Computers & Operations Research 67 (2016) 102–112

Contents lists available at ScienceDirect

Computers & Operations Research

journal homepage: www.elsevier.com/locate/caor

A branch-and-cut algorithm for two-level survivable network design problems

Inmaculada Rodríguez-Martín [a,*], Juan-José Salazar-González [a], Hande Yaman [b]

[a] DMEIO, Facultad de Ciencias, Universidad de La Laguna, Tenerife, Spain
[b] Department of Industrial Engineering, Bilkent University, Ankara, Turkey

ARTICLE INFO

Available online 28 September 2015

Keywords:
Network design
Survivability
Hierarchical networks
Valid inequalities
Branch-and-cut

ABSTRACT

This paper approaches the problem of designing a two-level network protected against single-edge failures. The problem simultaneously decides on the partition of the set of nodes into terminals and hubs, the connection of the hubs through a backbone network (first network level), and the assignment of terminals to hubs and their connection through access networks (second network level). We consider two survivable structures in both network levels. One structure is a two-edge connected network, and the other structure is a ring. There is a limit on the number of nodes in each access network, and there are fixed costs associated with the hubs and the access and backbone links. The aim of the problem is to minimize the total cost. We give integer programming formulations and valid inequalities for the different versions of the problem, solve them using a branch-and-cut algorithm, and discuss computational results. Some of the new inequalities can be used also to solve other problems in the literature, like the plant cycle location problem and the hub location routing problem.

1. Introduction

In this paper, we study several two-level network design problems with survivability requirements in both levels. Telecommunication networks are usually multilayer hierarchical networks where the traffic from different origins are collected and sent to upper levels to be routed towards their destinations. In a typical two-level network, the upper level is called the *backbone network*, and connects the hubs (concentrators, switches, multiplexers) among themselves. The lower level networks are called *access networks* and they connect the users to hubs. Klincewicz [18] uses the notation “backbone structure/access structure” to specify the structure of a two-level network. For instance in a “fully connected/ring network”, the backbone network is a complete graph between the hubs, and the access networks are rings, each visiting a subset of users and one hub.

Network survivability, which is the ability of a network to continue functioning in the case of failures, is one of the most critical issues in the design of telecommunications networks. A common assumption is that at most one edge can fail at a time in a network. To ensure survivability in case of single edge failures, the most common topology used is a ring. A ring is a special case of a 2-edge connected subgraph where each node has degree two. A 2-edge connected subgraph (2EC) provides the same level of survivability as a ring in case of edge failures, and may result in less redundant capacity reservation (see, e.g., Karaşan et al. [15] and Shi and Fonseka [29]). We consider these two topologies in the design of a two-level network with protection against a single edge failure. As a result, we study the design problems associated with four different networks: 2EC/2EC, 2EC/ring, ring/2EC and ring/ring networks. We will denote all these problems with the general term of 2-level survivable network design problem (2-LSNDP).

In a 2-LSNDP we are given a set of nodes. The cost of connecting a pair of nodes by a link in the backbone or in an access network is known. There is also a cost associated to select a node as hub. The number of nodes in each access network is limited by the capacity of the hubs, which is a priori given. The problem consists of choosing the nodes to act as hubs and connecting them through a backbone network, and of assigning the non-hub nodes to the hubs and connecting them through access networks, respecting the capacity and the topology requirements. The objective is to minimize the total cost of the resulting two-level network. Fig. 1 shows a ring/ring 2-LSNDP optimal solution for an instance with 15 nodes where the number of nodes in each access network is limited to 3. The solid lines represent the backbone network and the dashed lines represent the access networks. The nodes in the backbone network are the hubs. Fig. 2 shows the

* Corresponding author.
E-mail addresses: irguez@ull.es (I. Rodríguez-Martín), jjsalaza@ull.es (J.-J. Salazar-González), hyaman@bilkent.edu.tr (H. Yaman).

http://dx.doi.org/10.1016/j.cor.2015.09.008
0305-0548/© 2015 Elsevier Ltd. All rights reserved.

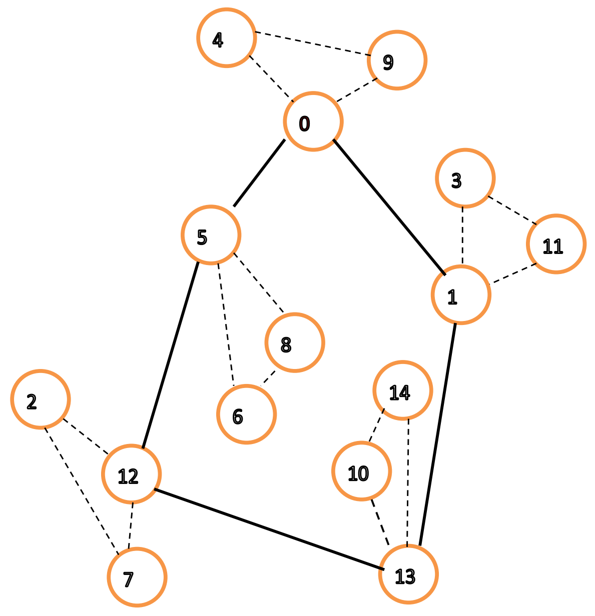

Fig. 1. A ring/ring solution example.

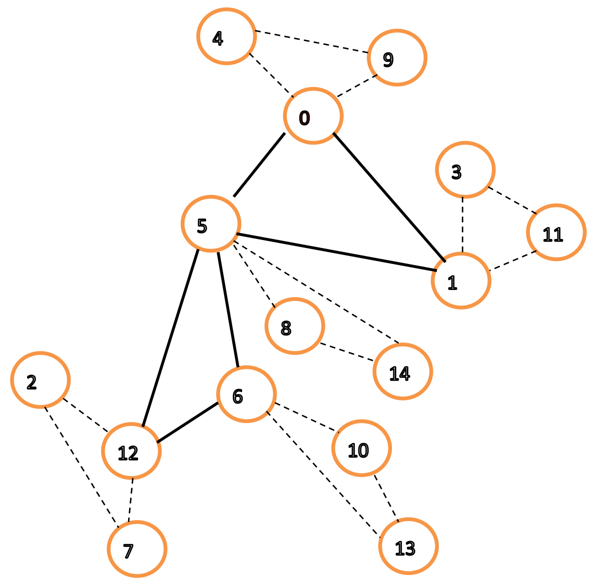

Fig. 2. A 2EC/2EC solution example.

optimal solution for the same instance when the required network structure is 2EC/2EC.

Even though survivability is critical for service providers, there are few studies on designing hierarchical survivable networks. Most studies on survivable network design problems consider a single layer of the network. For reviews of these studies, one can refer to Grötschel et al. [13] and Kerivin and Mahjoub [16]. Polynomially solvable special cases of the survivable network design problem are studied in Kerivin and Mahjoub [17]. The most common network structure in this field is a 2EC subgraph (see, e.g., [23,24,31,33]). Problems related with designing rings of bounded size are studied by Fortz and Labbé [7] and Fortz et al. [8–10]. Generalizations of 2EC networks are studied by Magnanti and Raghavan [22] and Balakrishnan et al. [2].

There are also studies on two-level networks with survivability requirements on the backbone network. For example, Labbé et al. [20] propose a branch-and-cut algorithm for designing a ring as backbone network, while the access networks are direct connections from users to a hub (i.e., star structure). Baldacci et al. [6] address a more general problem where the backbone network allows m rings instead of a single one. Fouilhoux et al. [11] study the variant where the ring structure is replaced by a 2EC network. In all these studies the access networks are forced to be star structures.

The studies that consider survivability at both layers of the network are few. Lee and Koh [21] study the ring/chain network design problem with dual homing where the ring topology in the backbone network is given. They study the design of the access networks. They show that the problem is NP-hard, propose an integer programming formulation and describe a tabu search heuristic. Thomadsen and Stidsen [32] study the ring/ring network design problem. They suggest to solve the design problems associated with different levels sequentially. They propose a branch-and-price algorithm for this purpose. Carroll and McGarraghy [3] also propose to decompose the problems of designing the rings in different levels. Shi and Fonseka [28] study the design of hierarchical self healing rings and propose a heuristic. Proestki and Sinclair [26] and Shi and Fonseka [29] propose heuristic algorithms for the problem with dual homing. Balakrishnan et al. [1] study a generalization of the two-level survivable network design problem to that of multitiers. They analyze worst-case performances of some heuristics for some special cases. Park et al. [25] study a node clustering problem with survivability requirements. Karasan et al. [15] propose a branch-and-cut algorithm for the problem where the backbone network is 2EC and each user node is connected directly to two distinct hub nodes. More recently, Hill and Voß [14] introduce the capacitated ring tree problem where nodes are connected with rings and trees, and rings intersect at a distributor node.

