Benders decomposition for the hop-constrained survivable network design problem

Quentin Botton
Center for Supply Chain Management (CESCM), Louvain School of Management, Université catholique de Louvain, Louvain-la-Neuve, Belgium, and CORE, Université catholique de Louvain, Louvain-la-Neuve, Belgium, quentin.botton@uclouvain.be

Bernard Fortz
GOM, Department of Computer Science, Faculté des Sciences, Université Libre de Bruxelles, Brussels, Belgium, bfortz@ulb.ac.be

Luis Gouveia
Faculdade de Ciências da Universidade de Lisboa, DEIO, CIO Bloco C6, Campo Grande, 1749-016 Lisbon, Portugal, legouveia@fc.ul.pt

Michael Poss
GOM, Department of Computer Science, Faculté des Sciences, Université Libre de Bruxelles, Brussels, Belgium, mposs@ulb.ac.be

Given a graph with nonnegative edge weights and node pairs Q, we study the problem 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. Using the layered representation introduced by Gouveia (1998), we present a formulation for the problem valid for any $K, L \geq 1$. We use a Benders decomposition method to efficiently handle the big number of variables and constraints. We show that our Benders cuts contain the constraints used by Huygens *et al.* to formulate the problem for $L = 2, 3, 4$, as well as new inequalities when $L \geq 5$. While some recent works on Benders decomposition study the impact of the normalization constraint in the dual subproblem, we focus here on when to generate the Benders cuts. We present a thorough computational study of various branch-and-cut algorithms on a large set of instances including the real based instances from *SNDlib*. Our best branch-and-cut algorithm combined with an efficient heuristic is able to solve the instances significantly faster than CPLEX 12 on the extended formulation.

Key words: survivable network; edge-disjoint paths; hop-constrained paths; Benders decomposition; branch-and-cut algorithm

---

1. Introduction

1.1 Problem motivation

Our worldwide society is largely dependent on the performance of huge information systems which are often organized in large-scale, complex and costly networks. With time, the equipment (routers, fiber optic cables, ...) deteriorates and the risk of failure must be controlled as well as possible by the network managers in order to guarantee the best service to users. As a consequence, the development of survivable networks became a crucial field of research and investigation. In this

paper, we define a survivable network as a network in which the various demands can be routed without loss of service quality even in case of network failure (link or node failure).

For each demand, we impose that at least K different paths exist for each origin-destination pair. These K paths can, for instance, be "edge-disjoint", i.e. if a particular edge belongs to one path, this particular edge cannot be used by the other $K-1$ paths. This guarantees that if $K-1$ edges break down, it is always possible to reroute all the demands by the K-th path which does not use the broken arcs. Another form of survivability considers the node-disjoint case. More formally consider an undirected graph $G=(V,E)$, where V represents the vertex set, and E the set of edges. We also associate an installation cost c_{ij} to each edge ij and introduce an auxiliary arc set A which is obtained from each edge ij of E by creating two arcs (i,j) and (j,i) with the same cost as the original edge. In order to incorporate the survivability considerations into the problem definition, we need to introduce the following graph theoretical concepts with elements from the sets *V* and *A*. Given two distinct nodes o (the origin vertex of demand) and d (the destination vertex of demand) of V, 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-paths is called edge-disjoint if any arc (i,j) and its symmetric arc (j,i) appears in at most one path. It is called node-disjoint if any node except for o and d appears in at most one path. A subgraph H of G is called K-edge-survivable (respectively, K-node-survivable) if for any $o,d\in V$, H contains at least a specified number K of edge-disjoint (respectively, node-disjoint) od-paths. Then, the K-edge-survivable network design problem, denoted by $ESNDP$, consists of finding a K-edge-survivable subgraph of G with minimum total cost, where the cost of a subgraph is the sum of the costs of its edges. Similarly, the node-survivable network-design problem, denoted by $NSNDP$, consists of finding a minimum-cost node-survivable subgraph of G. A polyhedral study of the problem for the K-$ESNDP$ with $K=2$ can be found in Stoer (1993) while the node variant is, among others, addressed in Grötschel et al. (1992) and reviewed in Fortz (2000). The reader is also referred to Raghavan (1995) for a discussion on flow-based models and projection from flow based models to original arc variables and to Magnanti and Raghavan (2005) for enhancements on standard flow based models.

In general, the survivability constraints alone may not be sufficient to guarantee a cost effective routing with a good quality of service. The reason for this is that the routing paths may be too "long", leading to unacceptable delays. Since in most of the routing technologies, delay is caused at the nodes, 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). Thus, to guarantee the required quality of service,

we impose a limit on the number of arcs of the routing paths. Hop-constraints were considered by Balakrishnan and Altinkemer (1992) as a means of generating alternative base solutions for a network design problem. Later on, Gouveia (1998) presented a layered network flow reformulation whose linear programming bound proved to be quite tight. This reformulation has, then, been used in several network design problems with hop-constraints (e.g, Pirkul and Soni (2003), Gouveia and Magnanti (2003) and Gouveia et al. (2003)) and even some hop-constrained problems involving survivability considerations (more on this below). It is also interesting to point out that the apparently simple general network design problem with $L = 2$ already contains a complex structure (see Dahl and Johannessen (2004) who also conduct a computational study of this variation of the problem).

In this paper, we study a problem which incorporates the two requirements, survivability and quality of service. More precisely, given a graph G and two parameters K and L, we consider an extension of the $ESNDP$ where each path is constrained to have at most L arcs. We note that this is not the first time that the two types of constraints are considered together. As far as we know, the earliest work that combined hop-constraints with the constraint that the required paths must be node-disjoint is the bounded ring network design problem studied by Fortz and Labbé (2002, 2004); Fortz et al. (2006), among others. The problem we study was first studied by Huygens et al. (2007) who only consider $L \leq 4$ and $K = 2$. The node-disjoint variant was studied by Gouveia et al. (2006) and later in Gouveia et al. (2008) who consider a more complicated version. The reader is referred to the survey by Kerivin and Mahjoub (2005) who consider the disjoint path case alone, network design problems only with hop constraints and the case where the two requirements are considered together.

1.2 Model and method motivation

Relevant for obtaining the good computational results for the K-$ESNDP$ is the fact that most of the best methods rely on so-called natural models, that is, models that use only one variable for each edge of the graph and an exponential sized set of constraints. However, for many of these inequalities the associated separation problem is well solved and thus, they can be efficiently separated leading to quite good cutting plane algorithms as shown for instance in Dahl and Stoer (1998). Unfortunately, finding a similar approach for the same type of problem with hop-constraints is much more complicated. The reason is that it is not straightforward to obtain a valid natural formulation for the particular case of finding a set of edges containing K edge-disjoint L-paths between the two given nodes. Itaí et al. (1982) and later Bley (1997) study the complexity of this problem

for the node-disjoint and the edge-disjoint cases. Recently, Bley and Neto (2010) also studied the approximability of the problem for $L = 3$ and $L = 4$.

For $K = 1$, Dahl (1999) has provided such a formulation and shown that it describes the corresponding convex hull for $L \leq 3$. Later on, Dahl et al. (2004) have shown that finding such a description for $L \geq 4$ would be much more complicated. For $K \geq 2$ the results are even worse. Huygens et al. (2004) have extended Dahl's result for $K = 2$ and $L \leq 3$. For $L \geq 4$, the only interesting result for the moment is the one given in Huygens and Mahjoub (2007) for $L = 4$ and $K = 2$ where a valid formulation has been given. However, in terms of valid inequalities and with exception to the well known L-path cut inequalities, nothing is known for larger values of L. This may also explain why the only cutting plane method for the more general problem with several sources and several destinations by Huygens et al. (2007) only considers $L \leq 3$.

Based on this history, it makes sense to look for alternative ways of formulating the problem. The layered approach described previously appears to be a good candidate for formulating the problem since it is easily generalized for the case with K disjoint paths. Furthermore, a similar approach has already been used for the version of the problem with node disjoint paths (see Gouveia et al. (2006) and Gouveia et al. (2008) for a more complicated version) and the results in these papers (although for a slightly more complicated variant) give a sort of motivation for the method developed and tested in this paper:

(i) the linear programming bound given by the model (if it can be solved) is often very good;

(ii) however, the model is difficult to use in a straightforward way with CPLEX because it has too many variables.

Thus, (i) and (ii) motivate the approach of using this kind of model within a decomposition algorithm, such as Benders decomposition. Finally, another outcome of this research is that the Benders cuts might give some relevant information for finding valid inequalities for the cases with $L \geq 5$.

Based on these observations, we formulate the problem as an integer program based on the layered representation from Gouveia (1998). To our knowledge, this is the first formulation for the problem that is valid for $L \geq 5$ and any $K > 1$.

As previously mentioned the model is too large to be used directly with CPLEX (or for that matter any other solver). Hence, we use a Benders decomposition method to efficiently handle the large number of variables and constraints. Although Benders decomposition has been widely used for hard mixed-integer problems — including fixed-charge network design problems (Costa, 2005)

— not much is said about the algorithmic aspects, most authors using "textbook implementations". Some recent works (Fischetti et al., 2010; Ljubic et al., 2009) have highlighted the importance of the normalization constraint in the separation problem. Herein, we investigate another aspect of the algorithm, namely, when to generate cuts throughout a branch-and-cut algorithm. We present a thorough computational study of branch-and-cut algorithms on a large set of instances including the real-world based instances from *SNDlib* (Orlowski et al., 2010). Some computational experiments (not reported here) also confirm previous results obtained by Fortz and Poss (2009) showing that branch-and-cut algorithms outperform cutting plane algorithms.

