Benders decomposition for the hop-constrainted survivable network design problem

Quentin Botton
Center for Supply Chain Managemqent (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, and CORE, Université catholique de Louvain, Louvain-la-Neuve, 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 a set of pairs of nodes 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, we present the first 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. 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 cutting plane and 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 11. Then, 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 $L \geq 5$.

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

1. Introduction

Our society is increasingly dependent on large-scale, networked information systems of remarkable scope and complexity. Incorporating survivability capabilities into a network has become unavoidable for network operators in order to mitigate the risks in case of failures. Herein, survivability is the capability of a system to fulfil its mission in a timely manner despite intrusions, failures, or accidents. This concept of survivability allows networks to remain functional when links or nodes fail. 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 of the two paths, this particular edge can not be used by the other path. 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 aspect considers the node-disjointness 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_e to each edge e and introduce an auxiliary arc set A which is obtained from each edge i,j 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 vertice of demand) and d (the destination vertice 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 $K-ESNDP$, consists in finding an 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. A polyhedral study of the problem for $K=2$ is can be found in Stoer (1993). Similarly, the node-survivable network-design problem, denoted by $NSNDP$, consists in finding a minimum-cost node-survivable subgraph of G, see M. Grötschel and Stoer (1992) among others. A variant of this problem only requires edge (or node) disjointness between pairs of node in a subset $S \subset V$. Note that a related network design problem with node-disjoint paths between some pairs of nodes which also incorporates hop-constraints in a disguised fashion is the bounded ring network design problem studied by Fortz and Labbé (2002, 2004); Fortz et al. (2006), among others. For more references about these problems, see the survey by Kerivin and Mahjoub (2005).

In general, the survivability constraints are not sufficient to guarantee a cost effective routing with a good quality of service since the primary path where the traffic flows or the secondary paths, if a path fails, may lead to unacceptable delays. Since in most of the routing technologies, delay is caused at the switching nodes because node processing times dominate over queuing delays, 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, so that the traffic may be routed, or rerouted if a path fails, on a different path with a quality of service guaranteed in terms of delay. These, so called hop constraints also guarantee a certain level of transmission reliability in the sense that the probability that all the transmission lines in the path are working decrease with the number of arcs (Woolston and Albin, 1988). We say that a L-path is a path with at most L arcs (or hops).

The L-hop constrained network design problem, consists in finding a least cost subgraph of G such that there exists a L-path between every pair of nodes, or the nodes in a subset S. This problem was studied by Balakrishnan and Altinkemer (1992) as a means of generating alternative base solutions for a network design problem. They presented a standard network flow formulation with an additional cardinality constraint for each commodity (to model the hop constraints). They have also derived a Lagrangean relaxation based method whose theoretical best bound improves the linear programming bound given by the previous formulation. Later on, in the context of a directed spanning tree problem (which can be seen as a special case of a single-source L-hop-constrained network design problem), Gouveia (1998) presented a layered network flow reformulation whose linear programming bound equals a lagrangean based bound similar to the one proposed by Balakrishnan and Altinkemer. Then this reformulation has 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 interesting to point out that the case with $L = 2$ already contains lots of structure. Dahl and Johannessen (2004) make a computational study of this variation of the problem.

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 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 for this is that it is not straightforward to obtain a valid natural formulation for a variant of the single commodity case. This variant consists in 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.

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 quite 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, nothing else is known which explains that the only cutting plane method for the more general problem with several sources and several destinations, Huygens et al. (2007) only considers $L \leq 3$. Based on this, it makes sense to look for alternative ways of formulating the problem, and 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. In fact, 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) should be reasonably good;

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

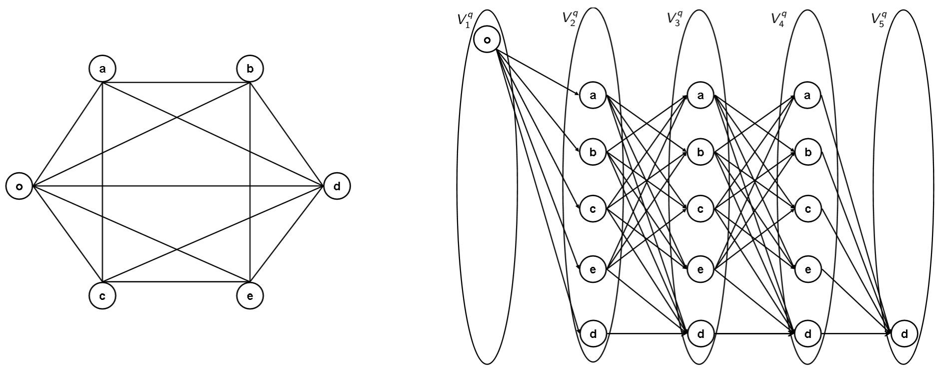
Thus, (i) and (ii) motivate the approach of using this kind of model but using a decomposition algorithm, such as the Benders decomposition. Furthermore, another outcome of this research is that the Benders cuts might give some relevant information for finding valid inequalities for the cases with $L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

