INFORMS Journal on Computing

Publication details, including instructions for authors and subscription information:
http://pubsonline.informs.org

Survivability in Hierarchical Telecommunications Networks Under Dual Homing

Oya Ekin Karaşan, A. Ridha Mahjoub, Onur Özkök, Hande Yaman

To cite this article:
Oya Ekin Karaşan, A. Ridha Mahjoub, Onur Özkök, Hande Yaman (2014) Survivability in Hierarchical Telecommunications Networks Under Dual Homing. INFORMS Journal on Computing 26(1):1-15. https://doi.org/10.1287/ijoc.1120.0541

INFORMS Journal on Computing
Vol. 26, No. 1, Winter 2014, pp. 1–15
ISSN 1091-9856 (print) | ISSN 1526-5528 (online)

http://dx.doi.org/10.1287/ijoc.1120.0541

Survivability in Hierarchical Telecommunications Networks Under Dual Homing

Oya Ekin Karaşan
Department of Industrial Engineering, Bilkent University, Bilkent, 06800 Ankara, Turkey, karasan@bilkent.edu.tr

A. Ridha Mahjoub
LAMSADE, Université Paris-Dauphine, 75775 Paris, France, mahjoub@lamsade.dauphine.fr

Onur Özkök, Hande Yaman
Department of Industrial Engineering, Bilkent University, Bilkent, 06800 Ankara, Turkey
{onuroz@bilkent.edu.tr, hyaman@bilkent.edu.tr}

The motivation behind this study is the essential need for survivability in the telecommunications networks. An optical signal should find its destination even if the network experiences an occasional fiber cut. We consider the design of a two-level survivable telecommunications network. Terminals compiling the access layer communicate through hubs forming the backbone layer. To hedge against single link failures in the network, we require the backbone subgraph to be two-edge connected and the terminal nodes to connect to the backbone layer in a dual-homed fashion, i.e., at two distinct hubs. The underlying design problem partitions a given set of nodes into hubs and terminals, chooses a set of connections between the hubs such that the resulting backbone network is two-edge connected, and for each terminal chooses two hubs to provide the dual-homing backbone access. All of these decisions are jointly made based on some cost considerations. We give alternative formulations using cut inequalities, compare these formulations, provide a polyhedral analysis of the small-sized formulation, describe valid inequalities, study the associated separation problems, and design variable fixing rules. All of these findings are then utilized in devising an efficient branch-and-cut algorithm to solve this network design problem.

Key words: hierarchical network design; two-edge connectedness; dual-homing survivability; facets; branch and cut; variable fixing

History: Accepted by S. Raghavan, Area Editor for Telecommunications and Electronic Commerce; received December 2011; revised July 2012; accepted October 2012. Published online in *Articles in Advance* January 17, 2013.

1. Introduction

Today's telecommunications networks are based on wavelength division multiplexing (WDM), a technology that allows each optical fiber to communicate hundreds of optical signals each carrying dozens of Gbps (10^9 bits per second) for a total of several Tbps (10^{12} bits per second). Zhang and Mukherjee (2004) reported the frequency of link failures as 4.39 fiber cuts per year per 1000 sheath miles. According to Kraushaar (1999), the total sheath miles owned by all interexchange carriers in the United States was 159,779 in the year 1998. These figures clearly sum up to a tremendous amount of traffic being affected during each fiber cut. Ultimately, network survivability—the ability of the network to continue providing services to applications even in the case of node or link failures—has become one of the most critical issues in the design of telecommunications networks. This critical and challenging aspect of network design problems arising in such telecommunications applications has encompassed many operations research studies, including the one presented in this paper.

In the two-layer network infrastructure considered in this paper, terminals are connected via a set of hub nodes to a backbone network. To provide higher availability, defined as the probability that the network services are in the operating state at a random time, we consider a two-layer survivability mechanism that provides dual homing in the access network and two-edge connectivity in the backbone network.

The most common survivable backbone design is that of a ring network because of its simplicity in rerouting in the case of link failures. A two-edge-connected design has all the survivability advantages of a ring design at a mesh topology. The gain in simplicity of survivability of ring topologies comes at the expense of more redundant capacity reservation in contrast to mesh designs. Similar conclusions were drawn by Shi and Fonseka (1997) in their comparison of mesh-restorable and self-healing ring networks under a class of survivability measures.

Dual homing is a technique for enhancing the survivability of the access networks by allowing the terminal nodes to be connected to two distinct hub nodes in the backbone network. One of these hubs provides the primary connection, and the other one is typically activated in case the primary connection fails. Clearly, dual homing hedges against single access link failures. Given a fixed backbone network topology, Din and Tseng (2002) considered the problem of assigning each terminal node to two hub locations. Their study proposes an integer programming formulation and a genetic algorithm for the challenging case when the backbone nodes are capacitated with respect to providing access.