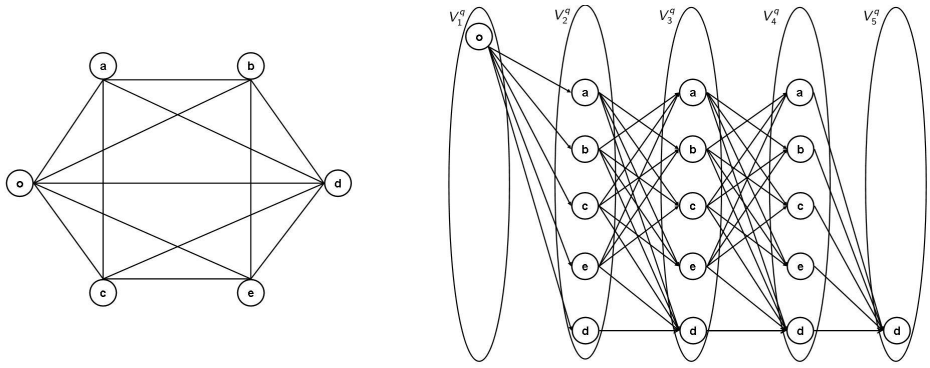
Another outcome of our research is that the Benders cuts may give some relevant information for finding valid inequalities for the cases with $L \geq 5$. We show that the Benders cuts contain the constraints used by Huygens et al. (2004) and Huygens and Mahjoub (2007) to formulate the problem for $L = 2, 3, 4$ and $K = 2$ in the space of natural design variables, as well as new valid inequalities when $L \geq 5$. Hence, for $L = 2, 3$ our branch-and-cut algorithms (polynomially) separate "cut inequalities" and "L-paths inequalities" while Huygens et al. (2007) need to separate both inequality types independently. Finally, we present a fast and efficient LP-based heuristic that provides the optimal solution for more than half of the instances.

In the next section we introduce the layered representation and describe our integer programming formulation. In Section 3, we reformulate the problem through Benders decomposition and discuss different algorithmic approaches. Section 4 compares the Benders cuts with previous known cuts for the problem, while computational results are presented in Section 5.

2. Problem description

The main idea of Gouveia (1998) is to model the subproblem associated with each commodity with a directed graph composed of $L+1$ layers as illustrated in Figure 1. Namely, 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 $V^q = V_1^q \cup \ldots \cup V_{L+1}^q$ 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 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 Figure 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



(a) (b)

Figure 1: Original network (a) and its alternative (or associated) layered representation (b) when $L = 4$

arcs (hops), which corresponds to a maximum of L edges (hops) in the original one. In fact this is the main idea of this transformation, since in the layered graph any path is feasible with respect to the hop-constraints. The usual network flow equations defined in this layered graph yield the following model:

$$
\begin{aligned}
\min \quad & \sum_{ij \in E} c_{ij} Z_{ij} \\
\text{s.t.} \quad & \sum_{j:(j,i,l-1)\in A^q} U_{ji}^{l-1\,q} - \sum_{j:(i,j,l)\in A^q} U_{ij}^{lq} = \begin{cases} -K & if \quad (i = o(q)) \\ K & if \quad (i = d(q)) \; and \; (l = L+1) \\ 0 & else \end{cases}, \\
& \qquad\qquad i \in V^q, l \in \{2, \ldots, L+1\}, q \in Q, \qquad (1) \\
& \sum_{l\in\{1,\ldots,L\}} \left(U_{ij}^{lq} + U_{ji}^{lq}\right) \le Z_{ij}, \qquad ij \in E, q \in Q, \qquad (2) \\
& Z_{ij} \in \{0,1\}, \qquad ij \in E, \qquad (3) \\
& U_{ij}^{lq} \quad \text{integer}, \qquad (i,j,l) \in A^q, q \in Q. \qquad (4)
\end{aligned}
\tag{P}
$$

Each variable Z_{ij} states whether edge $ij \in E$ is built and each variable U_{ij}^{lq} describes the amount of flow through arc (i, j, l) for commodity q in layered graph G^q. Constraints (1) are the flow conservation constraints at every node of the layered graph which guarantee that K units of flow go from $o(q)$ to $d(q)$, while constraints (2) guarantee edge-disjointness of the paths. Note that (2) together with (3) imply that $U_{ij}^{lq} \le 1$ for $i \neq j$, while (1) implies that $U_{ii}^{lq} \le K$.

In the sequel, we assume the reader familiar with standard notions of polyhedral theory, see for instance Nemhauser and Wolsey (1988).

3. Benders decomposition

3.1 Reformulation

When facing a complex mixed-integer optimization problem, the Benders decomposition method (Benders, 1962) can be used to project out complicating real variables. This projection results in the addition of many constraints to the problem. Benders decomposition has been widely studied for fixed charge network design problems (Costa, 2005). In these problems, multi-commodity flows are routed in some network to be designed. Therefore, the associated formulations contain many constraints and variables bound together by the capacity constraints. The Benders decomposition of these problems considers a master problem, with capacity variables only, and subproblems with flow variables for one commodity only. Hence, the subproblems are independent linear programs for each commodity (see, for instance, SP($q, \overline{Z}$) below), thus reducing significantly the size of the linear programs to solve. However, the classical framework does not apply to our model (P) because all of its variables are integer; classical duality theory does not allow us to project out variables with integer restrictions. It is well known indeed in the field of stochastic programming that integer recourse cannot be tackled through classical Benders decomposition, called L-shaped in stochastic programming (Birge and Louveaux, 2008). Although Carøe and Tind (1998) generalize the L-shape to integer recourse using general duality theory, their framework stays mainly theoretical.

To avoid this difficulty, we introduce a new formulation for the problem, (P'), where we relax the integrality restrictions on U variables in (P), replacing (4) by

$$U_{ij}^{lq} \geq 0, \qquad (i,j,l) \in A^q, q \in Q. \tag{5}$$

We discuss below and in Section 4 whether (P') provides the same optimal design $\overline{Z}$ as (P). Then, we can use the classical framework (described by Costa (2005) among others) to project out U variables from (P'). Given 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 (2). Defining $o := o(q)$ and $d := d(q)$, and adding the constraints $\pi_o^1 = 0$ and $\pi_d^{L+1} \leq 1$ to normalize the dual cone (see Fischetti et al. (2010); Ljubic et al. (2009) for alternative choices of normalization constraints), we get our dual subproblem SP($q, \overline{Z}$).

Note that for each commodity $q \in Q$, one of the constraints in (1) is redundant, so we set $\pi_o^1 = 0$ to avoid some extreme rays in the dual subproblem.

$$
\begin{aligned}
& \max && K\left(\pi_d^{L+1} - \pi_o^1\right) - \sum_{ij \in E} \overline{Z}_{ij}\sigma_{ij} && && (6)\\
& \text{s.t.} && \pi_i^2 - \pi_o^1 - \sigma_{oi} \leq 0, && && oi \in E,\\
& && \pi_i^{l+1} - \pi_j^l - \sigma_{ij} \leq 0, && && ij \in E, i, j \notin \{o,d\}, l \in \{2,\ldots,(L-1)\},\\
& \text{SP}(q,\overline{Z}) && \pi_j^{l+1} - \pi_i^l - \sigma_{ij} \leq 0, && && ij \in E, i, j \notin \{o,d\}, l \in \{2,\ldots,(L-1)\},\\
& && \pi_d^{l+1} - \pi_i^l - \sigma_{id} \leq 0, && && id \in E, l \in \{2,\ldots,L\},\\
& && \pi_d^{l+1} - \pi_d^l \leq 0, && && l \in \{2,\ldots,L\},\\
& && \pi_o^1 = 0,\\
& && \pi_d^{L+1} \leq 1,\\
& && \sigma_{ij} \geq 0, && && ij \in E.
\end{aligned}
$$

The Benders reformulation for (P') follows:

$$
\begin{aligned}
& \min && \sum_{ij \in E} c_{ij} Z_{ij}\\
\text{(BR)} \qquad & \text{s.t.} && K\overline{\pi}_{d(q)}^{L+1} - \sum_{ij \in E} Z_{ij}\overline{\sigma}_{ij} \leq 0, \quad (\overline{\pi},\overline{\sigma}) \in \bigcup_{q \in Q} \mathcal{R}^q,\\
& && Z_{ij} \in \{0,1\}, \qquad ij \in E,
\end{aligned}
$$

where $\mathcal{R}^q$ contains vertices of the feasibility polyhedron for the dual subproblem SP$(q,\overline{Z})$ (notice that the feasibility polyhedron of SP$(q,\overline{Z})$ depends only on q, not on $\overline{Z}$).