In this paper, we formulate the network design problem for any $L, K \geq 1$ with multiple pairs of terminals $(o(q), d(q))$, for $q \in Q$, as an integer program based on the layered representation from Gouveia (1998). Up to our knowledge, this is the first formulation for the problem valid for $L \geq 5$. As previously mentioned the model is too large to be used directly within CPLEX. 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 aspect, most authors using "textbook implementations". Some recent works (Fischetti et al., 2008; 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. We present a thorough computational study of various cutting plane and branch-and-cut algorithms on a large set of instances including the real based instances from SNDlib (Orlowski et al., 2007). We confirm previous results obtained by Fortz and Poss (2009) showing that branch-and-cut algorithms outperform cutting plane algorithms. We also show that our Benders cuts contain the constraints used by Huygens et al. (2004) and Huygens and Mahjoub (2007) to formulate the problem for $L = 2, 3, 4$ 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 separate at the same time (and polynomially) "cuts inequalities" and "L-paths inequalities" while the branch-and-cut algorithm from Huygens et al. (2007) needs to separate both 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 formulation. In Section 3, we reformulate the problem through Benders decomposition and discuss different algorithmical 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 $A^q = \{(i_l^q, j_{l+1}^q) \mid i_l^q \in V_l^q, j_{l+1}^q \in V_{l+1}^q, l \in \{1, \ldots, L\}\}$, see



(a) (b)

Figure 1: Basic Network (a) and its Layered Representation (b) when $L = 4$

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 layered graph G^q is composed of exactly L 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:

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

Each variable Z_{ij} states whether edge $ij \in E$ is selected 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 $U_{ii}^{lq} \leq K$.

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 additional constraints to the problem. Benders decomposition has been widely studied for fixed charge network design problems, (Costa, 2005). Indeed, these problems usually route multi-commodity flows on some network to be designed. Therefore, the associated formulations contain many constraints and variables bound together by the capacity constraints. Then, once we project out the flow variables, the subproblems become 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 (Costa (2005) among others) to project out U

variables from (P'), resulting in the Benders reformulation

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

where $\mathcal{R}^q$ contains vertices of the feasibility polyhedron for the dual subproblem $SP(q, \overline{Z})$ described next. Given commodity $q \in Q$, let us introduce a dual variable π^l_i, 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 constraint $\pi^{L+1}_d \leq 1$ to normalize the dual cone (see Fischetti et al. (2008); Ljubic et al. (2009) for alternative choices of normalization constraints), we get our dual subproblem $SP(q, \overline{Z})$

$$SP(q,\overline{Z}) \qquad \begin{array}{lll} \max & K\pi^{L+1}_d - \sum_{ij \in E} \overline{Z}_{ij}\sigma_{ij} & \\ \text{s.t.} & \pi^2_i - \pi^1_o - \sigma_{oi} \leq 0, & i \in V\backslash\{o\}, \\ & \pi^{l+1}_i - \pi^l_j - \sigma_{ij} \leq 0, & i,j \in V\backslash\{o\}, i \neq j, l \in \{2,\dots,L\}, \\ & \pi^{l+1}_j - \pi^l_i - \sigma_{ij} \leq 0, & i,j \in V\backslash\{o\}, i \neq j, l \in \{2,\dots,L\}, \\ & \pi^{L+1}_d - \pi^L_i - \sigma_{id} \leq 0, & i \in V, \\ & \pi^{l+1}_d - \pi^l_d \leq 0, & l \in \{2,\dots,L\}, \\ & \pi^{L+1}_d \leq 1, & \\ & \sigma_{ij} \geq 0, & ij \in E. \end{array} \tag{6}$$

Note that for each commodity $q \in Q$, one of the constraints in (1) is redundant, which can be represented by setting $\pi^1_o = 0$.