For further related studies, we refer the readers to the following surveys. Klincewicz [18] reviews design problems that involve location of hubs. Gourdin et al. [12] survey location problems encountered in telecommunications network design. Soriano et al. [30] provide an overview of design and dimensioning problems in survivable SDH/SONET networks. Studies on combined location and network design problems are reviewed by Contreras and Fernández [4].

In summary, there are few studies that consider the design of two-level networks with survivability requirements in both levels. Most of such studies are on designing ring/ring networks and most of the proposed approaches are of heuristic nature. The contribution of this paper is to propose strong formulations and exact solution methods for the two-level survivable network design problem where both rings and 2-edge connected networks are used to ensure survivability.

The remainder of the paper is organized as follows. The mathematical model for the different variants of the problem is given in Section 2. Section 3 presents several families of valid inequalities to strengthen the linear-programming relaxations. Section 4 details a branch-and-cut approach based on the formulation and inequalities presented in the previous sections. The performance of this approach is analyzed in Section 5 on a large collection of instances. Finally, the paper ends with conclusions in Section 6.

2. MIP models

We first introduce the notations. Let $V = \{0, 1, \ldots, n-1\}$ be the set of nodes, where node 0 stands for the root and is considered to be a hub. Note that this is not a restrictive assumption as one can solve the problem for different root nodes if one has not been a priori fixed. Let $E = \{\{i,j\} : i,j \in V, i<j\}$ be the set of potential links. We assume $G = (V,E)$ to be an undirected graph and we do not

allow multiple edges. Installing a hub at node $j \in V$ has a cost f_j. For each edge $e \in E$, the cost of installing a backbone link or an access link on e is denoted by b_e and a_e, respectively. For $S \subseteq V$, let $\delta(S)$ be the set of edges with one endpoint in S, and let $E(S)$ be the set of edges with both endpoints in set S. When S is a singleton, i.e., $S=\{i\}$, we use $\delta(i)$ for $\delta(\{i\})$.

The aim of the 2-LSNDP is to partition the set of nodes V into disjoint subsets, each with at most q nodes, choose one node from each subset to locate a hub, and connect the hubs and the subsets with survivable networks at minimum cost. The survivable network may be either a 2EC structure or a ring, depending on the problem variant. For simplicity in notation, rings or 2EC structures are assumed to involve at least three nodes. The four variants of the problem can be described with the same mathematical variables as follows. Let us define z_{ij} to be 1 if node $i \in V$ is assigned to hub $j \in V$, and to be 0 otherwise. Node j is a hub when z_{jj} is 1. In addition, we define x_e to be 1 if edge $e \in E$ is used in an access network and 0 otherwise, and y_e to be 1 if edge e is used in the backbone network, and 0 otherwise. For brevity in notation, we write $x(E')=\sum_{e \in E'} x_e$ and $y(E')=\sum_{e \in E'} y_e$ for all $E' \subseteq E$, and $z(S:T)$ instead of $\sum_{i \in S, j \in T} z_{ij}$ for all $S, T \subseteq V$.

The 2EC/2EC design problem, where the backbone and access networks are required to be 2-edge connected, can be modeled as follows:

$$\min \quad \sum_{i \in V} f_i z_{ii} + \sum_{e \in E} a_e x_e + \sum_{e \in E} b_e y_e \tag{1}$$

$$\text{s.t.} \quad z(i:V)=1 \quad \forall i \in V, \tag{2}$$

$$z(V:i) \le q z_{ii} \quad \forall i \in V, \tag{3}$$

$$z_{00}=1, \tag{4}$$

$$z_{ij}+y_{\{i,j\}} \le z_{jj} \quad \forall \{i,j\} \in E, \tag{5}$$

$$z_{ji}+y_{\{i,j\}} \le z_{ii} \quad \forall \{i,j\} \in E, \tag{6}$$

$$y(\delta(S)) \ge 2z(i:S) \quad \forall S \subseteq V \setminus \{0\}, \ i \in S, \tag{7}$$

$$x(\delta(S)) \ge 2z(i:V \setminus S) \quad \forall S \subset V, \ i \in S, \tag{8}$$

$$x_{\{i,j\}}+z_{ii'}+z_{jj'} \le 2 \quad \forall \{i,j\} \in E, \quad i',j' \in V : i' \ne j', \tag{9}$$

$$x_e, y_e \in \{0,1\} \quad \forall e \in E, \tag{10}$$

$$z_{ij} \in \{0,1\} \quad \forall i,j \in V. \tag{11}$$

The objective function (1) is the sum of the cost of locating hubs and the cost of installing access and backbone links. Constraints (2) ensure that each node is either a hub or it is assigned to another node. Constraints (3) are capacity constraints that limit to q the number of nodes assigned to a hub. They also ensure that no node is assigned to a non-hub node. Constraint (4) forces the root node to be a hub. If an edge is used in the backbone network then, due to constraints (5) and (6), both endpoints should be hubs. Otherwise, one endpoint can be assigned to the other only if the latter is a hub. Constraints (7) impose 2-edge connectedness of the backbone network. If node i is a hub or if it is assigned to a hub node in set S, then there exists at least one hub in set S and the constraint asks for at least two backbone edges on the cut $\delta(S)$ since the root is in $V \setminus S$. Similarly, constraints (8) ensure 2-edge connectedness of the access networks. If node $i \in S$ is allocated to a hub node in $V \setminus S$ then there should be at least two access links between S and $V \setminus S$. Constraints (9) make sure that if the access link $\{i,j\}$ is used then i and j are allocated to the same hub. Finally, constraints (10) and (11) are variable restrictions.

The above formulation needs minor modifications to model the other 2-LSNDP variants. In particular, to model the 2EC/ring and ring/ring design problems we add the degree constraints

$$x(\delta(i)) \le 2 \quad \forall i \in V. \tag{12}$$

These constraints limit to at most two the degree of a node in an access network and, together with the connectivity constraints (8), force those networks to be rings. Note that we cannot use an equation in (12) because not all nodes must be necessarily on an access network.

Similarly, to model the ring/2EC and ring/ring design problems we add the degree equations

$$y(\delta(i)) = 2z_{ii} \quad \forall i \in V, \tag{13}$$

to ensure that each hub node i has two backbone edges adjacent to it.

3. Valid inequalities

This section presents several families of valid inequalities for the four variants of the 2-LSNDP. We first give valid inequalities that are adopted from the literature. Validity proofs are omitted as they are very similar to the ones that appear in the cited references. Later the section presents theorems with new inequalities, some of which generalize other inequalities for different problems in the literature.

Rodríguez-Martín et al. [27] study the hub location routing problem. Several families of valid inequalities for that problem are also valid for our problems. In particular, let $[i,j] \in E$ and $S \subset V$ such that $i \in S$ and $j \in V \setminus S$. The inequality

$$x_{\{i,j\}} \le z(i:V \setminus S)+z(j:S) \tag{14}$$

is valid for 2-LSNDP. The inequality says that if node i is assigned to a hub in S and j to a hub in $V \setminus S$, i.e., if $z(i:V \setminus S)=0$ and $z(j:S)=0$, then as they are in separate access networks, edge $\{i,j\}$ cannot be used as an access edge. Note that these inequalities are stronger than constraints (9).

Fouilhoux et al. [11] derive F-partition inequalities for the network design problem where the backbone is 2EC and the access networks are stars. The feasible set of this problem is a relaxation of the 2EC/2EC network design problem obtained by dropping the constraints related with x_e variables. Hence the valid inequalities proposed by Fouilhoux et al. [11] are also valid for our problems. Let $V_0, \dots, V_p$ be a partition of V such that $V_l \ne \emptyset$, for $l=0,\dots,p$ and $0 \in V_0$. Let $i_l \in V_l$ for $l=1,\dots,p$ and $F \subseteq \delta(V_0)$ such that $|F|=2k+1$ for some integer $k \ge 0$. Let $\delta(V_0, \dots, V_p)$ be the set of edges whose endpoints are in different sets of the partition. The inequality

$$y(\delta(V_0,\dots,V_p) \setminus F)+\sum_{l=1}^{p} z(i_l : V \setminus V_l) \ge p-k \tag{15}$$

is called F-partition and it is valid for 2-LSNDP.

Baïou and Mahjoub [5] use the following constraints to avoid the occurrence of bridges, i.e., of cuts of cardinality one:

$$y(\delta(S)) \ge 2y_e \quad \forall S \subset V, \ e \in \delta(S), \tag{16}$$

which are valid for 2-LSNDP. Clearly a network is protected against the failure of an edge when no edge is a bridge.

We now present three new families of inequalities. The first family extends the classical subtour elimination constraints.