Next, we discuss how to extend this procedure to (P). We need to first introduce some notation. Given $\overline{Z} \in \{0,1\}^{|E|}$, let $\mathcal{U}_i(\overline{Z})$ be the set of binary vectors defined by (1),(2) and (4) and $\mathcal{U}_c(\overline{Z})$ the polyhedron defined by (1),(2) and (5). Then, define $\mathcal{Z}_i$ (respectively $\mathcal{Z}_c$) as the set of vectors $\overline{Z} \in \{0,1\}^{|E|}$ such that $\mathcal{U}_i(\overline{Z})$ (respectively $\mathcal{U}_c(\overline{Z})$) is nonempty, and let conv$(\mathcal{Z}_i)$ (respectively conv$(\mathcal{Z}_c)$) be its convex hull. Since $\mathcal{U}_i(\overline{Z}) \subseteq \mathcal{U}_c(\overline{Z})$ for every binary $\overline{Z}$, we have that $\mathcal{Z}_i \subseteq \mathcal{Z}_c$ so that any valid inequality for conv$(\mathcal{Z}_c)$ is valid for conv$(\mathcal{Z}_i)$. In particular, the Benders cut

$$
K\overline{\pi}_{d(q)}^{L+1} - \sum_{ij \in E} Z_{ij}\overline{\sigma}_{ij} \leq 0, \tag{7}
$$

with $(\overline{\pi},\overline{\sigma}) \in \mathcal{R}^q$ for some commodity $q \in Q$, is valid for conv$(\mathcal{Z}_i)$.

These definitions raise the following question. Are cuts (7) together with integrality restrictions (3) enough to characterize $\mathcal{Z}_i$? Namely, given a binary vector $\overline{Z}$, is it true that $\overline{Z}$ belongs to $\mathcal{Z}_i$ if and only if $\overline{Z}$ does not violate any cut (7)? It is easy to see that this is true when $K = 1$, and we prove in Section 4 that this is also the case for $L = 2, 3$ with any $K \geq 2$, and for $L = 4$ with

$K = 2$. For $L \geq 5$ and $K \geq 2$, sets $\mathcal{Z}_i$ and $\mathcal{Z}_c$ may be different (unless $P = NP$), so that we must in general check the existence of an integer flow in the graph described by $\overline{Z}$. We do so by solving the following problem for each commodity:

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

which is one among many ways to turn the decision problem into an optimization problem. Let $\overline{e}^q$ denote the optimal value of e^q, and $\overline{e}_{\overline{Z}} = \max_{q \in Q} \{\sum_{ij \in E} \overline{e}^q_{ij}\}$. If $\overline{e}_{\overline{Z}} = 0$, $\overline{Z} \in \mathcal{Z}_i$. Otherwise, we can add the inequality

$$\sum_{ij \in E^0(\overline{Z})} Z_{ij} \geq \overline{e}_{\overline{Z}}, \tag{8}$$

with $E^0(\overline{Z}) = \{ij \in E \text{ s.t. } \overline{Z}_{ij} = 0\}$, to move away from the current solution, as explained in the next subsection. Since FP($q, \overline{Z}$) looks for the minimal number of additional edges required by commodity q, it is easy to see that (8) is valid for (P). Notice that (8) can be seen as a reinforced feasibility cut typically used in logic-based Benders decomposition, see Hooker (2000) and Codato and Fischetti (2006) among others. It is interesting to point out that in our computational experiments (see Section 5), we never needed to add such a cut because we did not find a vector $\overline{Z} \in \mathcal{Z}_c \backslash \mathcal{Z}_i$.

3.2 Algorithmic approach

(BR) contains exponentially many constraints (this is a direct consequence of Lemma 1 in Section 4) while only a few of them are active at the optimum. Therefore, we can dynamically generate the required constraints throughout the solution method. Early papers on Benders decomposition for mixed-integer problems use cutting plane algorithms, cycling many times between master integer problems and continuous subproblems. However, modern developments in branch-and-cut frameworks such as the commercial CPLEX (IBM-ILOG, 2009) or the noncommercial SCIP (Achterberg, 2009), among others, have eased the development of a branch-and-cut algorithm to

solve the master problem, incorporating the Benders cut separation in the cutting plane callback. Recent works by Fortz and Poss (2009) and Bai and Rubin (2009) present examples for which there is an important time reduction when using a branch-and-cut algorithm instead of a cutting plane algorithm. Moreover, our experiments for the problem studied herein confirm that branch-and-cut algorithms are an order of magnitude faster than cutting plane algorithms. We describe next our algorithms.

Given a subset R^q of $\mathcal{R}^q$ for each $q \in Q$, and binary vectors $\overline{Z}^s$, $s = 1, \ldots, r$, let us define the master problem

$$
\text{(MP)} \qquad
\begin{array}{lll}
\min & \displaystyle\sum_{ij \in E} c_{ij} Z_{ij} & \\
\text{s.t.} & \displaystyle K\overline{\pi}^{L+1}_{d(q)} - \sum_{ij \in E} Z_{ij}\overline{\sigma}_{ij} \leq 0, & (\overline{\pi}, \overline{\sigma}) \in \displaystyle\bigcup_{q \in Q} R^q, \\
& \displaystyle\sum_{ij \in E^0(\overline{Z}^s)} Z_{ij} \geq \overline{e}_{\overline{Z}^s}, & s = 1, \ldots, r, \\
& Z_{ij} \in \{0, 1\}, & \forall ij \in E.
\end{array}
$$

Our branch-and-cut strategy solves (MP) only once. We aim at embedding the generation of violated feasibility cuts (7) (and (8) if needed) into the branch-and-cut framework solving (MP).

It is important to add many cuts early in the tree to avoid exploration of too many infeasible nodes. However, adding too many unnecessary cuts would slow down the linear programming relaxation at each node. Our first branch-and-cut algorithm, `bc-all`, checks for violated Benders cuts (7) at every node of the tree, while it tests for violated inequality (8) only at integer nodes. As noted by Fortz and Poss (2009), this algorithm is relatively slow, because too many cuts are added and too much time is spent in the solution of SP($q, \overline{Z}$). In `bc-int`, we add as many cuts (7) and (8) as we can find at the root node. Then we start branching and check for further violated cuts at integer nodes only. Finally, we developed a hybrid algorithm `bc-n`, described in Algorithm 1, checking for violated inequality (8) at integer nodes and for violated inequality (7) at integer nodes and nodes with a depth less than or equal to n. Note that `bc-n` generalizes both frameworks since `bc-int` is the same as `bc-0`, and `bc-all` is the same as `bc-`$|E|$.

In Algorithm 1, solving a node $o' \in N$ means solving the linear programming relaxation of (MP), augmented with branching constraints of o', while depth(o') counts the number of branching constraints of o'.

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

begin `/* Initialization */`
  $N = \{o\}$ where o has no branching constraints;
  $UB = +\infty$;
while N *is nonempty* **do**
  select a node $o' \in N$;
  $N \leftarrow N \backslash \{o'\}$; `/* 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 (7) 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 $f_q = FP(q, \overline{Z})$;
      ;
      if $f_q > 0$ *for some* $q \in Q$ **then** add (8) 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**
      $N \leftarrow N \cup \{o'\}$; `/* put node` o' `back in the tree */`
    else if $\overline{Z} \notin \{0,1\}^{|E|}$ **then**
      branch, resulting in nodes o^* and o^{**};
      $N \leftarrow N \cup \{o^*, o^{**}\}$; `/* add children to the tree */`
return Z^*

3.3 Heuristic

An intrinsic drawback of decomposition methods is that the solver does never see the complete model as a whole but only a part at the time, making difficult for the solver to detect and exploit the model structure. It is well known that special structures can help the solution of hard integer programs. For instance, detecting a flow structure within a more complicated problem can be used to add strong cut inequalities (Achterberg and Raack, 2010). For the same reason, it is hard for our MIP solver (CPLEX 12) to find good upper bounds. We present next a simple, yet efficient,

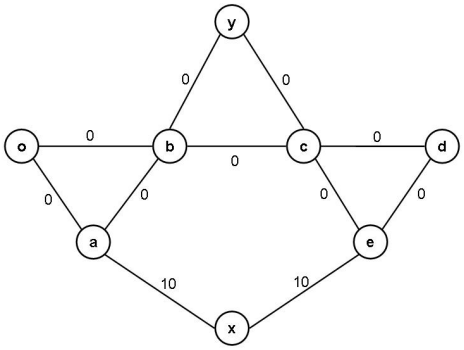


Figure 2: Graph obtained from $\overline{Z}$

heuristic. First, we solve the linear programming relaxation of (P') by a Benders decomposition algorithm, resulting in a fractional $\overline{Z}$. Then, for each $\overline{Z}_{ij} = 0$ we add the constraint $Z_{ij} = 0$ to (MP), and we solve the resulting problem with `bc-n`. This allows us to reduce significantly the number of variables of the problem, yielding a very good solution in a limited amount of time. The issue of whether or not the heuristic finds a feasible solution is discussed in Section 4.

We present in Section 5 the heuristic quality and the solution time of a new branch-and-cut algorithm, `bc-n-heur`, starting with the upper bound from the heuristic.