Next, we discuss how to extend this procedure to (P). We need first to introduce some notations. 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

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

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 $K = 2$. However, since we do not know the answer for general K and L, we must in general solve the following feasibility problem

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

If $e = 0$, $\overline{Z} \in \mathcal{Z}_i$. Otherwise, we must add the weak inequality

$$
\sum_{ij\in E^0(\overline{Z})} Z_{ij} \geq 1, \tag{8}
$$

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. It is interesting to point out that in our computational experiments, we never needed to add such a cut, see Section 5. As a consequence, we did not find a vector $\overline{Z} \in \mathcal{Z}_c \backslash \mathcal{Z}_i$.

3.2 Algorithmic approach

(BR) contains an exponential number of constraints while only a few of them are active at the optimum. Therefore, we must dynamically generate the required constraints throughout the solution method. First works 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 (ILOG, 2007) 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. In subsection 3.2.1 we briefly describe the multi-cut cutting plane algorithm, while we detail our different branch-and-cut algorithms in subsection 3.2.2.

3.2.1 Cutting plane approach

Algorithm 1: "Naive" Benders decomposition algorithm: `cp`

```
repeat
    solve $(MP)$;                                        /* solve an IP */
    let $\overline{Z}$ be an optimal solution;
    foreach $q \in Q$ do
        compute $s_q = SP(q, \overline{Z})$;             /* solve the dual subproblem */
        if $s_q > 0$ then add (7) to $(MP)$;
    if $s_q \leq 0$ for each $q \in Q$ then    /* are all dual suproblems feasible? */
        foreach $q \in Q$ do
            compute $f_q = FP(q, \overline{Z})$;         /* solve the feasibility subproblem */
        if $f_q > 0$ for some $q \in Q$ then add (8) to $(MP)$;
until $s_q, f_q \leq 0$ for each $q \in Q$ ;
return $\overline{Z}$
```

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

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

In Algorithms 1 and 2, we describe our cutting plane algorithms, `cp` and `cp-i`. Because the main computational burden is the solution of (MP), we implemented a multi-cut version of the algorithm: we solve the subproblem for each commodity, therefore adding up to $|Q|$ cuts per iteration. The improved version `cp-i` starts by solving the linear programming relaxation of (MP) in a cutting plane fashion. Various works enhance this classical solution algorithm. Among them, Magnanti and Wong (1981) study the effect of using special cuts

Algorithm 2: Improved Benders decomposition algorithm: `cp-i`

```
repeat
    solve the linear programming relaxation of $(MP)$;                /* solve a LP */
    let $\overline{Z}$ be an optimal solution;
    foreach $q \in Q$ do
        compute $s_q = SP(q, \overline{Z})$;
        if $s_q > 0$ then add (7) to $(MP)$;
until $s_q \leq 0$ for each $q \in Q$ ;
repeat
    solve $(MP)$;                                                     /* solve an IP */
    let $\overline{Z}$ be an optimal solution;
    foreach $q \in Q$ do
        compute $s_q = SP(q, \overline{Z})$;
        if $s_q > 0$ then  add (7) to $(MP)$;
    if $s_q \leq 0$ 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 $(MP)$;
until $s_q, f_q \leq 0$ for each $q \in Q$ ;
return $\overline{Z}$
```

called “pareto optimal”, Tsamasphyrou et al. (2000) describe a more subtile version of the multi-cut algorithm, grouping together subsets of subproblems, and Rei et al. (2009) use local branching to accelerate the overall algorithm. Sometimes, strong classes of cuts are known for the problem allowing to add even more cuts at each iteration (Gabrel et al., August 1999). Nevertheless we show herein that, at least for our problem, branch-and-cut algorithms are order of magnitude faster than cutting plane algorithms.

3.2.2 Branch-and-cut approach

An alternative strategy is to solve (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 check for cuts (7) and (8) only at integer nodes. Finally, we developed a hybrid algorithm bc-n, described in Algorithm 3, 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 3, solving a node $o' \in T$ 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 3: Hybrid branch-and-cut algorithm: bc-n

```
begin                                                        /* Initialization */
    $T = \{o\}$ where $o$ has no branching constraints;
    $UB = +\infty$;
end
while $T$ is nonempty do
    select a node $o' \in T$;
    $T \leftarrow T \setminus \{o'\}$;                     /* 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
            $T \leftarrow T \cup \{o'\}$;                    /* put node $o'$ back in the tree */
        else if $\overline{Z} \notin \{0,1\}^{|E|}$ then
            branch, resulting in nodes $o^*$ and $o^{**}$;
            $T \leftarrow T \cup \{o^*, o^{**}\}$;              /* add children to the tree */
return $Z^*$
```

