</td>
<td>4.46</td>
<td>7.15</td>
</tr>
<tr>
<td>Maximum savings realized (%)</td>
<td>27.23</td>
<td>34.02</td>
</tr>
</tbody>
</table>

Table 9 shows that in approximately 48% 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 4.46% of the total design cost when reliable edges are not included. Moreover in approximately 83% 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 7.15% 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 this testing phase is that the original problem becomes more difficult when the proportion of reliable edges increases as the gap increases. We can also conclude that for the upgraded 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 only dealing 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 Hop1 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 This research is supported by an “Actions de Recherche Concertées” (ARC) projet of the “Communauté française de Belgique” and a FRFC program “Groupe de Programmation Mathématique”. Part of this research was conducted while Quentin Botton visited the University of Lisbon, supported by the “Concours Bourses de Voyage” of the “Communauté française de Belgique” and Marie-Curie Research Training Network “ADONET”. The work of Luis Gouveia is supported by the grant MATH-LVF-Lisboa-152.
References