4. Feasibility problem

Note that the feasibility problems SP($q, \overline{Z}$) and FP($q, \overline{Z}$) and the Benders cuts (7) are independent for each commodity $q \in Q$, so that without loss of generality, we assume in this section that we have a unique commodity q going from o to d. Let us come back to the problem of knowing whether Benders cuts (7) together with integrality restrictions on Z are sufficient to describe $\mathcal{Z}_i$, or in other words, whether $\mathcal{Z}_i = \mathcal{Z}_c$. This is equivalent to knowing whether $\mathcal{U}_i(\overline{Z}) = \emptyset$ implies that $\mathcal{U}_c(\overline{Z}) = \emptyset$, for any binary vector $\overline{Z}$. Note that the inclusion conv($\mathcal{U}_i(\overline{Z})$) $\subseteq \mathcal{U}_c(\overline{Z})$ may be strict. Consider the example with the binary $\overline{Z}$ described in Figure 2, where each edge $ij \in E$ has a routing cost $\tilde{c}_{ij}$, which refers to an example with $L = 4$ and $K = 2$. There are 5 different 4-paths from o to d: the ones shown on Figure 3 and the path $o - b - c - d$. There are only two pairs of disjoint paths, $\{P3, P4\}$ from Figure 3 and $\{P4, o - b - c - d\}$, both with a cost equal to 20. Then, the fractional optimal solution routes 0.5 unit on each path from Figure 3 yielding a total routing cost equal to 10. Thus, the cheapest fractional routing is less than the cheapest integer routing, implying conv($\mathcal{U}_i(\overline{Z})$) $\subset \mathcal{U}_c(\overline{Z})$.

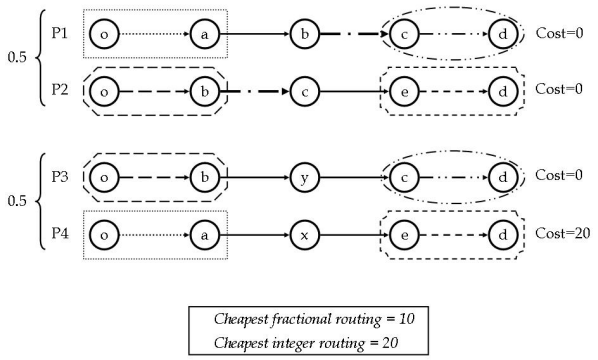


Figure 3: Fractional and integer minimum cost routing when $L = 4$ and $K = 2$.

Let us recall some well-known families of cuts used by Huygens et al. (2004) and Huygens and Mahjoub (2007) to describe a formulation for (P) using only design variables. In what follows, we show that cuts (7) imply these families of cuts. Therefore, using results from Dahl et al. (2006); Diarrassouba (2009) and the Lemmas below, we obtain that Benders cuts together with binary constraints completely describe $\mathcal{Z}_i$ for $L = 2, 3$ and any $K \geq 1$. Then, we give an example (see Figure 4) showing that cuts (7) may be interesting when $L \geq 5$.

We first introduce some notation. If $W \subset V$ is a node subset, then the set of edges that have one node in W and one node in $V \backslash W$ is called a cut and denoted by $\delta(W)$, and $Z(\delta(W)) := \sum_{ij \in \delta(W)} Z_{ij}$. For $o, d \in V$, a cut $\delta(W)$ such that $o \in W$ and $d \in V \backslash W$ is called a *od-cut*. Then, let $V_0, V_1, \ldots, V_{L+1}$ be a partition of V such that $o \in V_0$, $d \in V_{L+1}$ and $V_i \neq \emptyset$ for $i = 1, \ldots, L$. A set of edges $T \subset E$ is called a *L-path-cut* if for each $ij \in T$, $i \in V_v$, $j \in V_w$ such that $|v - w| > 1$. Then, consider the cut inequalities

$$Z(\delta(W)) \geq K, \quad \text{for all } od - cuts\ \delta(W), \tag{9}$$

and the *L-path-cut* inequalities

$$Z(T) \geq K, \quad \text{for all } \textit{L-path-cuts}\ T. \tag{10}$$

Dahl (1999) uses these inequalities for the special case $K = 1$ (for any $L \geq 1$). He proves that a binary vector $\overline{Z}$ belongs to $\mathcal{Z}_i$ if it satisfies (9) and (10). Huygens et al. (2004) extend (9) and (10) to $L = 2, 3$ and $K = 2$, which together with binary restrictions on Z, provide a valid formulation for the problem. In further work, Dahl et al. (2006) and Diarrassouba (2009) prove the formulation to be valid for any $K \geq 2$. Consider now a partition $V_0, V_1, \ldots, V_{L+r}$ of V such that $o \in V_0$,

$d \in V_{L+r}$ and $V_i \neq \emptyset$ for $i = 1, \ldots L + r - 1$. Generalizing (10), Dahl and Gouveia (2004) introduce the *generalized jump* inequality

$$\sum_{i \in V_v, j \in V_w, v \neq w} \min(|v - w| - 1, r) Z_{ij} \geq Kr. \tag{11}$$

Finally, Huygens and Mahjoub (2007) introduce in the *two-layered 4-path-cut* specifically for the case $L = 4$ and $K = 2$. Let $V_0, V_1, \ldots, V_6, W_1, \ldots, W_4$ be a partition of V such that $o \in V_0$, $d \in V_6$ and $V_i \neq \emptyset$ for $i = 1, \ldots 5$. They define the inequality

$$ax \geq 4, \tag{12}$$

with

$$\begin{array}{ll}
a_{ij} = \min(|v - w| - 1, 2), & i \in V_v, j \in V_w, i \neq j, \\
a_{ij} = 2, & i \in W_v, j \in W_w, |v - w| \geq 2, \\
a_{ij} = 2, & i \in V_v, j \in W_w, w - v \geq 2 \text{ or } v - w \geq 3, \\
a_{ij} = 1, & i \in V_v, j \in W_w, (v, w) = (2, 3), (3, 1), (3, 4), (4, 2), \\
a_{ij} = 0, & \text{otherwise.}
\end{array} \tag{13}$$

They have shown that inequalities (12), besides (9) and (10), are needed to obtain a valid formulation for $L = 4$ and $K = 2$.

Next, we prove that cuts (7) imply cuts (9), (11) and (12) (thus (10) because it is a special case of (11)) by showing that all of their coefficients belong to the feasibility polyhedron of one of the subproblems SP($q, \overline{Z}$).

Lemma 1. *Consider some $od - cut\ \delta(W)$. There exists a vector $(\overline{\pi}, \overline{\sigma})$ feasible for SP($q, \overline{Z}$) so that* (7) *written for $(\overline{\pi}, \overline{\sigma})$ is the same as inequality* (9) *for $\delta(W)$. Moreover, $(\overline{\pi}, \overline{\sigma})$ is an extreme point of the polyhedron of feasible solutions of SP($q, \overline{Z}$).*

Proof. Setting $\overline{\pi}_d^{L+1} = 1$, $\overline{\sigma}_{ij} = 1$ for $ij \in \delta(W)$ and 0 otherwise, (7) becomes

$$\sum_{ij \in \delta(W)} Z_{ij} \geq K.$$

Next, we set up $\overline{\pi}_i^l$ for $l < L + 1$ so that $(\overline{\pi}, \overline{\sigma})$ is feasible for SP($q, \overline{Z}$). It is easy to see that $\overline{\pi}_i^l = 0$ for $i \in W$, and $\overline{\pi}_i^l = 1$ for $i \in V \backslash W$, satisfies this requirement.

It remains to show that $(\overline{\pi}, \overline{\sigma})$ is an extreme point of the polyhedron of feasible solutions of SP($q, \overline{Z}$). Let us assume that

$$(\overline{\pi}, \overline{\sigma}) = \frac{1}{2} \left((\pi', \sigma') + (\pi'', \sigma'') \right)$$

for some (π', σ') and (π'', σ'') feasible for SP($q, \overline{Z}$). We show next that we must have $(\pi', \sigma') = (\pi'', \sigma'') = (\overline{\pi}, \overline{\sigma})$, and hence $(\overline{\pi}, \overline{\sigma})$ is an extreme point of the polyhedron of feasible solutions of SP($q, \overline{Z}$).

First note that if $\overline{\sigma}_{ij} = 0$, then, since $\sigma'_{ij} \geq 0$ and $\sigma''_{ij} \geq 0$, we have $\sigma'_{ij} = \sigma''_{ij} = 0$. Using similar arguments, $\pi_d^{'L+1} \leq 1$ and $\pi_d^{''L+1} \leq 1$, together with $\pi_d^{L+1} = 1$ imply that $\pi_d^{'L+1} = \pi_d^{''L+1} = 1$. Moreover, for all $l \leq L$, $\pi_d^{'l+1} - \pi_d^{'l} \leq 0$ implies that $\pi_d^{'l} \geq 1$. Similarly, $\pi_d^{''l} \geq 1$, and since $\overline{\pi}_d^l = 1$, we must have $\pi_d^{'l} = \pi_d^{''l} = 1$.

Consider now a node $i \notin W$, i.e. such that $\overline{\pi}_i^l = 1$. Then, $id \notin \delta(W)$ and $\overline{\sigma}_{id} = 0$. Therefore, $\sigma'_{id} = 0$ and $\pi_d^{'l+1} - \pi_i^{'l} - \sigma'_{id} \leq 0$ becomes $\pi_i^{'l} \geq 1$. Similarly, $\pi_i^{''l} \geq 1$, and since $\overline{\pi}_i^l = 1$, we must have $\pi_i^{'l} = \pi_i^{''l} = 1$.