3.3 Heuristic

An intrinsic difficulty of Benders decomposition is that we replace the well-structured problem (P) by a problem (MP) with no straightforward structure. Indeed, it is well known that special structures can help the solution of hard integer programs. For instance, detecting a a flow structure within a more complicated problem can be used to add strong cut inequalities (Achterberg and Raack, 2009). Moreover, for many problems Benders cuts have fractional coefficients, yielding numerical instability (Codato and Fischetti (2006) avoid this difficulty for combinatorial problems with certain classes of "big M" constraints). Finally, it will be hard for our default MIP solver (CPLEX 11 in our case) to find good upper bounds. We present next a simple, yet efficient, heuristic. First, we solve the linear programming relaxation of the Benders decomposition, 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 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 $\mathcal{Z}_i = \mathcal{Z}_c$. Stated simply, are the set of feasible network designs equal for (P) and (P') ? This is equivalent to knowing whether $\mathcal{U}_i(\overline{Z}) = \emptyset$ implies that $\mathcal{U}_c(\overline{Z}) = \emptyset$, for any binary vector $\overline{Z}$. Notice 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 on 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 $\{P3, o - b - c - d\}$, 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

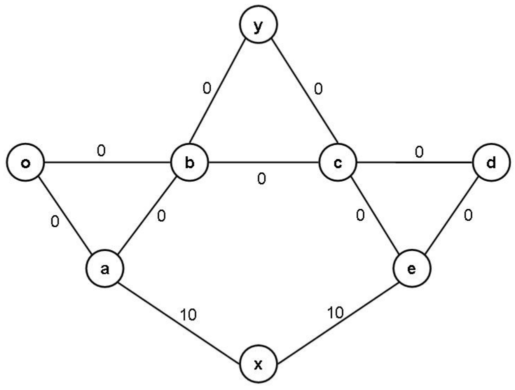


Figure 2: Graph obtained from $\overline{Z}$

10. Thus, the cheapest fractional routing is less than the cheapest integer routing, implying $conv(\mathcal{U}_i(\overline{Z})) \subset \mathcal{U}_c(\overline{Z})$.

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 the sequel, we show that cuts (7) generalize these families of cuts. Therefore, using results from Dahl et al. (2006); Diarrassouba (2009) and 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 $L \geq 5$.

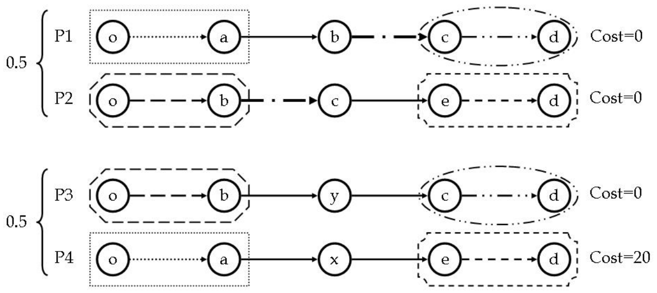
We introduce first some notations. If $W \subset V$ is a node subset, then the set of edges that have one node in W and one node in $V \backslash W$ is called a cut and denoted by $\delta(W)$, and $Z(\delta(W)) := \sum_{ij \in \delta(W)} Z_{ij}$. For $o, d \in V$, a cut $\delta(W)$ such that $o \in W$ and $d \in V \backslash W$ is called a *od-cut*. Then, let $V_0, V_1, \ldots, V_{L+1}$ 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 $|v - w| > 1$.

For any L and $K = 1$, Dahl (1999) proves that a binary vector $\overline{Z}$ belongs to $\mathcal{Z}_i$ if it satisfies the following inequalities:

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

and the $L-path-cut$ inequalities

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



Cheapest fractional routing = 10
Cheapest integer routing = 20

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

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 works Dahl et al. (2006); Diarrassouba (2009), the authors 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 $s \in V_0$, $t \in V_{L+r}$ 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

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

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) generalize cuts (9), (11) and (12) (thus (10) because it is a special case of (11)).

Lemma 1. *Consider some $od-cut$ $\delta(W)$. Inequality* (9) *for $\delta(W)$ is a Benders cut* (7) *for some vector $(\overline{\pi}, \overline{\sigma})$ feasible for $SP(q, \overline{Z})$.*

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

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

We are left to set up $\overline{\pi}_i^l$ for $l < L+1$ so that $(\overline{\pi}, \overline{\sigma})$ is feasible for $SP(q, \overline{Z})$. 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. □

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 $i = 1, \ldots L+r-1$. Inequality* (11) *for this partition is a Benders cut* (7) *for some vector $(\overline{\pi}, \overline{\sigma})$ feasible for $SP(q, \overline{Z})$.*