Within the literature involving hierarchical topologies, Gourdin et al. (2002) provided a survey that matches the telecommunications literature to that of the concentrator location problems that are more common in the operations research literature. A commonly accepted classification scheme based on the topologies of the backbone and access networks is provided in Klincewicz's (1998) survey. The most common topologies studied are stars, trees, rings, complete and mesh networks, with the access networks often being stars or trees. Klincewicz (1998) introduced the "backbone network structure/access network structure" notation to distinguish various two-layer hierarchical designs. In this survey, however, all of the access networks are considered as single-homed to the backbone network. To differentiate between single- and dual-homed access to backbone connections, we shall append Klincewicz's (1998) notation with the adjective "dual-homed" whenever necessary. In our problem, we seek to find the lowest cost survivable network design by partitioning a given set of nodes into terminals and hubs, by choosing a set of edges connecting the hub nodes such that the resulting backbone network is two-edge connected, and by assigning each terminal node to exactly two hub nodes. In particular, we look for a two-edge-connected/star (dual-homed) design. An example is given in Figure 1 where the squares correspond to hubs and the circles correspond to terminals. Solid edges represent the backbone connections, and the dashed arcs represent the connections between terminals and hubs.

Survivable network design problems are often restricted to a single layer. Grötschel et al. (1995) provided one of the earlier surveys in survivable network design. Kerivin and Mahjoub (2005a) reviewed the optimization techniques for both the capacitated and uncapacitated versions of the survivable network design problems under different connectivity requirements. Kerivin and Mahjoub (2005b) presented some polynomially solvable cases of the survivable network design problems under special

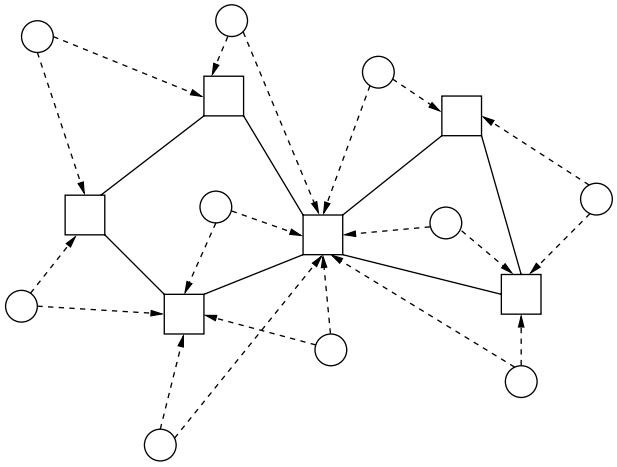

Figure 1 An Example of a Two-Edge-Connected/Star (Dual-Homed) Network

connectivity restrictions. Mahjoub (1994) provided an in-depth analysis of the polytope associated with the two-edge-connected subgraph problem. The studies by Mahjoub and Pesneau (2008), Stoer (1992), and Vandenbussche and Nemhauser (2005) also relate to the current work because they consider two-edge-connected survivable designs, though only in a single layer. Magnanti and Raghavan (2005) and Balakrishnan et al. (2009) focused on single-level survivable network design problems with connectivity requirements that generalize our two-edge-connectivity requirement in the backbone layer.

Labbé et al. (2004) performed a facial study of the ring/star network design problems and provided an efficient branch-and-cut algorithm. Fouilhoux et al. (2012) considered the two-edge-connected/star survivable network design problem, which has a close relationship to the problem under consideration. They provided a 0–1 integer programming model and a detailed analysis of the associated polytope. They studied classes of facet-defining inequalities along with the computational analysis of the respective separation problems, designed exact and/or heuristic separation algorithms, and introduced an efficient branch-and-cut algorithm. In this paper, a similar methodology is adopted for the dual-homing variation where the access network is also survivable. As it turns out, both the model development and the polyhedral analysis are more involved for this challenging design problem.

There are a few related studies that consider survivability at both layers of the hierarchical design. Lee and Koh (1997) considered the ring/ring (dual-homed) version. They assumed a fixed ring topology as the backbone network and studied the design problem of the access network, established the NP-completeness of this problem, proposed an integer

programming formulation for its solution, and finally, proposed an effective tabu search heuristic as a solution methodology. Thomadsen and Stidsen (2005) considered the same infrastructure. They provided a branch-and-price algorithm for the case when the design problems in the two levels are tackled as sequential optimization problems, hence attaining a suboptimal solution for the original problem. Proestki and Sinclair (2000) contributed an efficient heuristic for the same problem.