A similar proof, starting with $\overline{\pi}_o^1 = 0$ leads to $\pi_i^{'l} = \pi_i^{''l} = 0$ for all $i \in W$.

Let's turn to an edge $ij \in \delta(W)$, i.e. $\overline{\sigma}_{ij} = \overline{\pi}_j^{l+1} = 1$ and $\overline{\pi}_i^l = 0$ for all $l \leq L - 1$. We already know that $\pi_j^{'l+1} = 1$ and $\pi_i^{'l} = 0$, so from $\pi_i^{'l+1} - \pi_j^{'l} - \sigma'_{ij} \leq 0$ we have $\sigma'_{ij} \geq 1$. Similarly, $\sigma''_{ij} \geq 1$, and since $\overline{\sigma}_{ij} = 1$, we must have $\sigma'_{ij} = \sigma''_{ij} = 1$.

We thus proved that $(\pi', \sigma') = (\pi'', \sigma'') = (\overline{\pi}, \overline{\sigma})$. □

Since $(\overline{\pi}, \overline{\sigma})$ are extreme points of SP($q, \overline{Z}$), *od*-cut constraints are Benders cuts, which implies that the number of Benders cuts is exponential in the size of the network.

Lemma 2. *Let $(V_0, V_1, \ldots, V_{L+r})$ be some partition of V such that $o \in V_0$, $d \in V_{L+r}$ and $V_i \neq \emptyset$ for $i = 1, \ldots L + r - 1$. There exists a vector $(\overline{\pi}, \overline{\sigma})$ feasible for SP($q, \overline{Z}$) so that* (7) *written for $(\overline{\pi}, \overline{\sigma})$ is the same as inequality* (11) *for that partition.*

Proof. First, we must set $\overline{\pi}_d^{L+1} = 1$, $\overline{\sigma}_{ij} = r^{-1} \min(|v - w| - 1, r)$ for $i \in V_v, j \in V_w$, so that (7) becomes

$$r^{-1} \sum_{i \in V_v, j \in V_w, v \neq w} \min(|v - w| - 1, r) Z_{ij} \geq K,$$

equal to (11) by multiplying both sides by r. We are left to set up $\overline{\pi}_i^l$ for $l < L + 1$ so that $(\overline{\pi}, \overline{\sigma})$ is feasible for SP($q, \overline{Z}$). First, set $\overline{\pi}_o^1 = 0$ and $\overline{\pi}_d^l = 1$ for $l = 2, \ldots, L$. For each $0 \leq k \leq L + r$, let $i \in V_v$ and set $\overline{\pi}_i^l = r^{-1} \min(v + 1 - l, r)$ for $l = 1, \ldots, v$, $\overline{\pi}_i^l = 0$ for $l = v + 1, \ldots, L + r - 1$, see Table 1 for $r = 2$ and $L = 4$. We must check that for any $i \in V_v$, $j \in V_w$, and $1 \leq l \leq L$, the arc (i, j, l) satisfies $\overline{\sigma}_{ij} \geq \overline{\pi}_j^{l+1} - \overline{\pi}_i^l$. By definition of $\overline{\pi}$, $\overline{\pi}_j^{l+1} - \overline{\pi}_i^l = \frac{y}{r}$ for some $1 \leq y \leq r$ implies that $w > v + y + 1$ so that $\overline{\sigma}_{ij} \geq \frac{y}{r}$. Thus, $(\overline{\pi}, \overline{\sigma})$ satisfies all equations of SP($q, \overline{Z}$). □

Table 1: Values of $\overline{\pi}$ for the *generalized jump inequality* for $L = 4$ and $r = 2$.

l	1	2	3	4	5
V_0	–/0	0	0	0	–
V_1	–	0	0	0	–
V_2	–	0.5	0	0	–
V_3	–	1	0.5	0	–
V_4	–	1	1	0.5	–
V_5	–	1	1	1	–
V_6	–	1	1	1	–/1

Table 2: Values of $\overline{\pi}$ for nodes in W for the *two-layered 4-path-cut inequality*.

l	1	2	3	4	5
W_1	–	0	0	0	–
W_2	–	1	0	0	–
W_3	–	1	1	0	–
W_4	–	1	1	1	–

Lemma 3. *Let $V_0, V_1, \ldots, V_6, W_1, \ldots, W_4$ be a partition of V such that $o \in V_0$, $d \in V_6$ and $V_i \neq \emptyset$ for $i = 1, \ldots 5$. There exists a vector $(\overline{\pi}, \overline{\sigma})$ feasible for SP($q, \overline{Z}$) so that* (7) *written for $(\overline{\pi}, \overline{\sigma})$ is the same as inequality* (12) *for that partition.*

Proof. We set $\overline{\pi}_d^{L+1} = 1$ and $\overline{\sigma}_{ij} = \frac{1}{2} a_{ij}$, with a defined in (13), and $\overline{\pi}$ as in Tables 1 and 2. The rest of the proof of is similar to the proof of Lemma 2. □

As a result of Lemmas 1, 2 and 3, Dahl (1999), Theorem 2.2 from Huygens et al. (2004) (and its generalization to any K in Dahl et al. (2006) and Diarrassouba (2009)) and Theorem 3 from Huygens and Mahjoub (2007) we obtain:

Proposition 1. *The sets $\mathcal{Z}_c$ and $\mathcal{Z}_i$ are equal for $L = 2, 3$ with any $K \geq 2$, and $L = 4$ with $K = 2$.*

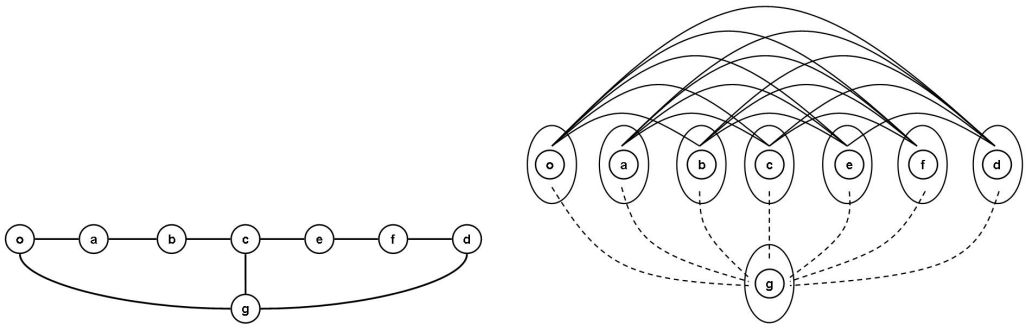
In particular, we obtain that `heuristic` described in subsection 3.3 shall find a feasible solution in this context:

Corollary 1. *For $K = 1$ with any $L \geq 1$, $L = 2, 3$ with any $K \geq 2$, and $L = 4$ with $K = 2$, the algorithm* `heuristic` *will always find a feasible design for (P).*

Proof. Let $(\overline{Z}, \overline{U})$ be an optimal solution to the linear programming relaxation of (P). Thus, $\overline{Z}$ satisfies all Benders cuts (7). Then, since each component of $\overline{\sigma}$ is positive or zero, $\lceil \overline{Z} \rceil$ satisfies all (7) as well, so that $\lceil \overline{Z} \rceil \in \mathcal{Z}_c$. Therefore, Proposition 1 implies that $\lceil \overline{Z} \rceil \in \mathcal{Z}_i$. □

It is natural to wonder whether the equality $\mathcal{Z}_c = \mathcal{Z}_i$ holds for $L = 4$ and $K \geq 3$, and $L \geq 5$ and $K \geq 2$. Although, we do not know the complete answer, Theorem 3.3 from Itaí et al. (1982) leads to the following partial answer.

Proposition 2. *For each $L \geq 4$, there exists a $K \geq 2$ for which the inclusion $\mathcal{Z}_i \subset \mathcal{Z}_c$ holds strictly, unless $\mathcal{P} = \mathcal{NP}$.*



(a) (b)

Figure 4: New inequality for $L = 5$.

Proof. We prove this result by contradiction. Consider some $L \geq 5$ and assume that $\mathcal{Z}_i = \mathcal{Z}_c$ for each $K \geq 2$. Consider some $\overline{Z} \in \mathcal{Z}_i$ and let $G = (V, E)$ be the graph described by $\overline{Z}$, i.e., $ij \in E$ if and only if $\overline{Z}_{ij} = 1$. For each $K \geq 2$, we can check whether there exists K edge-disjoint L-paths between o and d by solving SP($q, \overline{Z}$), because $\mathcal{Z}_i = \mathcal{Z}_c$. Thus, this existence question can be answered in polynomial time for any K. Since the maximum number of such paths is bounded by the number of vertices of V, the problem of finding the maximum number of edge-disjoint L-paths between o and d is polynomial, which contradicts Theorem 3.3 from Itaí et al. (1982) when $L \geq 5$ and Corollary 4 from Bley and Neto (2010) when $L = 4$, unless $P = NP$. □

Finally, let us show that cuts (7) contain new valid inequalities for the problem for $L \geq 5$. First, note that Huygens and Mahjoub (2007) introduce the *two-layered L-path-cut inequalities*, extending the *two-layered 4-path-cut inequalities* to general L. It can be easily seen that Lemma 3 can be extended to incorporate these generalized inequalities. More important is the fact that they show on three examples that these new inequalities, together with (9), (11) and binary restrictions on Z, are not sufficient to formulate the problem for $L \geq 5$. For instance, let G' be the graph shown in Figure 4(a), taken from Huygens and Mahjoub (2007). We see that it is impossible to find two edge-disjoint paths from o to d with length smaller than or equal to 5. Moreover, it is impossible to find a fractional flow satisfying (1),(2) and (5) in the layered graph constructed from G', so that there must exist a Benders cut that cuts off G'. However, Huygens and Mahjoub were not able to provide an inequality that cuts off G'.