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 $0 \leq v \leq L+r$, let $i \in V_k$ and set $\overline{\pi}_i^l = r^{-1} \min(v-l, r)$ for $l = 1, \ldots, v-1$, $\overline{\pi}_i^l = 0$ for $l = v, \ldots, L+r-1$, 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 $SP(q, \overline{Z})$. □

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 $i = 1, \ldots 5$. Inequality* (12) *for this partition is a Benders cut* (7) *for some vector $(\overline{\pi}, \overline{\sigma})$ feasible for $SP(q, \overline{Z})$.*

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, □

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	–

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 $\lceil \overline{Z} \rceil \in \mathcal{Z}_i$. $\square$

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 $L \geq 5$, there exists a $K \geq 2$ for which the inclusion $\mathcal{Z}_i \subset \mathcal{Z}_c$ holds strictly.*

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,

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

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). $\square$

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 incoporate 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, consider the graph shown in Figure 4. We see that it is impossible to find two edge-disjoints paths from o to d with length smaller than or equal to 5, but Huygens and Mahjoub were not able to provide any inequality that cuts off this design. We describe next a violated Benders cut that cuts off the solution depicted in Figure 4. Let $G = (V, E)$ be the graph in Figure 4, $\overline{Z}_{ij}$ the binary vector made of 9 consecutive ones, and q a commodity in G from o to d. First, note that a violated Benders cut must exist because it is impossible to find a fractional flow satisfying (1),(2) and (5) in the layered graph constructed from G (so that its dual is unbounded, yielding a violated Benders cut).

Then, 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. Morover, they yield the Benders cut

$$0.5Z_{og} + 0.5Z_{cg} + 0.5Z_{gd} \geq 2, \tag{14}$$

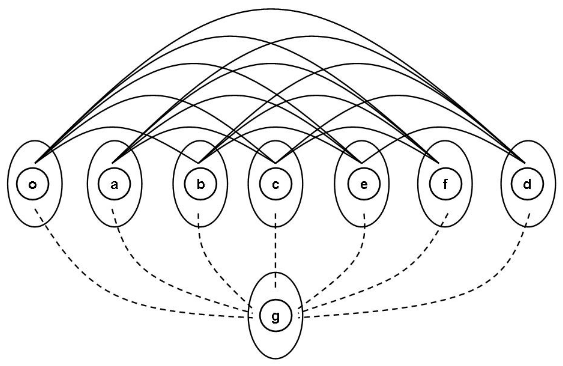


Figure 4: New inequality for $L = 5$.

violated by $\overline{Z}$. This cut can be extended for more general graphs, partitioning the nodes into 8 subsets, see Figure 4. Then, setting $(\overline{\pi}, \overline{\sigma})$ accordingly, we obtain a cut involving the edges represented in Figure 4, with coefficients equal to 1 (respectively 0.5) for plain (respectively dashed) edges. Similar cuts can be obtained in this way for the other examples in Huygens and Mahjoub (2007). Such cuts were independently discovered by Gouveia et al. (2009).

5. Computational results

In this section we compare the solution times of formulations (P) and (P') (see the commentary below), which we denote `layered` and `layered-r`, respectively, cutting plane algorithms `cp` and `cp-i`, 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, although 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 also 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 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. 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-bound tree with the dynamic search.

`cp`: We build an empty IP for the master problem, $|Q|$ LP's for the Benders subproblems and $|Q|$ IP for the feasibility subproblems. Then, we cycle between these problems, with all parameters kept to their default values. CPLEX uses the dynamic search for solving all IP's.

`cp-i`: We build an empty LP for the master problem and solve it through a cutting plane algorithm. Then, we build an IP for the master problem, the constraints of which are the Benders cuts just generated; we solve it through a cutting plane algorithm, with all parameters kept to their default values. CPLEX uses the dynamic search.

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

`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`.

`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 node, setting *IntParam.NodeLim* to 0). Then, we fix some of the variables to 0, and resolve the resulting problem with `bc-n`.

5.2 Instances details

We used three different tests 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: $|V| = 21$,

Table 4: Instances description.

Name	$\lvert N\rvert$	$\lvert E\rvert$	$\lvert Q\rvert$	# 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	27	1	false
di-yuan	11	42	48	1	false
dfn-gwin	11	47	9	1	false
polska	12	18	17	1	false
nobel-us	14	21	33	1	false

$|Q| \in \{5, 10\}$, 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, five instances are based on sparse networks from *SNDlib* (Orlowski et al., 2007): pdh, di-yuan, dfn-gwin, polska, nobel-us. Table 4 summarizes the size of the instances. We solved the problem for L ranging from 3 to 5 and K from 1 to 3. We set a time limit of 3600 seconds for all instances and algorithms.