Balakrishnan et al. (1998) generalized the two-level hierarchical survivable network design to that of multitiers and provided a modeling framework that unifies different technological layers and connectivity requirements under the viewpoint of cost effectiveness. Most of the hierarchical design problems in the literature can be seen as special cases of these general problems. Several special cases are analyzed and worst-case performance bounds of certain heuristics are provided.

Our study offers a contribution to the existing literature by providing an exact solution methodology to the two-edge-connected/star (dual-homed) hierarchical network design problem (2ECDHP). Unlike most of the existing work in the literature, we do not limit survivability and/or design considerations to a single level. In particular, the contribution includes two alternative 0–1 model developments based on cut inequalities; a comparison of the formulations based on the strength of the linear relaxations and the complexity of the associated separation problems; a detailed polyhedral study for the small size formulation; development of exact and heuristic separation algorithms for the families of facet-defining inequalities considered; and several variable fixing rules. Finally, all of this theoretical knowhow is utilized in the development of a branch-and-cut algorithm as an exact solution methodology. Our results on networks of nearly 200 nodes indicate that the valid inequalities and the variable fixing rules are very effective in CPU time savings.

The dual-homing problem has a different combinatorial structure than the single-homing problem, and its solution requires specific developments. We can list these as follows. (i) We need a larger size formulation to obtain cut constraints that are similar to those for the single-homing case. Unfortunately, the separation problem associated with the resulting cut constraints is NP-complete. As a result, in the dual-homing problem, we work with weaker cut constraints and use valid inequalities obtained from the projection of the large size formulation as cuts. (ii) We have new families of facet-defining inequalities specific to dual homing; some are obtained from the projection of the large size formulation, whereas others are based on the idea of dual homing. We propose exact and heuristic separation routines for these new families of valid inequalities. (iii) We extend the F-partition inequalities known for the single-homing problem to the dual-homing problem. (iv) We propose some variable fixing rules that are crucial in speeding up the separation process and the solution of large instances.

This paper is organized as follows. In §2, two integer programming formulations and valid inequalities are provided for the 2ECDHP. In §3, the underlying polytope is analyzed, and necessary and sufficient conditions for the valid inequalities to be facet defining are provided. Section 4 is reserved for exact and/or heuristic separation algorithms. In §5, several rules for variable fixing are discussed. Section 6 presents the computational study, and §7 has the concluding remarks.

2. Mathematical Models

In this section, we propose two integer programming models for our 2ECDHP. The first model, called the two-index model, uses $O(n^2)$ variables, whereas the second one, called the three-index model, uses $O(n^3)$ variables (n is the number of nodes). Both models have an exponential number of "cut" inequalities. We show that the three-index model is stronger than the two-index model, but the separation problem associated with its cut inequalities is NP-complete. We study the projection of the feasible set of the linear programming (LP) relaxation of the three-index model onto the space of the two-index model and derive some valid inequalities. First we give some notation.

2.1. Notation

Let $V = \{0, 1, \ldots, n\}$ be the set of terminal nodes. Node 0 is the root node of the two-level network infrastructure, and it is a hub. We define $E = \{\{i, j\}: i \in V,\ j \in V\backslash\{i\}\}$ to be the set of potential backbone links, and $G = (V, E)$. Note that we assume a complete graph and we do not allow multiple edges. We denote by c_e the fixed setup cost associated with the backbone link $e \in E$. Similarly, d_{ij} is the cost associated with installing an access link between terminal node $i \in V\backslash\{0\}$ and hub node $j \in V$. The value d_{ii} corresponds to the cost of installing a hub at node $i \in V\backslash\{0\}$. We define $A = \{(i, j): i \in V\backslash\{0\}, j \in V\backslash\{i\}\}$.

For two disjoint subsets V_1 and V_2 of V, we denote by $[V_1, V_2]$ the set of edges with one endpoint in V_1 and the other in V_2. For $S \subseteq V$, let $\delta(S) = [S, V\backslash S]$, and let $E(S)$ be the set of edges with both endpoints in S. For simplicity, we use $\delta(i)$ instead of $\delta(\{i\})$. We use $G(S)$ to denote the subgraph induced by S, i.e., $G(S) = (S, E(S))$.

As a design problem, in 2ECDHP, we would like to find a partition of V into C and T such that $0 \in C$, a set of backbone links $E' \subseteq E(C)$ such that the graph

(C, E') is two-edge connected, and an assignment of each node in T to two distinct nodes in C such that the total cost of installing backbone links, access links, and hubs is minimum.