We describe next a Benders cut that cuts off the solution depicted in Figure 4(a). Consider the complete graph $G = (V, E)$ with 8 nodes o, a, b, d, e, f, g, d and q a commodity in G from o to d. Define the following dual variables: $\overline{\sigma}_{og} = \overline{\sigma}_{cg} = \overline{\sigma}_{gd} = 0.5$, $\overline{\sigma}_{ij} = 0$ for $ij \in E \backslash \{og, cg, gd\}$, and $\overline{\pi}$ is described in Table 3. One can check that $(\overline{\pi}, \overline{\sigma})$ belongs to SP($q, \overline{Z}$) for G. Moreover, they

Table 3: Values of $\overline{\pi}$ for cut (14).

l	1	2	3	4	5	6
o	0	0	0	0	0	–
a	–	0	1	0	1	–
b	–	1	0	1	0	–
c	–	1	1	0	1	–
e	–	1	1	1	0	–
f	–	1	1	1	1	–
g	–	0.5	0.5	0.5	0.5	–
d	–	1	1	1	1	1

yield the Benders cut

$$ax \geq 2, \tag{14}$$

with $a_{ij} = 1$ for plain edges from Figure 4(b) and $a_{ij} = 0.5$ for dashed edges from Figure 4(b). Applying (14) to G' from Figure 4(a), we obtain $1.5 < 2$. This cut can be extended for more general graphs, partitioning the nodes into 8 subsets as in Figure 4(b) and setting $(\overline{\pi}, \overline{\sigma})$ accordingly. Similar cuts can be obtained in this way for the other examples in Huygens and Mahjoub (2007).

5. Computational results

In this section we compare the solution times of formulations (P) and (P'), which we denote `layered` and `layered-r`, respectively, and branch-and-cut approaches `bc-all`, `bc-int`, `bc-5` and `bc-5-heur`. Then, for the branch-and-cut approaches we compare the number of cuts generated and the number of nodes visited in the branch-and-cut tree. Finally, we evaluate the quality of the upper bound given by `heuristic`.

We have discussed in Section 4 whether we can relax the integrality restrictions (4). We have answered affirmatively for some values of L and K, see Proposition 1. Moreover, we have never encountered an instance for which a feasibility cut (8) was needed. Thus, Benders cuts were enough to describe the problem for all the instances in our computational experiments, so that models `layered` and `layered-r` coincide for these instances. Therefore, we also present the computational time required by `layered-r`. Note that we tested branch-and-cut algorithms without the feasibility part as well, but the speed-up was insignificant, so that we do not report them in the remainder.

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. 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. Moreover, for each algorithm we configure CPLEX as follows :

`layered` and `layered-r`: All parameters have been kept to their default values, CPLEX chooses to explore the branch-and-cut tree with the dynamic search.

`bc-all`, `bc-int`, and `bc-n`: Since the model does not contain explicitly all constraints, we must deactivate the dual presolve, setting *BooleanParam.PreLinear* to false and *IntParam.Reduce* to 1. Then, we implemented our (global) cuts generation with a *LazyConstraintCallback*, preventing CPLEX from using dynamic search.

`heuristic`: We first solve the linear programming relaxation by a cutting plane algorithm (in fact, we use a branch-and-cut algorithm with a limit of 0 nodes, setting *IntParam.NodeLim* to 0). Then, we fix some of the variables to 0, and re-solve the resulting problem with `bc-n`.

`bc-n-heur`: We use the algorithm `bc-n`, providing CPLEX with the upper bound found by `heuristic`. The CPU times reported do not consider the time spent in `heuristic`.

5.2 Instance details

We used three different test sets. The sets TC and TE were taken from a class of complete graphs $G = (V, E)$, reported in Gouveia (1996). They share the following features: $|V| = 21$, $|Q| \in \{5, 10\}$, and all point-to-point demands share one of their extremities (which we call rooted demands in Table 4). The cost matrix for each instance considers the integer part of the Euclidean distance between the coordinates of the 21 nodes, randomly placed among the integer points of a grid 100×100. The TC class contains 5 instances with 5 commodities and 5 instances with 10 commodities with the root located in the center of the grid and the TE class contains 5 instances with 5 commodities and 5 instances with 10 commodities with the root located on a corner of the grid. We see in the next section that instances TE are much harder to solve than instances TC. Then, two instances are based on sparse networks from *SNDlib* (Orlowski et al., 2010): pdh and di-yuan. Table 4 summarizes the size of the instances. We solved the problem for L in $\{3, 4, 5, 7, 10\}$ and K from 1 to 3. We set a time limit of 3600 seconds for all instances and algorithms.

Table 4: Instances description.

Name	$\|N\|$	$\|E\|$	$\|Q\|$	# of instances	Rooted demands?
TC-5	21	210	5	5	true
TC-10	21	210	10	5	true
TE-5	21	210	5	5	true
TE-10	21	210	10	5	true
pdh	11	34	24	1	false
di-yuan	11	42	22	1	false

We also tested the performance of our best branch-and-cut algorithm `bc-5-heur` on a set of larger instances described below. Those instances (see Table 5) are made of complete graphs composed of 41 vertices with a single source and many (5,10 or 20) destinations. The distinction between TC and TE instances is the same as before. This time, we test only the methods `layered`, `layered-r` and `bc-5-heur`, and fix a maximum CPU time limit of 3 hours. Because the whole execution of `heuristic` was taking too much time, we only run the heuristic for 5 minutes. Then, we start the branch-and-cut algorithm `bc-5-heur` with the best bound provided by `heuristic` during the 5 minutes.

Table 5: Large instances description.

Name	$\|N\|$	$\|E\|$	$\|Q\|$	# of instances	Rooted demands?
TC_h-5	41	820	5	1	true
TC_h-10	41	820	10	1	true
TC_h-20	41	820	20	1	true
TE_h-5	41	820	5	1	true
TE_h-10	41	820	10	1	true
TE_h-20	41	820	20	1	true

5.3 Results for medium size instances

First, we look at the quality of the linear programming relaxation of our model (P). Let IP^* and LP^* be the optimal value of (P) and its linear programming relaxation, respectively. Table 6 shows that the linear programming bound of the model improves when the value of K increases and that it is quite bad for $K = 1$. Apparently these results are not in agreement with the results provided in other papers (eg., Gouveia (1998)) which in a certain way have motivated the choice of model

Table 6: Arithmetic average of gap $\left(\frac{IP^*-LR^*}{IP^*} * 100\right)$ for TC and TE instances.

		K			
$\|Q\|$	L	1	2	3	Average
5	3	18.07	4.79	1.98	8.28
	4	20.78	5.87	4.87	10.50
	5	22.58	0.00	5.28	9.29
	7	22.60	0.00	5.51	9.37
	10	22.60	0.00	5.51	9.37
Average $\|Q\|$=5		21.33	2.13	4.63	9.36
10	3	25.58	14.67	8.02	16.09
	4	27.58	11.65	6.62	15.28
	5	28.82	9.65	5.49	14.66
	7	30.33	5.24	6.15	13.91
	10	32.32	0.00	6.59	12.97
Average $\|Q\|$=10		28.93	8.24	6.57	14.58
Average Total		25.13	5.19	5.60	11.97

for this work. The reason is that the problem studied in Gouveia is a Steiner tree problem, which permitted a different enhancement technique, namely the use of the technique "of directing the formulation". This leads to much stronger linking constraints between flow variables and design variables. We note that without explicitly stating that our problem is a tree problem for $K = 1$, it is not difficult to see that the optimal solutions are Steiner trees spanning the root node and the nodes in Q and thus we could have used the "directing the formulation" technique. However, the focus of our paper is on the disjoint case and thus, we have not considered this enhancement technique for $K = 1$.

Before comparing the different algorithms, we need to determine the "best" value for the depth parameter of branch-and-cut algorithm `bc-n`. We select a group of complicated instances (instances that `layered` cannot solve to optimality within 3600 seconds) and we test different values of the depth parameter n. On Figure 5, we plot the result of this tuning stage. For both curves, the minimum is reached when $n = 5$. Therefore, in the sequel we always consider `bc-5` for the hybrid branch-and-cut algorithm.

We compare the performance in terms of resolution time for the different methods by plotting the performance profile (Dolan and More, 2002) on Figure 6. Clearly, algorithms `bc-5`, `bc-int` and `bc-5-heur` are the fastest algorithms.