Theorem 1. *Let $S \subset V$ be a non-empty set. Let $(S_1, \dots, S_{m_1})$ be a partition of S and $(T_1, \dots, T_{m_2})$ be a partition of $V \setminus S$. Consider $i_1, \dots, i_{m_2}$ distinct nodes in S and $j_1, \dots, j_{m_1}$ distinct nodes in $V \setminus S$. Then the* *inequality*

$$x(\delta(S)) \geq 2\left(\sum_{k=1}^{m_2} z(i_k : T_k) + \sum_{l=1}^{m_1} z(j_l : S_l)\right) \tag{17}$$

is valid for 2-LSNDP.

Proof. If $z(i_k : T_k) = 0$ and $z(j_l : S_l) = 0$ for all k and l then the constraint vanishes. Otherwise, each i_k assigned to a hub in T_k and each j_l assigned to a hub in S_l involves a different access network, which implies that at least two access edges cross the cut $\delta(S)$ for each.□

Inequalities (17) generalize the following inequalities introduced by Labbé et al. [19] for the plant cycle location problem:

$$x(\delta(S)) \geq 2\big(z(i : V\backslash S) + z(j : S)\big) \quad \forall S \subset V, \ i \in S, \ j \in V\backslash S. \tag{18}$$

Note that inequalities (18) dominate constraints (8).

The second family of inequalities exploits the capacity limitation, as in the classical vehicle routing problem.

Theorem 2. *Consider $S \subset V$, a partition $(S_1, \ldots, S_m)$ of S and distinct nodes $j_1, \ldots, j_m$ be in $V\backslash S$. Let r(S) be a lower bound on the number of access networks serving nodes in S. Then the inequality*

$$x(\delta(S)) \geq 2\left(r(S) - \sum_{i \in S} z_{ii} + \sum_{k=1}^{m} z(j_k : S_k)\right) \tag{19}$$

is valid for 2-LSNDP.

Proof. There are at least $r(S) - \sum_{i \in S} z_{ii}$ distinct hubs in set $V\backslash S$ that serve nodes in S. There are also at least $\sum_{k=1}^{m} z(j_k : S_k)$ distinct hubs in S serving nodes in $V\backslash S$. Hence, overall at least 2 $(r(S) - \sum_{i \in S} z_{ii} + \sum_{k=1}^{m} z(j_k : S_k))$ access edges cross the cut $\delta(S)$.□

Examples of lower bounds $r(S)$ are $\lceil \frac{|S|}{q} \rceil$ and $\frac{|S| + x(\delta(S))/2}{q}$, though the latter is valid only when the access networks are required to be rings and not for the general 2-LSND. Rodríguez-Martín et al. [27] use the inequalities

$$x(\delta(S)) \geq 2\left(\lceil \frac{|S|}{q} \rceil - \sum_{i \in S} z_{ii}\right) \tag{20}$$

$$x(\delta(S)) \geq 2\left(\frac{|S| + x(\delta(S))/2}{q} - \sum_{i \in S} z_{ii}\right) \tag{21}$$

to solve the hub location routing problem. Note that inequalities (19) dominate these inequalities.

A particular case of inequalities (19) are

$$x(\delta(S)) \geq 2\left(\lceil \frac{|S|}{q} \rceil - \sum_{i \in S} z_{ii} + z(j : S)\right) \quad \forall S \subset V, \ j \in V\backslash S. \tag{22}$$

Finally, the next result proposes a third family of inequalities. These inequalities are different from previous inequalities as they involve variables associated with both access and backbone edges.

Theorem 3. *Let $S \subseteq V\backslash\{0\}$ be such that $S \neq \emptyset$, let $(S_1, \ldots, S_m)$ be a partition of S, and consider $j_1, \ldots, j_m$ be distinct nodes in $V\backslash S$. Let r(S) be a lower bound on the number of access networks serving nodes in S. The inequality*

$$y(E(S)) + y(\delta(S)) + x(\delta(S)) \geq r(S) + 1 + 2\sum_{k=1}^{m} z(j_k : S_k) \tag{23}$$

is valid for 2-LSNDP.

Proof. Given a feasible solution, let $S'_1, \ldots, S'_p$ be the partition of the set S into access networks. We know that $p \geq r(S)$ since each access network can have at most q nodes. Let i_k be the hub serving the access network S'_k, $K = \{1, \ldots, p\}$, $K' = \{k \in K : i_k \in S'_k\}$ and $S' = \cup_{k \in K'} S'_k$.

If $K' \neq \emptyset$, then $y(E(S)) + y(\delta(S)) = y(E(S')) + y(\delta(S')) \geq |K'| + 1$. For $k \in K\backslash K'$, we know that there are at least two access edges between S'_k and $V\backslash S$, i.e., $x(S'_k : V\backslash S) \geq 2$. In addition $x(\delta(S)) \geq \sum_{k \in K\backslash K'} x(S'_k : V\backslash S)$. So $y(E(S)) + y(\delta(S)) + x(\delta(S)) \geq |K'| + 1 + 2(|K\backslash K'|)$. If $K' = \emptyset$, then since $x(S'_k : V\backslash S) \geq 2$ for $k = 1, \ldots, p$, we have $y(E(S)) + y(\delta(S)) + x(\delta(S)) \geq 2p$. Both $|K'| + 1 + 2(|K\backslash K'|)$ and $2p$ are greater than or equal to $p+1$, and $p + 1 \geq r(S) + 1$. So inequality $y(E(S)) + y(\delta(S)) + x(\delta(S)) \geq r(S) + 1$ is satisfied.

Now there are at least $\sum_{k=1}^{m} z(j_k : S_k)$ distinct nodes in $V\backslash S$ that are assigned to distinct hub nodes in S. Hence there are also 2 $\sum_{k=1}^{m} z(j_k : S_k)$ access edges crossing the cut.□

A particular case of inequality (23) is

$$y(E(S)) + y(\delta(S)) + x(\delta(S)) \geq \left\lceil \frac{|S|}{q} \right\rceil + 1 + 2z(j : S) \tag{24}$$

where $j \in V\backslash S$. When S a is singleton, inequality (24) becomes

$$x(\delta(i)) + y(\delta(i)) \geq 2 + 2z_{ji} \tag{25}$$

where $i, j \in V$ and $i \neq j$.

4. Branch-and-cut algorithm

We propose an exact branch-and-cut algorithm to solve the 2-LSNDP based on the MIP model strengthened with the valid inequalities presented in the previous section. The branch-and-cut approach consists of a cutting plane technique embedded into a branch-and-bound framework. We describe next the main features of the algorithm.

4.1. Initialization

To start the optimization we solve the linear program (LP) given by the constraints (2)–(6) and variable bounds. To these inequalities, we add the degree inequalities (12) when the access networks are required to be rings, i.e., when the variant to solve is the 2EC/ring or the ring/ring. Similarly, to solve the design problems with required ring structure in the backbone network (i.e., ring/2EC and ring/ring) we add constraints (13). When the backbone network does not need to be a ring, we replace (13) by (7) with $S = \{i\}$ for all $i \in V$ in the initial LP.

4.2. Cutting plane phase

If the optimal solution of the LP relaxation is integer, we check whether it is a feasible solution for the 2-LSNDP by applying the separation routines for constraints (14), (7) and (8). Otherwise, we apply the separation procedures for constraints (14), (25), (7), (8), (24), (22), (16), (15), and (18), in this sequence. The separation procedure for the last family of constraints is applied only if no other violated cuts have been found, due to its computational cost. The cutting plane phase is performed only each 10 branch-and-cut nodes. Moreover, the number of violated cuts of each family added to the model is limited to 15, and the total number of cuts added in each cut generation step is limited to 75.

Constraints (25) are separated in $O(n^2)$ by complete enumeration. To separate constraints (7), (8), (14), and (18), we follow approaches presented in [20] and [27]. We devised separation methods for constraints (16), (22), and (24). To separate the F-partition inequalities (15) we use a heuristic algorithm given in [11].

Finally, constraints (17) are not directly separated, but their violation is checked each time a violated inequality (8) or (18) is found. The left hand side of all these inequalities is the same and they do not involve variables related to the backbone network. The violation test for (17) is done heuristically by checking whether

$$x(\delta(S)) \geq 2\left(\sum_{k \in V\setminus S} \max_{i \in S} z(i:k) + \sum_{l \in S} \max_{j \in V\setminus S} z(j:l)\right).$$

We next outline the separation procedures we have implemented. We refer to the variable values of the current fractional solution by (x^*, y^*, z^*).