2.2. The Two-Index Formulation

We first present a two-index 0–1 model for 2ECDHP. We define the following decision variables:

$$x_e = \begin{cases} 1 & \text{if edge } e \in E \text{ is used in the backbone network,} \\ 0 & \text{otherwise;} \end{cases}$$

$$y_{ij} = \begin{cases} 1 & \text{if node } i \in V\backslash\{0\} \text{ is assigned to hub node } j \in V\backslash\{i\}, \\ 0 & \text{otherwise;} \end{cases}$$

$$t_i = \begin{cases} 1 & \text{if } i \in V\backslash\{0\} \text{ is a hub,} \\ 0 & \text{otherwise.} \end{cases}$$

The two-index formulation is as follows:

$$\min\left\{\sum_{i\in V\backslash\{0\}} d_{ii}t_i + \sum_{e\in E} c_e x_e + \sum_{(i,j)\in A} d_{ij}y_{ij}\right\} \tag{1}$$

s.t.

$$2t_i + \sum_{j\in V\backslash\{i\}} y_{ij} = 2 \quad \forall\, i \in V\backslash\{0\}, \tag{2}$$

$$x_e + y_{ij} \le t_j \quad \forall\,(i,j) \in A\colon j \ne 0,\ e = \{i,j\}, \tag{3}$$

$$x_e + y_{i0} \le 1 \quad \forall\, i \in V\backslash\{0\},\ e = \{i,0\}, \tag{4}$$

$$x_e \le t_i \quad \forall\, i \in V\backslash\{0\},\ e = \{i,0\}, \tag{5}$$

$$x(\delta(S)) \ge 2t_i + \sum_{j\in S\backslash\{i\}} y_{ij} \quad \forall\, S \subseteq V\backslash\{0\},\ i \in S, \tag{6}$$

$$x_e \in \{0,1\} \quad \forall\, e \in E, \tag{7}$$

$$y_{ij} \in \{0,1\} \quad \forall\,(i,j) \in A, \tag{8}$$

$$t_i \in \{0,1\} \quad \forall\, i \in V\backslash\{0\}. \tag{9}$$

The objective function (1) is the cost of installing hubs and the cost of installing backbone and access links. Constraints (2) are the *assignment constraints* that ensure that a node is either a hub or assigned to two distinct hubs. Because of constraints (3)–(5), if a node is not a hub, then no other node can be assigned to it, and no backbone link can be adjacent to it. These constraints are referred to as *conflict constraints*. Constraints (6) are the *cut inequalities* and they impose the two-edge connectedness requirement for the backbone network. They are similar to the cut inequalities proposed by Labbé et al. (2004) for the ring/star problem. Let $S \subseteq V\backslash\{0\}$ and $i \in S$. If i is a hub, then the constraint becomes $x(\delta(S)) \ge 2$ and should be satisfied because there exists at least one hub, namely, the root node, in the set $V\backslash S$. If i is not a hub but is assigned to two other hubs in S, again the constraint becomes $x(\delta(S)) \ge 2$ and imposes the installation of at least two backbone edges on the cut between S and $V\backslash S$. If i is assigned to a hub l in set S and a hub in $V\backslash S$, then the constraint reduces to $x(\delta(S)) \ge 1$. However, in this case, the cut inequality for the same set S and the hub node l is $x(\delta(S)) \ge 2$, which dominates the former inequality. Finally, if node i is assigned to two hubs in $V\backslash S$, then we do not know whether there is a hub in set S, and the right-hand side of the constraint is zero. Constraints (7)–(9) are variable restrictions.

The cut inequalities of the two-index formulation can be separated exactly in polynomial time (see Labbé et al. 2004, Fouilhoux et al. 2012) by solving a series of minimum cut problems.

2.3. The Three-Index Formulation

Now we present a stronger three-index formulation. For $i \in V\backslash\{0\}$, let $E_i = \{\{j,k\} \in E\backslash\delta(i)\}$. Set E_i is the set of distinct pairs of nodes to which node i can be assigned if it is not a hub itself. We define the following decision variables:

$$u_{i\{j,k\}} = \begin{cases} 1 & \text{if node } i \in V\backslash\{0\} \text{ is assigned to nodes } j \text{ and } k,\ \{j,k\} \in E_i, \\ 0 & \text{otherwise.} \end{cases}$$

Now, we present our three-index model for 2ECDHP:

$$\min\left\{\sum_{i\in V\backslash\{0\}} d_{ii}t_i + \sum_{e\in E} c_e x_e + \sum_{i\in V\backslash\{0\}} \sum_{\{j,k\}\in E_i} (d_{ij} + d_{ik})u_{i\{j,k\}}\right\} \tag{10}$$

s.t.