5.3 Results

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 5 shows that the gap $\left(\frac{IP^* - LR^*}{IP^*} * 100\right)$ decreases as K increases. It means that the extended formulation integrates well the survivability constraints.

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` can not 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

Table 5: Geometric average of gap $\left(\frac{IP^* - LR^*}{IP^*} * 100\right)$ for all instances.

L/K	1	2	3
3	21.40	10.90	5.47
4	24.30	9.45	5.69
5	26.70	6.00	5.16

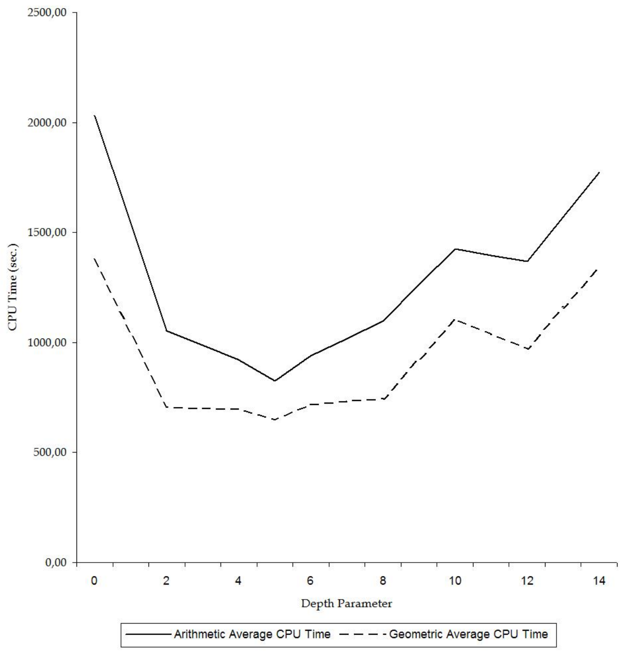


Figure 5: `bc-n` depth parameter tuning by average CPU time (sec.)

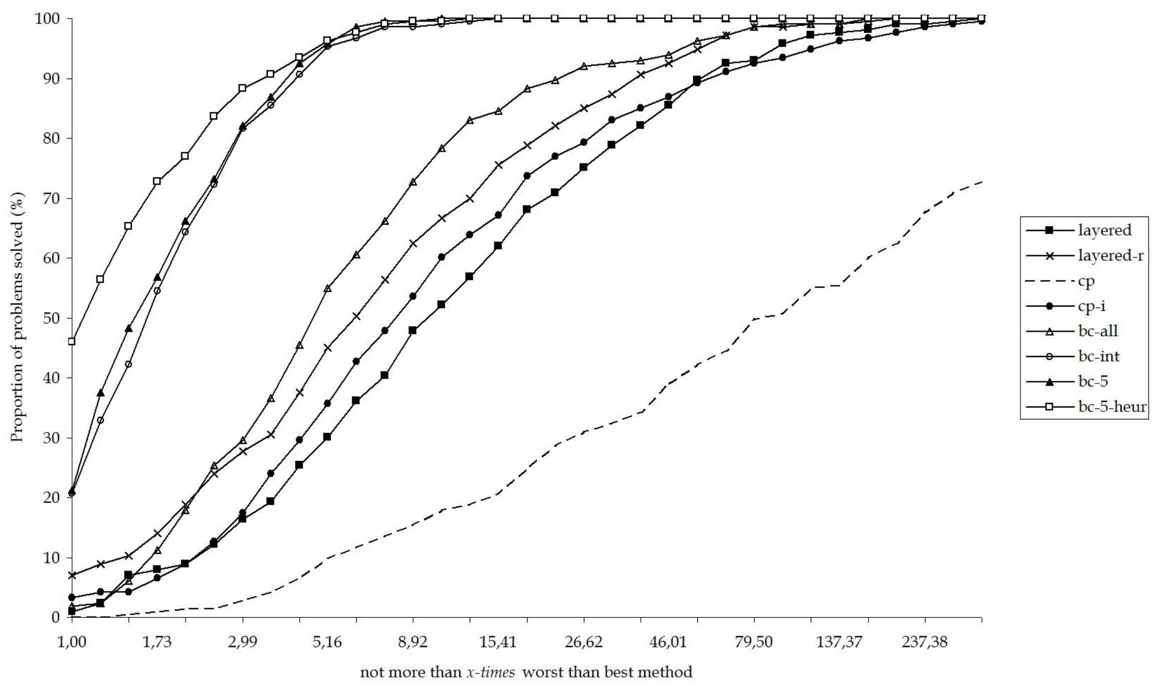


Figure 6: Performance profile comparing methods on the entire test set.