4.2.1. Separation of inequalities (7)

To separate constraints (7) we use an exact and polynomial procedure presented in [20], and inspired by the known separation algorithm for the subtour elimination constraints for the travelling salesman problem. The procedure consists of solving max-fow/min-cut problems on an appropriately defined support graph. Note that inequalities (7) can be written as

$$y(\delta(S)) + 2z(i : V\setminus S) \geq 2 \quad \forall S \subseteq V\setminus\{0\}, \ \ i \in S.$$

For each node $i \in V\setminus\{0\}$, we define a support graph $G' = (V', E')$ with $V' = V$ and $E' = E$. The capacity of edges $\{i,j\} \in E$ with $j \neq i$ is set to $y^*_{ij} + 2z^*_{ij}$, and all other edges $e \in E$ are assigned a capacity equal to y^*_e. Next we determine a min-cut set $S \subset V'$ with $i \in S$ and $0 \notin S$, and finally we check the violation of inequality (7) for that set.

4.2.2. Separation of inequalities (8)

Constraints (8) can be separated in polynomial time with a procedure similar to the previous one. Just note that inequalities (8) can be written as

$$x(\delta(S)) + 2z(i : S) \geq 2 \quad \forall S \subset V, \ \ i \in S.$$

Since constraints (8) are dominated by constraints (18), each time we find a violated inequality (8) we replace it by the constraint (18) defined as

$$x(\delta(S)) \geq 2\big(z(i : V\setminus S) + z(j : S)\big),$$

where j is a node in $V\setminus S$ that maximizes $z^*(j : S)$.

4.2.3. Separation of inequalities (14)

Inequalities (14) can be separated in polynomial time. For a given edge $\{i,j\} \in E$, we define $S = \{i\} \cup \{k \in V\setminus\{j\} : z^*_{ik} \geq z^*_{jk}\}$. If the inequality for this choice of S is not violated, then there exists no violated inequality (14) for edge $\{i,j\}$.

4.2.4. Separation of inequalities (15)

Constraints (15) are heuristically separated using a procedure presented in [11] that works as follows. Let $G' = (V', E')$ be a support graph with $V' = \{i \in V : z^*_{ii} > 0\} \cup \{i \in V : z^*(i : V\setminus\{i\}) > 0$ and $y^*(\delta(i)) > 0\}$ and $E' = \{\{i,j\} \in E : y^*_{ij} > 0\}$. We look for sets of nodes $\{v_1, \ldots, v_p\}$ that determine odd cycles in G'. For each such set, we define $V_0 = V\setminus\{v_1, \ldots, v_p\}$, and $V_i = \{v_i\}$ for $i = 1, \ldots, p$. The set F is formed by taking an odd number of edges from $\delta(V_0)$ with fractional values $y^*_e > 1/2$. Then, the corresponding inequality is checked for violation.

4.2.5. Separation of inequalities (16)

The bridge inequalities (16) can be separated exactly in polynomial time with the following algorithm. For each edge $e \in E$ with $y^*_e > 0$ we determine in the support graph $G' = (V', E')$ with $V' = V$ and $E' = \{e' \in E : y^*_{e'} > 0\}$ the min-cut set S separating the two nodes associated with e. If the capacity of this cut is less than $2y^*_e$, the corresponding inequality (16) is violated.

Although this algorithm is polynomial it might produce a large number of similar violated inequalities, with the consequent loss of time. Therefore we use the following strategy to reduce the number of min-cut computations. We consider only edges $e = \{i,j\} \in E$ with $y^*_e > 0.5$ and such that i and j have not been extremes of an edge producing a violated inequality previously. Moreover, we associate to each set S in a violated inequality a given label, and we check for duplicates before adding the new cut to the LP. Note that, with these changes, the separation procedure for inequalities (16) becomes heuristic.

4.2.6. Separation of inequalities (18)

Constraints (18) can be written as

$$x(\delta(S)) + 2z(i : S) + 2z(j : V\setminus S) \geq 4 \quad \forall S \subset V, \ \ i \in S, \ \ j \in V\setminus S.$$

For each pair of nodes $i,j \in V$ let us define a support graph $G' = (V', E')$ where $V' = V$ and E' contains all edges $\{i,k\} \in E$ such that $k \in V$ and $x^*_{ik} + 2z^*_{jk} > 0$, all edges $\{j,k\} \in E$ such that $k \in V$ and $x^*_{jk} + 2z^*_{ik} > 0$, and all other edges $e \in E$ such that $x^*_e > 0$. The capacity of the edges in E' is set to the positive value considered for their definition. Let $S \subset V'$ be such that $i \in S$, $j \notin S$, and $\delta(S)$ is the minimum cut between i and j in G'. If the capacity of $\delta(S)$ is smaller than 4, S defines the most violated constraint (18) for i and j. Therefore, again the separation problem can be solved exactly by performing a min-cut computation for each pair of nodes.

4.2.7. Separation of inequalities (22)

To separate constraints (22) we use a heuristic procedure based on the separation algorithm for (20) described in [27]. It starts by separating exactly and in polynomial-time the inequalities without the rounding up operator, i.e.

$$x(\delta(S)) \geq 2\left(\frac{|S|}{q} - \sum_{i \in S} z_{ii} + z(j : S)\right)$$

with $S \subset V\setminus\{0\}$ and $j \in V\setminus S$. These inequalities are equivalent to

$$x(\delta(S)) + \frac{2|V\setminus S|}{q} + 2z(j : V\setminus S) + \sum_{i \in S} 2z_{ii} \geq 2\left(\frac{|V|}{q} + 1\right).$$

Then, for each $j \in V$, finding a set S defining the most violated inequality (if any) by a given solution (x^*, y^*, z^*) aims at performing a s–t min-cut computation on a capacitated network $G' = (V', E')$ with $V' = V \cup \{s\} \cup \{t\}$, being s and t two dummy nodes. The edges in E' are the edges in E, with capacity x^*_e; the edges connecting s with each $i \in V$, with capacity $2(z^*_{ji} + 1/q)$; and the edges connecting each $i \in V$ with t, with capacity $2z^*_{ii}$. To guarantee that $j \notin S$ we must increase the capacity of the edge $\{j,t\}$ by adding a large amount. We finally check the potential violation of the inequality (22) defined by the set S in the side of s generated by the s–t min-cut computation.

4.2.8. Separation of inequalities (24)

This section describes a heuristic separation for (24) based on an exact and polynomial-time separation for a closely related family of inequalities. These inequalities are the ones obtained by not considering the rounding up operator in (24), i.e.

$$y(E(S)) + y(\delta(S)) + x(\delta(S)) \geq \frac{|S|}{q} + 1 + 2z(j : S),$$

with $S \subset V\setminus\{0\}$ and $j \in V\setminus S$. These inequalities are equivalent to

$$y(E(S)) + \frac{y(\delta(S))}{2} - \frac{|S|}{q} + x(\delta(S)) + \frac{y(\delta(S))}{2} + 2z(j : V\setminus S) \geq 3,$$

and the first three terms in the left-hand side can be written as

$$y(E(S)) + \frac{y(\delta(S))}{2} - \frac{|S|}{q} = \sum_{i \in S}\left(\frac{y(\delta(i))}{2} - \frac{1}{q}\right) = \sum_{\substack{i \in S: \\ y(\delta(i)) > 2/q}}\left(\frac{y(\delta(i))}{2} - \frac{1}{q}\right)$$

$$+\sum_{\substack{i\notin S:\\ y(\delta(i))<2/q}}\left(\frac{1}{q}-\frac{y(\delta(i))}{2}\right)-\sum_{\substack{i\in V:\\ y(\delta(i))<2/q}}\left(\frac{1}{q}-\frac{y(\delta(i))}{2}\right).$$

Then, for each $j\in V$, finding a set S defining the most violated inequality (if any) by a given solution (x^*, y^*, z^*) aims at performing a s–t min-cut computation on a capacitated network G' with $V' = V\cup\{s\}\cup\{t\}$ and where each edge $e\in E$ has capacity $x_e^*+y_e^*/2$; each edge connecting s with $i\in V$ has capacity $2z_{ji}^*$, plus $1/q-y^*(\delta(i))/2$ if this value is positive; and each edge connecting t with $i\in V$ has capacity $y^*(\delta(i))/2-1/q$ if this value is positive. In addition, to guarantee that $j\notin S$ and $0\notin S$, we must increase the capacity of the edges $\{j,t\}$ and $\{0,t\}$ by adding a large amount. We check the potential violation of the inequality (24) defined by the set S in the side of s generated by this min-cut computation.

This separation algorithm can be adapted to deal with other inequalities (23) with $m=1$, like for example when $r(S)$ is defined as in (21).

4.3. Branching strategy

The branching strategy we devised prioritizes the set of variables z_{jj}. This is done because fixing the set of hubs contributes to determine other features of the solution. Branching on the other variables is done only when all variables z_{jj} are integral. In any case, we choose to branch on the fractional variable whose value in the linear relaxation solution is closer to 0.5.