$$t_i + \sum_{\{j,k\}\in E_i} u_{i\{j,k\}} = 1 \quad \forall\, i \in V\backslash\{0\}, \tag{11}$$

$$x_e + \sum_{k\in V\colon \{j,k\}\in E_i} u_{i\{j,k\}} \le t_j \quad \forall\,(i,j) \in A,\ e = \{i,j\},\ j \ne 0, \tag{12}$$

$$x_e + \sum_{k\in V\colon \{0,k\}\in E_i} u_{i\{0,k\}} \le 1 \quad \forall\, i \in V\backslash\{0\},\ e = \{i,0\}, \tag{13}$$

$$x_e \le t_i \quad \forall\, i \in V\backslash\{0\},\ e = \{i,0\}, \tag{14}$$

$$x(\delta(S)) \ge 2t_i + 2 \sum_{\{j,k\}\in E_i\colon |\{j,k\}\cap S|\ge 1} u_{i\{j,k\}} \quad \forall\, S \subseteq V\backslash\{0\},\ i \in S, \tag{15}$$

$$x_e \in \{0,1\} \quad \forall\, e \in E, \tag{16}$$

$$u_{i\{j,k\}} \in \{0,1\} \quad \forall\, i \in V\backslash\{0\},\ \{j,k\} \in E_i, \tag{17}$$

$$t_i \in \{0,1\} \quad \forall\, i \in V\backslash\{0\}. \tag{18}$$

Here, the objective function (10) is equal to the cost of installing hubs, backbone links, and access links.

The assignment constraints (11) ensure that each node is a hub or it is assigned to two distinct nodes, and the conflict constraints (12)–(14) ensure that nodes are assigned to hub nodes and backbone links can be installed between hub nodes. The cut inequalities (15) model the requirement that the backbone network is two-edge connected. Let $S \subseteq V\backslash\{0\}$ and $i \in S$. If i is a hub, i.e., $t_i = 1$, or if i is assigned to at least one hub in set S, i.e., $\sum_{\{j,k\}\in E_i:|\{j,k\}\cap S|\geq 1} u_{i\{j,k\}} = 1$, then the constraint becomes $x(\delta(S)) \geq 2$. Otherwise, i is assigned to two hubs in $V\backslash S$, and the constraint reduces to $x(\delta(S)) \geq 0$. Constraints (16)–(18) are variable restrictions.

2.4. Comparison of the Formulations

Next, we compare the strength of the LP relaxations of these two formulations. We first remark that for $S \subseteq V\backslash\{0\}$ and $i \in S$, when i is assigned to one hub in S and one hub in $V\backslash S$, the right-hand side of the cut inequality (15) of the three-index model is equal to 2 (hence this constraint imposes the installation of a minimum of two edges on the cut between S and $V\backslash S$), whereas the cut inequality (6) of the two-index formulation has the right-hand side equal to 1.

We append the constraints $y_{ij} = \sum_{k\in V\backslash\{i,j\}} u_{i\{j,k\}}$ for all $(i,j) \in A$ to the three-index formulation. Let F' be the feasible set of the LP relaxation of the resulting formulation, and let F be the feasible set of the LP relaxation of the two-index formulation.

Theorem 1. $\text{Proj}_{x,t,y} F' \subseteq F$.

Proof. It is easy to show that the assignment and conflict constraints are the same in both formulations using the equivalence $y_{ij} = \sum_{k\in V\backslash\{i,j\}} u_{i\{j,k\}}$ for all $(i,j) \in A$. Now we study the cut inequalities. Let $S \subseteq V\backslash\{0\}$ and $i \in S$. The right-hand side of constraint (15) is equal to

$$\begin{aligned} &2t_i + 2 \sum_{\{j,k\}\in E_i:|\{j,k\}\cap S|\geq 1} u_{i\{j,k\}} \\ &\quad = 2t_i + \sum_{j\in S\backslash\{i\}} \sum_{k\in V\backslash\{i,j\}} u_{i\{j,k\}} + \sum_{\{j,k\}\in E_i\cap\delta(S)} u_{i\{j,k\}} \\ &\quad = 2t_i + \sum_{j\in S\backslash\{i\}} y_{ij} + \sum_{\{j,k\}\in E_i\cap\delta(S)} u_{i\{j,k\}}, \end{aligned}$$

and is greater than or equal to the right-hand side of the cut inequality (6). Hence, we can conclude that for any $(x,t,y,u) \in F'$, we have $(x,t,y) \in F$ and $\text{Proj}_{x,t,y} F' \subseteq F$. □

Even though the three-index formulation is stronger, our preliminary tests showed that it is not advantageous to use it in a branch-and-cut algorithm because it takes much longer to solve its LP relaxations compared to those of the two-index formulation. Next, we give another negative result about this formulation. Consider the separation problem associated with the cut inequalities (15). In particular, given a nonnegative vector (x,t,u), a fixed node $i \in V\backslash\{0\}$, and a scalar $0 < \epsilon < 2$, the separation problem seeks to find a subset $S \subseteq V\backslash\{0\}$ with $i \in S$ such that $2t_i + 2\sum_{\{j,k\}\in E_i:|\{j,k\}\cap S|\geq 1} u_{i\{j,k\}} - x(\delta(S)) \geq \epsilon$ or equivalently by (11) that $x(\delta(S)) + 2\sum_{\{j,k\}\in E_i:|\{j,k\}\cap S|=0} u_{i\{j,k\}} \leq 2-\epsilon$. We show that this problem is NP-complete.