Out of the 330 instances which compose the entire test set, `layered`, `layered-r`, `bc-all`,

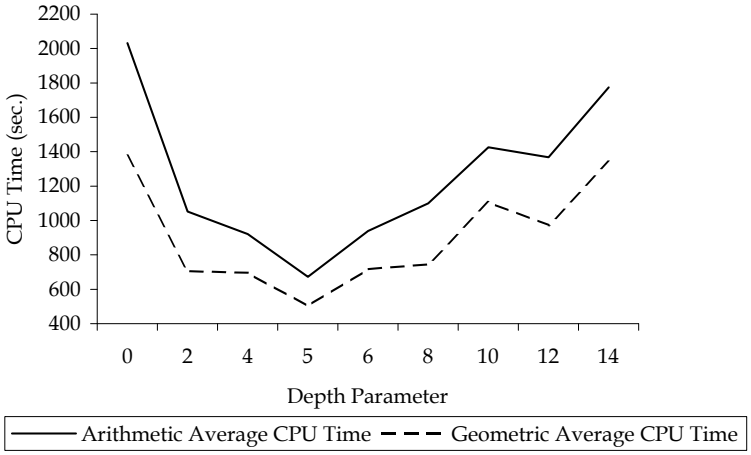


Figure 5: `bc-n` depth parameter tuning by average CPU time (sec.)

and `bc-int` could respectively not solve 21, 20, 39, and 2 of them within 3600 seconds. In contrast, `bc-5` and `bc-5-heur` could solve them all. Among the 21 instances that `layered` and 20 instances that `layered-r` cannot solve to optimality, only 2 cannot be solved by `bc-int`. `bc-5` and `bc-5-heur` can solve all instances to optimality. The geometric averages of CPU times in seconds are > 31.62, > 25.97, > 36.80, > 9.46, 9.61, 7.58, and 2.45, for `layered`, `layered-r`, `bc-all`, `bc-int`, `bc-5`, `bc-5-heur`, and `heuristic` respectively ($>$ indicates that one or more instances could not be solved to optimality).

Table 7 indicates the arithmetic average values of the LP relaxation and `heuristic` gaps, given by $\frac{IP^*-LP^*}{IP^*}$ and $\frac{heuristic^*-IP^*}{IP^*}$, respectively, as well as `heuristic` CPU time in seconds and the number of instances for which the solution of `heuristic` is optimal. It can be seen that `heuristic` always provides a very good solution to the problem. Furthermore, `heuristic` is also pretty fast, taking around 3 seconds whereas `layered` and `bc-5-heur` take respectively on average 31.62 and 7.58 seconds. In 174 cases out of 330 (around 53%), the solution given by the heuristic is the optimal one. Finally, Tables 8 and 9 respectively present geometric averages of the number of Benders cuts generated by the branch-and-cut algorithms, and number of nodes explored by branch-and-cut and extended formulations, and Table 10 provides means of CPU time and the corresponding percentage of total time spent for solving Benders subproblems (means have been taken over all instances).

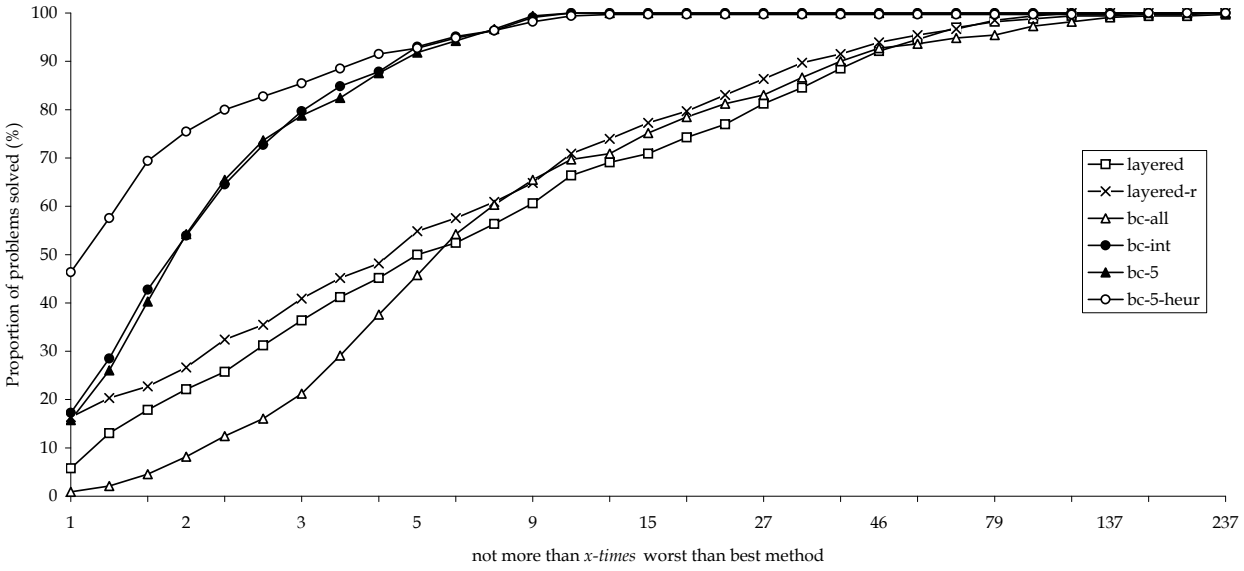


Figure 6: Performance profile comparing methods on the entire test set.

Table 7: Linear relaxation gap and `heuristic` performance for the entire test set.

Instances	LP Relaxation	`heuristic`	
	Gap(%)	Gap(%)	Optimal
TC-5	8.07	2.45	38/75
TC-10	12.67	1.85	42/75
TE-5	10.64	3.28	33/75
TE-10	16.48	1.23	38/75
pdh	18.82	0.62	10/15
di-yuan	14.46	0.86	13/15
Arithmetic mean	12.39	2.07	-

5.4 Results for large instances

Out of the 54 instances that belong to the test sets of larger instances, 25, 25, and 22 instances could not be solved to optimality within the time limit by `layered`, `layered-r`, and `bc-5-heur`, respectively. The difference in performance between the methods is less striking than with the medium-size instances. Indeed `layered` and `layered-r` do not reach optimality for 25 instances which is close to the 22 instances that `bc-5-heur` cannot solve optimality.

Table 11 shows the number of large instances that could be solved to optimality by all three methods, depending on $|Q|$, L and K. Out of the 54 instances tested, only 27 instances could be solved to optimality by the three methods. The geometric average of their solution times (in

Table 8: Number of cuts generated for the entire test set.

Instances	Number of cuts generated			
	bc-all	bc-int	bc-5	bc-5-heur
TC-5	133.45	85.51	89.35	67.63
TC-10	1950.49	391.05	470.82	229.19
TE-5	277.87	146.59	148.22	114.42
TE-10	5420.83	809.77	943.42	516.66
pdh	1328.24	491.61	501.22	458.86
di-yuan	358.36	203.88	195.24	185.26
Geometric mean	781.49	256.36	280.03	182.34

Table 9: Number of nodes visited for the entire test set.

Instances	Number of nodes visited				
	bc-all	bc-int	bc-5	bc-5-heur	layered
TC-5	25.27	50.23	32.84	16.22	16.98
TC-10	354.39	1863.52	967.46	600.49	471.27
TE-5	47.81	147.42	81.59	103.11	60.25
TE-10	675.94	8832.68	3690.13	3900.22	1278.58
pdh	764.30	1621.85	1325.91	559.40	1299.96
di-yuan	64.43	140.82	114.56	31.33	107.15
Geometric mean	136.89	579.57	319.05	236.12	170.43

Table 10: CPU time spent for solving Benders subproblems.

	bc-all	bc-int	bc-5	bc-5-heur	heuristic
CPU time (Geometric mean)	19.09	4.01	4.73	0.16	0.07
Percentage of total time (Arithmetic mean)	65.54 %	62.68 %	64.83 %	23.17 %	21.42 %

seconds) are 69.72, 62.19, 140.84, and 15.09, for layered, layered-r, bc-5-heur, and heuristic respectively. Clearly for those instances, the two layered models layered-r and layered are faster than our branch-and-cut algorithm bc-5-heur. As we did before, for the same 27 instances we evaluate the quality of the linear programming relaxation of our model (P) by computing the gap at the root. Table 12 shows the evolution of this gap depending on $|Q|$, K and L. The conclusion is similar to the one we reached before since the gap is relatively large

Table 11: Number of large instances solved to optimality by all methods in the 10800 seconds time windows.

		K		
$\|Q\|$	L	1	2	3
5	3	2/2	2/2	2/2
	4	2/2	2/2	2/2
	5	2/2	2/2	1/2
10	3	2/2	2/2	2/2
	4	1/2	1/2	2/2
	5	0/2	0/2	0/2
20	3	0/2	0/2	0/2
	4	0/2	0/2	0/2
	5	0/2	0/2	0/2
Total		9/18	9/18	9/18

Table 12: Arithmetic average of gap (%) $\left(\frac{IP^*-LR^*}{IP^*} * 100\right)$ for the 27 large instances TC and TE solved to optimality.