5. Computational results

The branch-and-cut algorithm described in the previous section was coded in C++ and ran on a personal computer with a processor Intel Core i7 CPU at 3.4 GHz and 16 GB of RAM. We used CPLEX 12.5 as mixed integer linear programming solver. Default settings for CPLEX were used, except for the variable selection strategy that we set to "strong branching".

The performance of the algorithm was tested on two different classes of instances. They consist of networks with a number of nodes n in $\{15, 20, 30, 40\}$. In the instances of the Class I, edge costs c_{ij} are randomly generated in $[1, 100]$, while in the instances of the Class II the nodes are placed in a plane with the coordinates uniformly distributed in $[0, 100]$ and the edge costs c_{ij} are computed as the Euclidean distance between the points i and j. In all cases, the access and backbone link costs are defined as $a_e = c_e$ and $b_e = 4c_e$ respectively. Thus, on our benchmarks, the construction costs of the links are symmetric and the cost of installing a backbone link between two nodes is four times higher than the cost of installing an access link. The hub capacity q takes values in $\{\lceil 3n/4\rceil, \lceil n/2\rceil, \lceil n/4\rceil\}$. For each combination of number of nodes and capacity, we tried three different hub fix cost settings, randomly generating f_j in $[1, 500]$, $[500, 1000]$ or $[1200, 1700]$. So there are 36 different instances in each class.

For each instance we solved the four 2-LSNDP variants given by the different possible combinations of backbone and access

Table 1
Results for the ring/ring network design problem.

f_j	n	q	Class I instances opt	%-gap	cpu	nodes	nCuts	Class II instances opt	%-gap	cpu	nodes	nCuts
	15	$\lceil 3n/4\rceil$	825	0.85	0.22	7	143	1386	0.00	0.14	0	159
		$\lceil n/2\rceil$	825	0.97	0.16	3	133	1444	2.88	0.73	73	349
		$\lceil n/4\rceil$	1105	6.78	2.57	105	598	1792	2.51	1.97	13	460
	20	$\lceil 3n/4\rceil$	1193	0.21	0.34	4	214	1291	3.15	2.28	117	629
		$\lceil n/2\rceil$	1198	0.60	0.44	16	187	1323	5.38	12.07	313	1196
		$\lceil n/4\rceil$	1359	4.25	33.56	421	1988	1713	6.84	319.96	1294	4340
$[1, 500]$	30	$\lceil 3n/4\rceil$	963	0.03	2.59	17	589	1315	2.21	3.93	95	813
		$\lceil n/2\rceil$	967	0.34	1.62	16	487	1336	2.21	12.29	143	1158
		$\lceil n/4\rceil$	1139	3.92	87.77	701	2034	1633	7.74	362.87	959	5024
	40	$\lceil 3n/4\rceil$	728	0.10	2.09	3	483	1426	1.15	56.16	373	2113
		$\lceil n/2\rceil$	731	0.51	13.98	151	948	1438	1.16	62.48	371	2056
		$\lceil n/4\rceil$	852	2.75	3484.22	6311	5683	1715	5.12	2368.50	2124	7460
	15	$\lceil 3n/4\rceil$	2533	0.34	0.16	11	191	2737	0.00	0.08	0	168
		$\lceil n/2\rceil$	2553	1.21	0.37	63	252	2752	0.38	0.17	5	204
		$\lceil n/4\rceil$	3284	1.94	1.89	53	497	3581	1.83	1.00	35	370
	20	$\lceil 3n/4\rceil$	2682	0.21	0.27	8	185	2733	0.59	0.64	58	290
		$\lceil n/2\rceil$	2686	0.41	0.97	58	318	2799	2.91	10.80	254	1178
		$\lceil n/4\rceil$	3497	1.41	11.61	98	1051	3775	2.69	93.46	466	2617
$[500, 1000]$	30	$\lceil 3n/4\rceil$	2551	0.32	1.37	44	395	2735	1.48	14.21	238	927
		$\lceil n/2\rceil$	2560	0.47	7.64	175	859	2763	1.54	62.07	973	2038
		$\lceil n/4\rceil$	3262	2.73	145.24	939	2526	3564	3.17	1186.59	1609	7058
	40	$\lceil 3n/4\rceil$	2446	0.20	2.64	31	540	2482	0.58	23.81	211	1304
		$\lceil n/2\rceil$	2446	0.14	11.95	118	764	2505	0.95	66.44	602	2149
		$\lceil n/4\rceil$	3207	3.33	1869.63	2420	6306	3364	3.12 (1.25)	t.l.	3307	11003
	15	$\lceil 3n/4\rceil$	4425	0.23	0.14	11	125	4986	0.00	0.11	0	111
		$\lceil n/2\rceil$	4425	0.27	0.17	12	145	5044	0.77	0.83	89	406
		$\lceil n/4\rceil$	5905	1.21	3.21	159	561	6592	0.47	1.61	11	586
	20	$\lceil 3n/4\rceil$	4793	0.04	0.62	31	366	4891	0.66	1.28	46	375
		$\lceil n/2\rceil$	4798	0.13	0.48	30	216	4923	1.46	5.29	169	994
		$\lceil n/4\rceil$	6159	0.86	27.22	394	1568	6513	1.38	101.46	700	2667
$[1200, 1700]$	30	$\lceil 3n/4\rceil$	4563	0.01	1.33	3	475	4677	0.63	9.06	213	973
		$\lceil n/2\rceil$	4567	0.09	2.34	12	528	4682	0.37	7.27	108	962
		$\lceil n/4\rceil$	5939	0.73	72.07	614	1838	6215	1.29	947.82	2411	5403
	40	$\lceil 3n/4\rceil$	4328	0.02	1.78	6	484	5026	0.33	31.40	257	1693
		$\lceil n/2\rceil$	4331	0.09	15.85	113	1040	5038	0.56	70.37	454	2541
		$\lceil n/4\rceil$	5652	0.41	1942.87	4708	5514	6515	1.66	1496.46	1835	9233

Table 2
Results for the ring/2EC network design problem.

f_j	n	q	Class I instances					Class II instances				
			opt	%-gap	cpu	Nodes	nCuts	opt	%-gap	cpu	Nodes	nCuts
	15	$\lceil 3n/4\rceil$	813	0.68	0.17	11	132	1386	0.00	0.09	0	150
		$\lceil n/2\rceil$	824	2.00	0.30	25	174	1444	1.55	0.59	41	365
		$\lceil n/4\rceil$	1105	7.25	5.46	202	761	1792	3.03	1.44	17	563
	20	$\lceil 3n/4\rceil$	1190	0.62	0.58	14	366	1291	2.99	0.98	78	326
		$\lceil n/2\rceil$	1198	1.05	0.62	32	276	1323	5.45	12.78	274	1224
		$\lceil n/4\rceil$	1359	5.14	13.45	311	829	1713	5.44	154.16	865	2974
$[1, 500]$	30	$\lceil 3n/4\rceil$	960	0.15	0.58	3	357	1315	2.18	11.78	161	1147
		$\lceil n/2\rceil$	963	0.23	1.51	10	535	1336	0.90	7.04	82	851
		$\lceil n/4\rceil$	1139	4.00	122.68	845	2637	1633	8.21	137.36	510	3166
	40	$\lceil 3n/4\rceil$	727	0.61	4.10	81	616	1426	1.15	23.63	352	1116
		$\lceil n/2\rceil$	728	0.75	3.39	45	514	1438	1.04	23.15	210	1230
		$\lceil n/4\rceil$	844	2.37	985.26	2703	3560	1715	5.33	5406.92	2404	14,500
	15	$\lceil 3n/4\rceil$	2529	0.66	0.27	21	220	2737	0.00	0.11	0	188
		$\lceil n/2\rceil$	2532	0.84	0.20	21	182	2752	0.21	0.37	4	283
		$\lceil n/4\rceil$	3284	1.62	3.67	39	604	3581	2.69	3.51	23	467
	20	$\lceil 3n/4\rceil$	2674	0.15	0.30	5	177	2733	0.59	0.42	29	213
		$\lceil n/2\rceil$	2683	0.54	0.53	33	232	2799	2.93	13.17	392	1145
		$\lceil n/4\rceil$	3497	1.74	13.60	156	1078	3775	2.63	119.17	638	2598
$[500, 1000]$	30	$\lceil 3n/4\rceil$	2551	0.23	1.26	36	364	2735	0.94	27.71	282	1803
		$\lceil n/2\rceil$	2559	0.69	5.12	173	719	2763	1.44	55.99	761	1984
		$\lceil n/4\rceil$	3261	2.22	164.25	1236	2161	3564	3.85	1099.14	1653	6886
	40	$\lceil 3n/4\rceil$	2438	0.08	2.31	17	427	2482	0.37	13.99	76	1577
		$\lceil n/2\rceil$	2444	0.32	5.60	89	579	2505	1.36	49.06	302	1908
		$\lceil n/4\rceil$	3206	3.34	4396.75	6173	6741	3384	3.62 (1.74)	t.l.	2970	13,259
	15	$\lceil 3n/4\rceil$	4413	0.07	0.17	7	136	4986	0.00	0.12	0	107
		$\lceil n/2\rceil$	4424	0.32	0.23	13	151	5044	0.82	1.08	121	392
		$\lceil n/4\rceil$	5905	1.94	2.84	142	610	6592	0.78	1.17	11	537
	20	$\lceil 3n/4\rceil$	4790	0.16	0.55	19	282	4891	0.66	2.36	90	644
		$\lceil n/2\rceil$	4798	0.19	1.64	67	423	4923	1.44	8.67	330	1062
		$\lceil n/4\rceil$	6159	1.21	9.02	241	869	6513	1.48	155.66	969	2834
$[1200, 1700]$	30	$\lceil 3n/4\rceil$	4560	0.00	0.39	0	329	4677	0.42	5.21	124	752
		$\lceil n/2\rceil$	4563	0.07	4.17	88	744	4682	1.04	29.30	299	1944
		$\lceil n/4\rceil$	5939	0.71	175.17	1353	2539	6215	1.18	2794.35	4670	6806
	40	$\lceil 3n/4\rceil$	4327	0.10	5.05	68	596	5026	0.33	59.02	524	2213
		$\lceil n/2\rceil$	4328	0.13	11.14	131	890	5038	0.32	19.47	232	1026
		$\lceil n/4\rceil$	5644	0.35	1346.27	3711	4026	6515	1.48	1826.69	2043	7920