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. Moreover, we see that the simple and naive implementation of the Benders decomposition `cp` performs much worse than the original model `layered`. Indeed, Benders decomposition suffers from the loss of problem structure, so that each of the iterations of the master problem requires a sensible amount of time (see Table 9 for averages numbers of iterations). Thus, a careful implementation is required to make the decomposition efficient.

Table 6 indicates, for each algorithm, the number of instances (out of the 213 instances which compose the entire test set) that can not be solved within the 3600 seconds. Among the 13 or 12 instances that `layered` or `layered-r` can not solve to optimality, only 1 can not be solved by `bc-5`. `b-c-heur` can solve all instances to optimality. The reader can find the arithmetic averages of CPU times on Table 5.3 (> indicates that one or more instances could not be solved to optimality).

Table 5.3 indicates the arithmetic average values of the optimal solutions (IP*), linear programming relaxations (LP*), heuristic solutions (heuristic*), and `heuristic` CPU time

Table 6: Number of instances (out of 213) unsolved within one hour.

layered	layered-r	cp	cp-i	bc-all	bc-int	bc-5	bc-5-heur
13	12	90	30	20	4	1	0

Table 7: Average CPU times for all approaches.

	layered	layered-r	cp	cp-i	bc-all	bc-int	bc-5	bc-5-heur	heuristic
Arithmetic Mean	> 582.05	> 498.09	> 1817.87	> 680.27	> 494.40	> 151.87	> 89.78	74.03	3.78

in seconds. Moreover, the arithmetic averages of the GAP between LP* and IP* are computed, $\frac{IP^{*}-LP^{*}}{IP^{*}}$ and between heuristic* and IP*, $\frac{heuristic^{*}-IP^{*}}{IP^{*}}$. Column "Optimal" provides the number of instances for which the heuristic solution is optimal. It can be seen that `heuristic` always provide a very good solution to the problem. Furthermore, `heuristic` is also pretty fast, taking around 4 seconds whereas `layered` and `bc-5-heur` take respectively on average 582.05 and 74.03 seconds. In 139 cases out of 213 (around 65%), the solution given by the heuristic is the optimal one. Finally, Table 10 presents arithmetic 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 11 provides means of CPU time spent for solving Benders subproblems and the corresponding fraction in the total CPU time (means have been taken over all instances). The results detailed for each instance can be found at

`http://www.ulb.ac.be/di/gom/publications/technical/2010/BendersHOP/DetailedResults.pdf`.

Moreover, all our instances can be downloaded in text format at

`http://www.ulb.ac.be/di/gom/publications/technical/2010/BendersHOP/data.zip`.

Table 8: LP Relaxation, Integer optimal solution and heuristic performances for the entire test set.

Instances	**IP***	**LP***	**Gap(%)**	**heuristic***	**Gap(%)**	**Optimal**	**heuristic CPU Time**
TC-5	248.27	231.97	8.04	252.07	1.77	24/45	0.76
TC-10	382.51	343.34	12.96	384.00	0.42	31/45	4.78
TE-5	316.78	291.03	10.67	322.36	2.17	24/45	0.97
TE-10	446.20	379.06	17.72	447.98	0.36	30/45	9.39
pdh	1480322.56	1216582.21	21.25	1480285.35	0.00	9/9	6.51
di-yuan	46156666.67	37860533.01	20.96	46156666.67	0.00	9/9	2.21
dfn-gwin	114280.00	97639.51	16.33	114753.33	0.41	3/6	0.59
polska	302420.00	274481.67	11.11	302420.00	0.00	5/5	0.37
nobel-us	13966000.00	12806125.00	10.00	13966000.00	0.00	4/4	1.39
Arithmetic Mean	2285715.50	1901093.08	13.13	2285729.93	1.01	-	3.78

Table 9: Average numbers of iterations.

Instances	`cp`	`cp-i`	
		linear (MP)	integer (MP)
TC-5	54.07	20.53	11.11
TC-10	97.00	28.22	22.09
TE-5	229.87	34.71	36.64
TE-10	126.29	37.58	51.51
pdh	174.67	13.33	111.44
di-yuan	81.33	9.89	20.67
dfn-gwin	94.33	17.00	29.50
polska	56.60	11.40	4.80
nobel-us	147.25	11.50	21.75
Arithmetic Mean	124.73	27.52	32.57

Table 10: Number of cuts generated and of nodes visited for the entire test set.