Theorem 2. *The separation problem associated with the cut inequalities* (15) *is NP-complete.*

Proof. Clearly, the separation problem is in NP. To establish this result, we provide a reduction from the decision version of the *Vertex Cover* problem, which is defined as follows. Given a graph $G' = (V', E')$ and a positive integer K, does there exist $S' \subseteq V'$ such that $|S'| \leq K$ and nodes in S' jointly cover all edges in E' (Garey and Johnson 1979)? Given such an instance, consider the following instance of the separation problem. Let $V = V' \cup \{0,i\}$, $E = E' \cup \{\{i,j\}: j \in V'\} \cup \{\{0,j\}: j \in V\backslash\{0\}\}$, $E_i = E' \cup \{\{j,0\}: j \in V'\}$, $x_{\{j,0\}} = (2-\epsilon)/K$ for $j \in V'$, $x_{\{i,j\}} = 0$ for $j \in V\backslash\{i\}$, $x_e = 0$ for $e \in E'$, $u_{i\{j,k\}} = 1$ for $\{j,k\} \in E'$, and $u_{i\{j,0\}} = 0$ for $j \in V'$. Now, let S' be a vertex cover of size at most K in G'. Let $S = S' \cup \{i\}$. Because S' is a cover, no edge in E' has both endpoints in $V'\backslash S'$ and $\sum_{\{j,k\}\in E_i:|\{j,k\}\cap S|=0} u_{i\{j,k\}} = 0$. Then, $x(\delta(S)) + 2\sum_{\{j,k\}\in E_i:|\{j,k\}\cap S|=0} u_{i\{j,k\}} = |S'|(2-\epsilon)/K \leq 2-\epsilon$. Similarly, if inequality (15) is violated by at least ϵ for some $S \subseteq V\backslash\{0\}$ and $i \in S$, then let $S' = S\backslash\{i\}$. No edge of E' can join two nodes in $V'\backslash S'$ for otherwise $2\sum_{\{j,k\}\in E_i:|\{j,k\}\cap S|=0} u_{i\{j,k\}} \geq 2$. Thus, S' is a node cover for G'. □

2.5. Projection Inequalities

Because the three-index formulation has large linear relaxations and it is difficult to separate its cut inequalities, we use the two-index formulation in our branch-and-cut algorithm. It is possible to strengthen this formulation using valid inequalities that are obtained from the projection of the three-index formulation.

We present two such families of valid inequalities. Let $S \subseteq V\backslash\{0\}$, $i \in S$, and $j \in S\backslash\{i\}$. The "in-cut" inequality is

$$x(\delta(S)) \geq 2t_i + 2y_{ij}. \tag{19}$$

The validity of this inequality can be explained as follows. If node i is a hub or if it is assigned to hub j in set S, then because there exists at least one hub in set S, the inequality imposes the installation of at least two backbone edges on the cut between S and $V\backslash S$. Otherwise, the inequality is redundant.

Similarly, the validity of the following inequality is intuitive. Let $S \subseteq V\backslash\{0\}$, $i \in S$, and $j \in V\backslash S$. The "out-cut" inequality is

$$x(\delta(S)) \geq 2t_i + \sum_{k\in S\backslash\{i\}} y_{ik} + y_{ij} - \sum_{k\in V\backslash(S\cup\{j\})} y_{ik}. \tag{20}$$

If node i is a hub, or if it is assigned to two hubs in set S, or if it is assigned to one hub in set S and to hub j in set $V\backslash S$, then the inequality is $x(\delta(S)) \geq 2$ and should be satisfied because there exists at least one hub in set S. In the remaining cases, the right-hand side is nonpositive and the inequality is redundant.

More formally, we describe Theorem 3 as follows.

Theorem 3. *In-cut* (19) *and out-cut* (20) *inequalities are implied by the three-index formulation*.