network topologies. The detailed results for all the instances are shown in Tables 1–4. Column headings stand for:

- f_j: Hub fix cost.
- n: Number of nodes.
- q: Hub capacity.
- *opt*: Optimal objective function value.
- *%-gap*: Percentage gap between the optimal value (*opt*) and the lower bound (*lb*) at the end of the root node, that is, $100(opt-lb)/opt$.
- *cpu*: Total computing time, in seconds.
- *nodes*: Number of nodes in the search tree.
- *nCuts*: Total number of generated cuts.

We imposed a time limit of two hours for each run. When this time limit is exceeded, we report "t.l." in column *cpu*, use the best solution at the end of the computation instead of the optimal to compute the figure in *%-gap*, and report in brackets the final gap (i.e., the percentage gap between the best solution found and the lower bound at the end of the computation).

For the instances with random distances (Class I) we observe that as the hub fix costs increase, the LP gaps tend to decrease. Note that the maximum gap for the solved instances with $f_j \in [1, 500]$ is 8.33%, and it is 1.94% (though it is usually below 1%) for the instances with $f_j \in [1200, 1700]$. All the problems but one were solved within the time limit. Regarding the computing times, the hardest instances are those with 40 nodes and the most restrictive capacity ($q = \lceil n/4\rceil$). Note as well that, when the optimal values of the four problems over the same instance are different (see, for example, the case of instance with $n=30$ and $q=\lceil n/2\rceil$), the minimum optimal value is attained at problem 2EC/2EC, as expected when the edge costs do not satisfy the triangular inequality.

For the instances with Euclidean distances (Class II), the four 2-LSND problems have an optimal solution with rings at the two levels, even if we do not explicitly ask for it, because of the edge cost characteristics. In other words, the optimal solutions of the four variants coincide in all cases. These instances are more difficult to tackle, though only the problems dealing with the one with $n=40$, $q=\lceil n/4\rceil$ and $f_j \in [500, 1000]$ were unsolved within the time limit. The gaps are higher in general than for Class I, but it also holds that they decrease when costs f_j increase. For a given number of nodes n and fix hub cost setting f_j, the hardest instances are those with tighter capacity.

To assess the influence of the different sets of valid inequalities proposed in this work, we compare four versions of the branch-and-cut algorithm that differ by the sets of valid inequalities considered. Tables 5 and 6 show the result obtained for a subset of instances, concretely the instances of Class I and Class II with 15 nodes and $f_j \in [1, 500]$. We also conducted the experiment on larger instances, but we omit the results here because the findings are analogous to those derived from these small instances. The four branch-and-cut versions compared are:

Table 3
Results for the 2EC/ring network design problem.

f_j	n	q	Class I instances					Class II instances				
			opt	%-gap	cpu	Nodes	nCuts	opt	%-gap	cpu	Nodes	nCuts
	15	$\lceil 3n/4\rceil$	825	1.25	0.22	15	140	1386	0.14	0.12	0	145
		$\lceil n/2\rceil$	825	0.84	0.23	5	147	1444	2.69	0.45	54	299
		$\lceil n/4\rceil$	1105	8.33	4.95	158	810	1792	2.09	2.00	18	570
	20	$\lceil 3n/4\rceil$	1193	0.17	0.33	4	196	1291	3.15	2.98	126	730
		$\lceil n/2\rceil$	1198	0.50	0.95	31	427	1323	4.01	16.18	313	1257
		$\lceil n/4\rceil$	1359	4.65	17.00	339	1003	1713	7.02	554.37	1907	4459
$[1, 500]$	30	$\lceil 3n/4\rceil$	963	0.03	1.83	3	496	1315	2.58	5.21	104	815
		$\lceil n/2\rceil$	967	0.34	2.20	15	505	1336	1.58	70.29	463	2983
		$\lceil n/4\rceil$	1139	4.07	153.75	1100	2358	1633	7.84	281.52	737	4359
	40	$\lceil 3n/4\rceil$	728	0.09	1.79	4	461	1426	1.01	77.80	385	2628
		$\lceil n/2\rceil$	731	0.51	6.61	54	649	1438	2.15	92.40	420	2704
		$\lceil n/4\rceil$	852	2.75 (0.18)	t.l.	8723	7370	1715	5.76	2601.74	1805	10,206
	15	$\lceil 3n/4\rceil$	2533	0.42	0.27	16	182	2737	0.00	0.11	0	196
		$\lceil n/2\rceil$	2553	1.26	0.56	37	280	2752	0.24	0.16	5	206
		$\lceil n/4\rceil$	3284	1.41	2.11	47	725	3581	1.55	0.69	14	348
	20	$\lceil 3n/4\rceil$	2682	0.22	0.27	9	196	2733	0.61	1.17	83	632
		$\lceil n/2\rceil$	2686	0.51	0.56	52	267	2799	2.93	3.12	138	578
		$\lceil n/4\rceil$	3497	1.35	11.36	118	973	3775	2.92	102.49	618	2944
$[500, 1000]$	30	$\lceil 3n/4\rceil$	2551	0.12	2.03	27	393	2735	1.57	33.96	420	2199
		$\lceil n/2\rceil$	2560	0.50	10.05	192	884	2763	1.41	52.48	823	2062
		$\lceil n/4\rceil$	3262	2.02	204.91	1230	2584	3564	3.21	370.49	864	5263
	40	$\lceil 3n/4\rceil$	2446	0.25	2.11	31	473	2482	0.58	12.57	182	1110
		$\lceil n/2\rceil$	2446	0.14	1.92	22	438	2505	0.93	35.38	318	1721
		$\lceil n/4\rceil$	3207	3.26	968.30	1579	5088	3394	4.07 (2.09)	t.l.	3434	14,271
	15	$\lceil 3n/4\rceil$	4425	0.12	0.34	17	191	4986	0.00	0.17	0	134
		$\lceil n/2\rceil$	4425	0.28	0.22	7	138	5044	0.77	0.55	66	275
		$\lceil n/4\rceil$	5905	1.08	4.77	110	714	6592	0.82	3.53	27	569
	20	$\lceil 3n/4\rceil$	4793	0.06	0.27	4	214	4891	0.77	1.19	108	400
		$\lceil n/2\rceil$	4798	0.15	0.70	39	275	4923	1.41	4.88	263	774
		$\lceil n/4\rceil$	6159	0.92	10.30	241	876	6513	1.30	152.02	939	2643
$[1200, 1700]$	30	$\lceil 3n/4\rceil$	4563	0.05	1.28	6	450	4677	0.82	11.67	271	1098
		$\lceil n/2\rceil$	4567	0.07	1.39	13	460	4682	0.26	1.42	17	484
		$\lceil n/4\rceil$	5939	0.99	120.56	957	2406	6215	1.17	890.66	2128	6665
	40	$\lceil 3n/4\rceil$	4328	0.02	2.12	9	459	5026	0.33	53.79	410	2303
		$\lceil n/2\rceil$	4331	0.09	10.44	70	886	5038	0.56	141.62	547	3542
		$\lceil n/4\rceil$	5652	0.41	6745.20	8728	7846	6515	1.56	2454.00	2005	10,605

- *basic1 B and C*: It just solves model (1)–(11). Constraints (9) are incorporated to the initial LP, and inequalities (7) and (8) are separated. In the separation procedure for (8), the replacement of these constraints by (18) is deactivated.
- *basic2 B and C*: Like the previous one, but constraints (9) are removed from the initial LP, and instead constraints (14) are separated.
- *basic3 B and C*: Includes also inequalities (18), that replace the violated constraints (8), and that are as well separated but only if no other violated cuts have been found.
- *complete B and C*: Complete branch-and-cut algorithm described in Section 4.