Instances	**Number of cuts generated**				**Number of nodes visited**					
	`bc-all`	`bc-int`	`bc-5`	`bc-5-heur`	`bc-all`	`bc-int`	`bc-5`	`bc-5-heur`	`layered`	`layered-r`
TC-5	342.98	120.33	128.20	94.71	135.27	405.69	311.98	917.58	118.27	149.09
TC-10	6616.56	567.80	780.49	378.36	885.60	6825.44	3545.40	13972.93	1241.31	741.11
TE-5	757.13	236.62	248.40	159.73	169.29	1165.84	520.51	2204.31	206.98	177.56
TE-10	17154.53	1693.60	2008.31	1029.40	1310.00	48490.58	13983.40	43201.62	3402.56	2207.69
pdh	6270.33	1219.22	1205.89	1103.22	2652.78	7831.33	6773.78	9663.11	4464.56	2009.89
di-yuan	1349.44	585.22	592.22	541.56	210.89	753.78	549.33	1614.56	1202.89	243.78
dfn-gwin	1221.33	289.67	310.17	200.83	415.00	1029.33	909.17	787.17	632.00	410.00
polska	110.00	110.20	103.80	105.00	16.00	31.40	25.00	88.80	8.20	14.80
nobel-us	224.25	238.75	219.25	234.75	17.75	47.00	28.25	242.50	45.25	14.25
Arithmetic Mean	5617.64	644.65	760.01	433.20	661.60	12411.86	4215.30	13244.02	1308.13	799.38

Table 11: CPU time spent for solving Benders subproblems.

	`cp`	`cp-i`	`bc-all`	`bc-int`	`bc-5`	`bc-5-heur`	`heuristic`
CPU time (Arithmetic mean)	1.39	0.81	86.31	9.94	10.46	6.14	1.91
Fraction of total time (Geometric mean)	0.25	1.34	37.59	23.98	29.81	23.41	17.31

Acknowledgments

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".

References

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

Achterberg, T., C. Raack. 2009. The MCF-separator – detecting and exploiting multi-commodity flows in MIPs. ZIB-Report 09-38, Konrad Zuse Zentrum fuer Informationstechnik Berlin.

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.

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. doi: http://dx.doi.org/10.1016/j.orl.2005.09.001.

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. 2008. Minimal infeasible subsystems and Benders cuts. Technical Report.

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. Programming* **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. doi: http://dx.doi.org/10.1016/j.orl.2009.05.007.

Gabrel, V., A. Knippel, M. Minoux. August 1999. Exact solution of multicommodity network optimization problems with general step cost functions. *Oper. Res. Lett.* **25** 15–23(9).

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. doi: http://dx.doi.org/10.1002/net.10069.

Gouveia, L., A.R. Mahjoub, P. Pesneau. 2009. Personal communication.

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.

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. doi: http://dx.doi.org/10.1002/net.v49:2.

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

ILOG. 2007. *ILOG CPLEX 11.0 Reference Manual.*. ILOG CPLEX Division, Gentilly, France.

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

Kerivin, H., A. R. Mahjoub. 2005. Design of survivable networks: A survey. *Networks* **46** 1–21. doi:http://dx.doi.org/10.1002/net.20072.

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

M. Grötschel, 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.

Magnanti, T.L., R.T. Wong. 1981. Accelerating Benders decomposition: Algorithmic enhancement and model selection criteria. *Oper. Res.* **29** 464–484.

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. 2007. SNDlib 1.0–Survivable Network Design Library. *Proceedings of INOC, Spa, Belgium*. URL `http://www.zib.de/orlowski/Paper/OrlowskiPioroTomaszewskiWessaely2007-SNDlib-INOC.pdf`. Http://sndlib.zib.de.

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

Rei, W., J.-F. Cordeau, M. Gendreau, P. Soriano. 2009. Accelerating Benders decomposition by local branching. *INFORMS J. Comput.* **21** 333–345. doi: http://dx.doi.org/10.1287/ijoc.1080.0296.

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

Tsamasphyrou, P., A. Renaud, P. Carpentier. 2000. Transmission network planning under uncertainty with Benders decomposition. *Lecture Notes Econ. Math. Systems* **481** 457–472.

Woolston, K. A., S. L. Albin. 1988. The design of centralized networks with reliability and availability constraints. *Comput. & OR* **15** 207–217. doi:http://dx.doi.org/10.1016/0305-0548(88)90033-0.