Proof. By Farkas' Lemma (see, e.g., Schrijver 1986), given (x, t, y), there exists a vector u that satisfies

$$\sum_{\{j,k\}\in E_i\cap\delta(S)} u_{i\{j,k\}} \leq x(\delta(S)) - 2t_i - \sum_{j\in S\backslash\{i\}} y_{ij} \quad \forall S \subseteq V\backslash\{0\},\ i \in S, \tag{21}$$

$$\sum_{k\in V:\,\{j,k\}\in E_i} u_{i\{j,k\}} = y_{ij} \quad \forall (i,j) \in A, \tag{22}$$

$$u_{i\{j,k\}} \geq 0 \quad \forall i \in V\backslash\{0\},\ \{j,k\} \in E_i, \tag{23}$$

if and only if

$$\sum_{i\in V\backslash\{0\}} \sum_{S\subseteq V\backslash\{0\}:\, i\in S} \left(x(\delta(S)) - 2t_i - \sum_{j\in S\backslash\{i\}} y_{ij}\right)\alpha_{iS} + \sum_{(i,j)\in A} \beta_{ij} y_{ij} \geq 0, \tag{24}$$

for all vectors (α, β) such that

$$\sum_{S\subseteq V\backslash\{0\}:\,|S\cap\{j,k\}|=1,\, i\in S} \alpha_{iS} + \beta_{ij} + \beta_{ik} \geq 0 \quad \forall i \in V\backslash\{0\},\ \{j,k\} \in E_i, \tag{25}$$

$$\alpha_{iS} \geq 0 \quad \forall S \subseteq V\backslash\{0\},\ i \in S. \tag{26}$$

First note that the above system can be disaggregated for each node $i \in V\backslash\{0\}$. Now consider a vector (α, β) where $\alpha_{iS} = 1$ for some $S \subseteq V\backslash\{0\}$ with $i \in S$ and all other entries of α are zero. One feasible β vector is $\beta_{ij} = -1$ for some $j \in S\backslash\{i\}$, $\beta_{ik} = 1$ for all $k \in S\backslash\{i, j\}$, and other entries of β are zero. This yields the in-cut projection inequality.

Next consider a vector (α, β) where $\alpha_{iS} = 1$ for some $S \subseteq V\backslash\{0\}$ with $i \in S$ and all other entries of α are zero. Let $j \in V\backslash S$. Consider the vector β where $\beta_{ij} = -1$, $\beta_{ik} = 1$ for all $k \in V\backslash(S\cup\{j\})$, and other entries of β are zero. The resulting projection inequality is the out-cut inequality. □

We conclude this section with the following remark. If we replace the cut inequalities (6) with the in-cut inequalities (19) or with the out-cut inequalities (20) in the two-index formulation, we obtain two alternative formulations for 2ECDHP. It is not possible to compare these two-index formulations among themselves (later, we prove that all three families of inequalities (6), (19), and (20) are facet defining under some conditions). However, we can conclude that the three-index formulation is stronger than these two new formulations because inequalities (19) and (20) are projection inequalities.

2.6. Dual-Homing and Extended F-Partition Inequalities

If a node $i \in V\backslash\{0\}$ is assigned to a hub, say $j \in V\backslash\{i\}$, then it is not a hub node and must be assigned to a second hub node. This yields the following family of valid inequalities named as "dual-homing" inequalities:

$$y_{ij} \leq \sum_{k\in V\backslash\{i,j\}} y_{ik}. \tag{27}$$

We can also extend the family of F-partition inequalities introduced by Mahjoub (1994) to 2ECDHP. A similar extension was performed by Fouilhoux et al. (2012) for the single assignment version of the problem. Let $V_0, \ldots, V_p$ be a partition of V such that $V_l \neq \varnothing$, for $l = 0, \ldots, p$ and $0 \in V_0$. Let $i_l \in V_l$ be a fixed node for $l = 1, \ldots, p$ and $F \subseteq \delta(V_0)$ such that $|F| = 2k+1$ for some $k \geq 0$ and integer. Let $\delta(V_0, \ldots, V_p)$ be the set of edges whose endpoints are in different sets of the partition.

Consider the following valid inequalities for 2ECDHP:

$$x(\delta(V_l)) + \sum_{j\in V\backslash V_l} y_{i_l j} \geq 2 \quad l = 1, \ldots, p, \tag{28}$$

$$-x_e \geq -1 \quad \forall e \in F, \tag{29}$$

$$x_e \geq 0 \quad \forall e \in \delta(V_0)\backslash F. \tag{30}$$

Adding up these inequalities, dividing the resulting inequality by 2, and rounding up the right-hand side yields:

$$x(\delta(V_0, \ldots, V_p)\backslash F) + \frac{\sum_{l=1}^{p} \sum_{j\in V\backslash V_l} y_{i_l j}}{2} \geq p - k. \tag{31}$$

These inequalities will be called "extended F-partition inequalities." Observe that the left-hand side of inequality (31) may be fractional. In Theorem 4, we show that the right-hand side can be rounded up even if the left-hand side is fractional.

Theorem 4. *The extended F-partition inequality* (31) *is valid for* $\mathcal{P}$.