For each branch-and-cut version we report the gap of the linear programming relaxation at the end of the root node and the total computing time. Note that the gaps and the computing times are large for the first and most basic algorithm, and that they decrease as we add new families of valid inequalities. The incorporation of the separation procedure for constraints (14) substantially reduces the computing times, though does not have a great impact in the gaps. In fact, in a few cases, the gaps slightly worsen. This is because the separation procedure adds to the model only those cuts whose violation exceeds a given threshold. The addition of constraints (18) reduces the gaps and the running times, this being more evident for Class II instances. The inclusion of the remaining families of valid inequalities in the complete branch-and-cut serves to further improve the results, as shown in the last two columns.

Finally, Table 7 shows the average percentage of cuts generated. The column with heading (18)' shows the number of violated constraints (18) found by applying the separation procedure for constraints (8), as described in Section 4.2.2. For each hub fix cost setting, we report in a single line the data corresponding to the 12 instances with that setting. The number of violated cuts of each family found during the branch-and-cut algorithm is highly dependant on the sequence in which the different separation procedures are applied. Taking this in mind, we observe that the most numerous cuts are, by far, (14). This may be due to the own statement of constraints (14) together with the fact that they replace constraints (9), which are necessary to define the problem, and they are separated at the beginning of the cutting plane phase. The second position corresponds to constraints (18) and (17), whose number approximately coincides. This happens because the violation of constraints (17) is checked inside the separation routines for (8) and (18), and we are certain to find a violated constraint of the first type when any of the two latter is violated. On the other extreme, only very few cuts (15) and (16) are generated. We separate them, nevertheless, because they contribute to reduce the gaps in some particular cases.

Table 4
Results for the 2EC/2EC network design problem.

f_j	n	q	Class I instances					Class II instances				
			opt	%-gap	cpu	Nodes	nCuts	opt	%-gap	cpu	Nodes	nCuts
	15	$\lceil 3n/4 \rceil$	813	0.68	0.17	9	129	1386	0.00	0.11	0	155
		$\lceil n/2 \rceil$	824	1.92	0.30	24	227	1444	2.88	1.33	127	492
		$\lceil n/4 \rceil$	1105	7.13	4.80	114	708	1792	2.05	1.78	11	655
	20	$\lceil 3n/4 \rceil$	1190	0.41	0.83	23	437	1291	2.99	1.34	94	439
		$\lceil n/2 \rceil$	1198	0.75	1.11	36	346	1323	5.45	5.93	233	975
		$\lceil n/4 \rceil$	1359	5.13	25.19	294	1501	1713	6.18	112.66	876	2614
$[1, 500]$	30	$\lceil 3n/4 \rceil$	960	0.00	0.55	0	389	1315	2.21	7.11	116	868
		$\lceil n/2 \rceil$	963	0.30	1.28	12	457	1336	1.25	6.10	99	870
		$\lceil n/4 \rceil$	1139	2.44	129.78	1183	2319	1633	7.70	217.26	740	4251
	40	$\lceil 3n/4 \rceil$	727	0.61	5.46	68	649	1426	1.45	42.84	325	2646
		$\lceil n/2 \rceil$	728	0.75	18.97	175	1258	1438	1.96	134.57	469	4047
		$\lceil n/4 \rceil$	844	2.37	1113.05	2538	4559	1715	5.99	2261.41	2293	9523
	15	$\lceil 3n/4 \rceil$	2529	0.72	0.25	11	183	2737	0.00	0.08	0	162
		$\lceil n/2 \rceil$	2532	1.10	0.39	15	210	2752	0.24	0.22	10	208
		$\lceil n/4 \rceil$	3284	1.93	2.03	50	556	3581	1.77	0.58	12	370
	20	$\lceil 3n/4 \rceil$	2674	0.26	0.39	18	222	2733	0.59	0.30	24	209
		$\lceil n/2 \rceil$	2683	0.60	0.61	39	237	2799	2.38	7.30	268	1085
		$\lceil n/4 \rceil$	3497	1.11	27.57	149	1579	3775	2.89	90.04	560	2859
$[500, 1000]$	30	$\lceil 3n/4 \rceil$	2551	0.42	2.87	79	478	2735	1.67	19.97	208	1704
		$\lceil n/2 \rceil$	2559	0.69	5.40	177	710	2763	1.42	42.06	609	2199
		$\lceil n/4 \rceil$	3261	2.29	231.94	1053	3366	3564	5.06	334.62	1069	5905
	40	$\lceil 3n/4 \rceil$	2438	0.27	1.34	20	381	2482	0.64	9.53	149	1153
		$\lceil n/2 \rceil$	2444	0.49	7.63	157	655	2505	1.37	43.42	327	2616
		$\lceil n/4 \rceil$	3206	3.38	1641.08	2842	5702	3368	3.35 (1.50)	t.l.	3944	12,624
	15	$\lceil 3n/4 \rceil$	4413	0.04	0.25	4	184	4986	0.00	0.08	0	111
		$\lceil n/2 \rceil$	4424	0.35	0.33	24	252	5044	0.82	1.20	127	474
		$\lceil n/4 \rceil$	5905	1.31	3.59	142	531	6592	0.81	2.87	26	574
	20	$\lceil 3n/4 \rceil$	4790	0.16	1.19	54	459	4891	0.66	2.28	81	638
		$\lceil n/2 \rceil$	4798	0.23	0.89	34	286	4923	1.46	6.94	245	1068
		$\lceil n/4 \rceil$	6159	1.09	13.56	299	967	6513	1.59	93.57	755	2551
$[1200, 1700]$	30	$\lceil 3n/4 \rceil$	4560	0.03	1.31	19	432	4677	0.69	7.99	154	992
		$\lceil n/2 \rceil$	4563	0.06	1.73	21	438	4682	0.11	5.35	81	770
		$\lceil n/4 \rceil$	5939	0.51	130.45	1533	1990	6215	4.52	1077.12	2834	6661
	40	$\lceil 3n/4 \rceil$	4327	0.10	2.76	51	593	5026	0.29	12.29	181	1091
		$\lceil n/2 \rceil$	4328	0.13	8.28	101	759	5038	0.37	182.26	582	4226
		$\lceil n/4 \rceil$	5644	0.35	1085.06	2788	3893	6515	1.45	1719.02	1616	9560

Table 5
Effect of adding valid inequalities on the Class I instance with $n=15$ and $f_j \in [1, 500]$.

	q	basic1 B&C		basic2 B&C		basic3 B&C		Complete B&C	
		%-gap	cpu	%-gap	cpu	%-gap	cpu	%-gap	cpu
ring/ring	$\lceil 3n/4 \rceil$	1.57	3.39	1.60	0.23	1.56	0.19	0.85	0.22
	$\lceil n/2 \rceil$	1.51	3.60	1.50	0.14	1.48	0.16	0.97	0.16
	$\lceil n/4 \rceil$	14.89	200.96	12.58	13.32	11.63	7.46	6.78	2.57
ring/2EC	$\lceil 3n/4 \rceil$	0.86	5.10	0.68	0.11	0.68	0.12	0.68	0.17
	$\lceil n/2 \rceil$	2.18	4.23	2.21	0.16	2.14	0.14	2.00	0.30
	$\lceil n/4 \rceil$	15.02	252.50	13.09	19.38	12.81	12.96	7.25	5.46
2EC/ring	$\lceil 3n/4 \rceil$	1.56	6.80	1.60	0.16	1.56	0.19	1.25	0.22
	$\lceil n/2 \rceil$	1.62	4.37	1.47	0.17	1.47	0.14	0.84	0.23
	$\lceil n/4 \rceil$	14.64	195.94	12.87	16.88	12.03	12.68	8.33	4.95
2EC/2EC	$\lceil 3n/4 \rceil$	0.86	3.67	0.82	0.11	0.82	0.12	0.68	0.17
	$\lceil n/2 \rceil$	2.21	3.28	1.92	0.23	1.92	0.20	1.92	0.30
	$\lceil n/4 \rceil$	14.48	275.81	14.01	12.92	12.43	15.30	7.13	4.80

6. Conclusions

This paper addresses a two-level survivable network design problem in which location, assignment, and network design tasks are jointly tackled. Moreover, survivability is required in the two levels of the network, and not only in one of them as in most other works in the literature.