		K			
$\|Q\|$	L	1	2	3	Average
5	3	17.31	9.82	4.13	10.42
	4	18.13	5.31	4.74	9.39
	5	18.24	0.00	3.55	8.00
Average $\|Q\|$=5		17.89	5.04	4.26	9.35
10	3	25.94	13.06	8.32	15.77
	4	25.62	8.05	4.79	10.81
Average $\|Q\|$=10		25.84	11.39	6.55	13.79
Average Total		20.54	7.16	5.28	10.99

for $K = 1$ and becomes smaller for $K = 2, 3$. The time spent by `heuristic` to find a good upper bound is again relatively low (on average 15.09 seconds). Moreover the quality of this upper bound is good since for 13 out of 27 instances (around 48% of the cases), the solution given by the `heuristic` is the optimal solution and on average the gap between the heuristic and the optimal solution is around 3.19%.

Table 13 shows the gap remaining to close after 3 hours of computation for the 27 instances that at least one of the methods could not solve to optimality within the time limit. This gap is given by $\frac{UB-LB}{UB}$, where UB is the objective value of the best feasible solution found, and LB the best guaranteed lower bound after 3 hours. For $|Q| \in \{5, 10\}$ we see, on average, that the gap remaining

Table 13: Arithmetic average of gap (%) remaining to close for the 27 large instances TC and TE not solved to optimality.

		layered	layered-r	bc-5-heur	Average
$\|Q\|$	5	0.00	1.48	0.00	0.49
	10	1.09	10.07	3.24	8.13
	20	25.91	17.66	19.01	20.86
Average Total		20.56	14.81	13.63	

to close for the model layered is the smallest one even if the gap proposed by the branch-and-cut algorithm is not far. For $|Q| = 20$, the situation is different since the gap of layered-r is the smallest (17.66% compared to 25.91% and 19.01 respectively for the models layered and bc-5-heur). Nevertheless our branch-and-cut algorithm bc-5-heur seems to be the best method from a global point of view since its global gap is the smallest one. This conclusion is reinforced by the following observation: bc-5-heur leads to the smallest gap remaining to close in 74.07% of the 27 large instances not solved to optimality. This figure goes down to 11.11% and 25.93% for layered and layered-r, respectively.

From this numerical experiment it seems that the difficulty of a given instance is not only linked to the number of commodities $|Q|$. Indeed in the *SNDlib* instances, the number of commodities is high but the instances are easy to solve because the graphs from *SNDlib* are sparse. In contrast TC and TE instances are difficult instances to solve because there are complete graphs and this kind of instances become much more difficult to solve as the number of commodities increases.

Detailed results for each instance can be found at http://bit.ly/k85Khm (pdf) or http://bit.ly/eSWTsE (xls) Moreover, all our instances can be downloaded in text format at http://bit.ly/iA8pqu.

Acknowledgments

The authors would like to thank the Associate Editor and the three referees for their comments and suggestions that help in improving the presentation of this paper. 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 Mathémathique". Michael Poss is a research fellow of the "Fonds pour la Formation à la Recherche dans l'Industrie et dans l'Agriculture" (FRIA). 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-LVT-Lisboa-152.

References

Achterberg, T. 2009. SCIP: Solving constraint integer programs. *Math. Programming Comput.* **1** 1–41.

Achterberg, T., C. Raack. 2010. The MCF-separator – detecting and exploiting multi-commodity flows in MIPs. *Mathematical Programming C* 125–165.

Bai, L., P. A. Rubin. 2009. Combinatorial Benders cuts for the minimum toll booth problem. *Oper. Res.* **57** 1510–1522.

Balakrishnan, A., K. Altinkemer. 1992. Using a hop-constrained model to generate alternative communication network design. *ORSA J. Comput.* **4** 192–205.

Benders, J. 1962. Partitioning procedures for solving mixed-variables programming problems. *Numerische Mathematik* **4** 238–252.

Birge, J. R., F. V. Louveaux. 2008. *Introduction to Stochastic programming (2nd edition)*. Springer Verlag, New-York.

Bley, A. 1997. Node-disjoint lenght-restricted paths. Master's thesis, TU Berlin.

Bley, A., J. Neto. 2010. Approximability of 3- and 4-hop bounded disjoint paths problems. *Proceedings of IPCO 2010, Lausanne, Switzerland*. No. 6080 in LNCS, Springer, 205–218.

Carøe, C. C., J. Tind. 1998. L-shaped decomposition of two-stage stochastic programs with integer recourse. *Math. Programming* **83** 451–464.

Codato, G., M. Fischetti. 2006. Combinatorial Benders' cuts for mixed-integer linear programming. *Oper. Res.* **54** 756–766.

Costa, A. M. 2005. A survey on Benders decomposition applied to fixed-charge network design problems. *Comput. Oper. Res.* **32** 1429–1450.

Dahl, G. 1999. Notes on polyhedra associated with hop-constrained walk polytopes. *Oper. Res. Lett.* **25** 97–100.

Dahl, G., N. Foldnes, L. Gouveia. 2004. A note on hop-constrained walk polytopes. *Oper. Res. Lett.* **32** 345–349.

Dahl, G., L. Gouveia. 2004. On the directed hop-constrained shortest path problem. *Oper. Res. Lett.* **32** 15–22.

Dahl, G., D. Huygens, A. R. Mahjoub, P. Pesneau. 2006. On the k edge-disjoint 2-hop-constrained paths polytope. *Oper. Res. Lett.* **34** 577–582.

Dahl, G., B. Johannessen. 2004. The 2-path network problem. *Networks* **43** 190–199.

Dahl, G., M. Stoer. 1998. A cutting plane algorithm for multicommodity survivable network design problems. *INFORMS J. Comput.* **10** 1–11.

Diarrassouba, I. 2009. Survivable network design problems with high survivability requirements. Ph.D. thesis, Université Blaise Pascal - Clermond-Ferrand II.

Dolan, E. D., J. J. More. 2002. Benchmarking optimization software with performance profiles. *Math. Programming* **91** 201–213.

Fischetti, M., D. Salvagnin, A. Zanette. 2010. A note on the selection of Benders cuts. *Math. Program.* **124** 175–182.

Fortz, B. 2000. *Design of Survivable Networks with Bounded Rings*, *Network Theory and Applications*, vol. 2. Kluwer Academic Publishers.

Fortz, B., M. Labbé. 2002. Polyhedral results for two-connected networks with bounded rings. *Math. Programming* **93** 27–54.

Fortz, B., M. Labbé. 2004. Two-connected networks with rings of bounded cardinality. *Comput. Optim. and Appl.* **27** 123–148.

Fortz, B., A.R. Mahjoub, S.T. Mc Cormick, P. Pesneau. 2006. Two-edge connected subgraphs with bounded rings: Polyhedral results and branch-and-cut. *Math. Prog.* **105** 85–111.

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

Gouveia, L. 1996. Multicommodity flow models for spanning trees with hop constraints. *Eur. J. Oper. Res.* **95** 178–190.

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

Gouveia, L., T. L. Magnanti. 2003. Network flow models for designing diameter-constrained minimum-spanning and Steiner trees. *Networks* **41** 159–173.

Gouveia, L., P.F. Patrício, A.F. Sousa. 2006. *Telecommunications Planning: Innovations in Pricing, Network Design and Management*, chap. Compact models for hop-constrained node survivable network design. Springer, New York, 167–180.

Gouveia, L., P.F. Patrício, A.F. Sousa. 2008. Hop-contrained node survivable network design: An application to MPLS over WDM. *Networks Spatial Econom.* **8** 3–21.

Gouveia, L. E., P.F. Patrício, A.F. de Sousa, R. Valadas. 2003. MPLS over WDM Network Design with Packet Level QoS Constraints based on ILP Models. *Proceedings of IEEE Infocom*. 576–586.

Grötschel, M., C. L. Monma, M. Stoer. 1992. Facets for polyhedra arising in the design of communication networks with low-connectivity constraints. *SIAM J. Optim.* **2** 274–504.

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

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

Huygens, D., A. R. Mahjoub. 2007. Integer programming formulations for the two 4-hop-constrained paths problem. *Networks* **49** 135–144.

Huygens, D., A. R. Mahjoub, P. Pesneau. 2004. Two edge-disjoint hop-constrained paths and polyhedra. *SIAM J. Discrete Math.* **18** 287–312.

IBM-ILOG. 2009. IBM-ILOG Cplex. Http://www.ilog.com/products/cplex/.

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

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

Ljubic, I., P. Putz, J.J. Salazar. 2009. Exact approaches to the single-source network loading problem. Tech. Rep. 2009-05, University of Vienna.

Magnanti, T.L., S. Raghavan. 2005. Strong formulations for network design problems with connectivity requirements. *Networks* **45** 61–79.

Nemhauser, G.L., L.A. Wolsey. 1988. *Integer and combinatorial optimization*. Wiley-Interscience series in discrete mathematics and optimization, Wiley.

Orlowski, S., M. Pióro, A. Tomaszewski, R. Wessäly. 2010. SNDlib 1.0–Survivable Network Design Library. *Networks* **55** 276–286.

Pirkul, H., S. Soni. 2003. New formulations and solution procedures for the hop constrained network design problem. *Eur. J. Oper. Res.* **148** 126–140.

Raghavan, S. 1995. Strong formuations for network design problems with connectivity requirements. Ph.D. thesis, Massachusetts Institute of Technology.

Stoer, M. 1993. *Design of survivable networks*. Springer.