We present an integer programming formulation and valid inequalities for the problem. This formulation models, with minor modifications, the four possible combinations of the two single-edge protected topologies considered (rings and two-edge connected networks). Several valid inequalities generalize inequalities given in the literature for related problems, and therefore they can also be used to solve them. We have designed an exact branch-

Table 6
Effect of adding valid inequalities on the Class II instance with $n=15$ and $f_j \in [1,500]$.

	q	basic1 B&C		basic2 B&C		basic3 B&C		Complete B&C	
		%-gap	cpu	%-gap	cpu	%-gap	cpu	%-gap	cpu
ring/ring	$\lceil 3n/4 \rceil$	0.22	2.40	0.22	0.17	0.00	0.12	0.00	0.14
	$\lceil n/2 \rceil$	4.22	12.68	4.09	0.98	2.88	0.70	2.88	0.73
	$\lceil n/4 \rceil$	8.13	178.08	8.58	21.09	4.33	6.47	2.51	1.97
ring/2EC	$\lceil 3n/4 \rceil$	0.22	2.00	0.07	0.12	0.00	0.11	0.00	0.09
	$\lceil n/2 \rceil$	4.22	23.77	4.10	0.67	2.88	1.15	1.55	0.59
	$\lceil n/4 \rceil$	8.13	222.49	8.43	8.38	4.33	6.07	3.03	1.44
2EC/ring	$\lceil 3n/4 \rceil$	0.22	2.54	0.22	0.12	0.00	0.11	0.14	0.12
	$\lceil n/2 \rceil$	4.22	15.04	4.22	0.94	2.88	0.78	2.69	0.45
	$\lceil n/4 \rceil$	8.13	143.01	8.04	16.21	3.05	4.99	2.09	2.00
2EC/2EC	$\lceil 3n/4 \rceil$	0.22	2.25	0.07	0.12	0.00	0.08	0.00	0.11
	$\lceil n/2 \rceil$	4.22	22.28	4.22	1.50	2.88	1.19	2.88	1.33
	$\lceil n/4 \rceil$	8.13	227.56	8.04	22.21	3.19	4.88	2.05	1.78

Table 7
Average percentages of cuts added during de branch-and-cut algorithm.

	f_j	(7)	(14)	(15)	(16)	(17)	(18)'	(18)	(22)	(24)	(25)
Class I	[1, 500]	4.59	47.55	0.02	0.07	14.36	11.63	2.73	8.31	6.60	4.15
	[500, 1000]	4.41	47.63	0.00	0.04	13.95	11.82	2.13	8.23	6.37	5.41
	[1200, 1700]	4.29	48.71	0.03	0.05	14.10	11.30	2.79	8.26	6.31	4.14
Class II	[1, 500]	5.12	40.78	0.04	0.19	17.62	10.24	7.37	7.52	5.63	5.48
	[500, 1000]	4.66	37.76	0.02	0.13	19.19	10.67	8.52	8.48	5.85	4.71
	[1200, 1700]	3.65	40.43	0.02	0.15	18.00	10.84	7.16	8.77	6.08	4.90

and-cut algorithm and we have tested its performance on two sets of instances with different degree of difficulty. The results are satisfactory since they show that instances with up to 40 nodes, which is already a large size for this kind of problems, are solved to optimality in a reasonable amount of time.

Acknowledgements

This research has been supported by “Ministerio de Economía y Competitividad”, Spain (research project MTM2012-36163-C06-01). The research of the third author is supported by the Turkish Academy of Sciences.

References

[1] Balakrishnan A, Magnanti TL, Mirchandani P. Designing hierarchical survivable networks. Oper Res 1998;46(1):116–36.
[2] Balakrishnan A, Mirchandani P, Natarajan HP. Connectivity upgrade models for survivable network design. Oper Res 2009;57(1):170–86.
[3] Carroll P, McGarraghy S. A decomposition algorithm for the ring spur assignment problem. Int Trans Oper Res 2013;20:119–39.
[4] Contreras I, Fernández E. General network design: a unified view of combined location and network design problems. Eur J Oper Res 2012;219:680–97.
[5] Baïou M, Mahjoub AR. Steiner 2-edge connected subgraph polytopes on series–parallel graphs. SIAM J Discrete Math 1997;10(3):505–14.
[6] Baldacci R, Dell'Amico M, Salazar-González JJ. The capacitated m-ring star problem. Oper Res 2007;55(6):1142–62.
[7] Fortz B, Labbé M. Two-connected networks with rings of bounded cardinality. Comput Optim Appl 2004;27:123–48.
[8] Fortz B, Labbé M, Maffioli F. Solving the two-connected network with bounded meshes problem. Oper Res 2000;48:866–77.
[9] Fortz B, Mahjoub AR, McCormick S, Pesneau P. Two-edge connected subgraphs with bounded rings: polyhedral results and branch-and-cut. Math Program 2006;105:85–111.
[10] Fortz B, Soriano P, Wynants C. A tabu search algorithm for self-healing ring network design. Eur J Oper Res 2003;151:280–95.
[11] Fouilhoux P, Karasan OE, Mahjoub AR, Ozkok O, Yaman H. Survivability in hierarchical telecommunications networks. Networks 2012;59(1):37–58.
[12] Gourdin E, Labbé M, Yaman H. Telecommunication and location. In: Drezner Z, Hamacher HW, editors. Facility location: applications and theory. Berlin, Heidelberg: Springer; 2002. p. 275–305.
[13] Grötschel M, Monma CL, Stoer M. Design of survivable communications networks. In: Ball MO, Magnanti TL, Monma CL, Nemhauser GL, editors. Network models. Amsterdam: North-Holland; 1995. p. 617–71.
[14] Hill A, Voß S. Optimal capacitated ring trees. Technical report. University of Antwerp, Faculty of Applied Economics; 2014.
[15] Karasan O, Mahjoub AR, Ozkok O, Yaman H. Survivability in hierarchical telecommunications networks under dual homing. INFORMS J Comput 2014;26:1–15.
[16] Kerivin H, Mahjoub AR. Design of survivable networks: a survey. Networks 2005;46:1–21.
[17] Kerivin H, Mahjoub AR. On survivable network polyhedra. Discrete Math 2005;290:183–210.
[18] Klincewicz JG. Hub location in backbone/tributary network design: a review. Locat Sci 1998;6:307–35.
[19] Labbé M, Rodríguez-Martín I, Salazar-González JJ. A branch-and-cut algorithm for the plant-cycle location problem. J Oper Res Soc 2004;55:513–20.
[20] Labbé M, Laporte G, Rodríguez-Martín I, Salazar-González JJ. The ring star problem: polyhedral analysis and exact algorithm. Networks 2004;43:177–89.
[21] Lee CY, Koh SJ. A design of the minimum cost ring-chain network with dual-homing survivability: a tabu search approach. Comput Oper Res 1997;24:883–97.
[22] Magnanti TL, Raghavan S. Strong formulations for network design problems with connectivity requirements. Networks 2005;45:61–79.
[23] Mahjoub AR. Two-edge connected spanning subgraphs and polyhedra. Math Program 1994;64:199–208.
[24] Mahjoub AR, Pesneau P. On the Steiner 2-edge connected subgraph polytope. RAIRO Oper Res 2008;42:259–83.
[25] Park K, Lee K, Park S, Lee H. Telecommunication node clustering with node compatibility and network survivability requirements. Manag Sci 2000;46:363–74.
[26] Proestki A, Sinclair MC. Design and dimensioning of dual-homing hierarchical multi-ring networks. IEEE Proc Commun 2000;147:96–104.
[27] Rodríguez-Martín I, Salazar-González JJ, Yaman H. A branch-and-cut algorithm for the hub location and routing problem. Comput Oper Res 2014;50:161–74.

[28] Shi JJ, Fonseka JP. Hierarchical self-healing rings. IEEE/ACM Trans Netw 1995;3:690–7.
[29] Shi JJ, Fonseka JP. Analysis and design of survivable communications networks. IEEE Proc Commun 1997;144:322–30.
[30] Soriano P, Wynants C, Seguin R, Labbé M, Gendreau M, Fortz B. Design and dimensioning of survivable SDH/sonet networks. In: Sanso B, Sariano P, editors. Telecommunications network planning. New York: Springer; 1999. p. 147–67.
[31] Stoer M. Design of survivable networks. Lecture notes in mathematics, vol. 1531. Berlin, Heidelberg: Springer-Verlag; 1992.
[32] Thomadsen T, Stidsen T. Hierarchical ring network design using branch-and-price. Telecommun Syst 2005;29:61–76.
[33] Vandenbussche D, Nemhauser GL. The 2-edge-connected subgraph polyhedron. J Comb Optim 2005;9:357–79.