Proof. If the left-hand side of inequality (31) is integer, then $p - |F|/2$ can be rounded up to $p - k$, and hence (31) is a valid inequality. Now suppose that $\sum_{l=1}^{p} \sum_{j\in V\backslash V_l} y_{i_l j}$ is odd. Then there exists at least one $\hat{l} \in \{1, \ldots, p\}$ such that $i_{\hat{l}}$ is assigned to one hub in $V_{\hat{l}}$ and one hub in $V\backslash V_{\hat{l}}$. Because there exists a hub in set $V_{\hat{l}}$, the inequality $x(\delta(V_{\hat{l}})) \geq 2$ is satisfied. Now summing inequalities (28) for $l \neq \hat{l}$, (29), (30), and $x(\delta(V_{\hat{l}})) \geq 2$ and dividing by 2

gives $x(\delta(V_0,\ldots,V_p)\backslash F)+(\sum_{l=1:\,l\neq\hat{l}}^{p}\sum_{j\in V\backslash V_l}y_{i_lj})/2\geq p-|F|/2$. Because $(\sum_{l=1:\,l\neq\hat{l}}^{p}\sum_{j\in V\backslash V_l}y_{i_lj})/2$ is integer, the right-hand side of this inequality can be rounded up. This yields inequality $x(\delta(V_0,\ldots,V_p)\backslash F)+(\sum_{l=1:\,l\neq\hat{l}}^{p}\sum_{j\in V\backslash V_l}y_{i_lj})/2\geq p-k$, which dominates inequality (31). Hence, we can conclude that (31) is valid. □

3. Polyhedral Analysis

In this section, we conduct a polyhedral analysis. First we eliminate the variables t_i for $i\in V\backslash\{0\}$ to work with a full-dimensional polytope.

For $i\in V\backslash\{0\}$, from constraints (2), we have $t_i=1-(\sum_{j\in V\backslash\{i\}}y_{ij})/2$. Substituting this in our two-index model yields

$$z=\sum_{i\in V\backslash\{0\}}d_{ii}+\min\left\{\sum_{e\in E}c_ex_e+\sum_{(i,j)\in A}d'_{ij}y_{ij}\right\}$$

s.t.

$$2x_e+2y_{ij}+\sum_{k\in V\backslash\{j\}}y_{jk}\leq 2 \quad \forall(i,j)\in A\colon j\neq 0,\ e=\{i,j\}, \tag{32}$$

$$x_e+y_{i0}\leq 1 \quad \forall i\in V\backslash\{0\},\ e=\{i,0\}, \tag{33}$$

$$2x_e+\sum_{k\in V\backslash\{i\}}y_{ik}\leq 2 \quad \forall i\in V\backslash\{0\},\ e=\{i,0\}, \tag{34}$$

$$x(\delta(S))+\sum_{j\in V\backslash S}y_{ij}\geq 2 \quad \forall S\subseteq V\backslash\{0\},\ i\in S, \tag{35}$$

$$x_e\in\{0,1\} \quad \forall e\in E, \tag{36}$$

$$y_{ij}\in\{0,1\} \quad \forall(i,j)\in A, \tag{37}$$

where $d'_{ij}=d_{ij}-d_{ii}/2$ for $(i,j)\in A$.

To show that this formulation is equivalent to the two-index formulation of §2, we need to ensure that $\sum_{j\in V\backslash\{i\}}y_{ij}\in\{0,2\}$ and so $t_i\in\{0,1\}$ for all $i\in V\backslash\{0\}$.

Let $X=\{(x,y)\in R^{|E|+|A|}\colon (x,y)$ satisfies (32)–(37)$\}$ and $\mathcal{P}=\mathrm{conv}(X)$. Let $(x,y)\in X$ and $i\in V\backslash\{0\}$. If $x(\delta(i))>0$, then constraints (32)–(34), (36), and (37) imply that $\sum_{j\in V\backslash\{i\}}y_{ij}=0$. On the other hand, if $x(\delta(i))=0$, the cut inequality (35) for $S=\{i\}$ implies that $\sum_{j\in V\backslash\{i\}}y_{ij}\geq 2$. Because $\sum_{j\in V\backslash\{i\}}y_{ij}\leq 2$ from constraints (34), we have $\sum_{j\in V\backslash\{i\}}y_{ij}=2$. Hence, $\sum_{j\in V\backslash\{i\}}y_{ij}\in\{0,2\}$ in any feasible solution to the above model.

We assume that G is complete and $|V|\geq 7$ in the sequel. The proofs of the theorems of this section are provided as supplemental material (available at http://dx.doi.org/10.1287/ijoc.1120.0541).

Theorem 5. *$\mathcal{P}$ is full dimensional.*

Theorem 6. (i) *For $e\in E$, inequality $x_e\geq 0$ is facet defining for $\mathcal{P}$.*

(ii) *For $(i,j)\in A$, inequality $y_{ij}\geq 0$ is facet defining for $\mathcal{P}$.*

(iii) *For $(i,j)\in A$ such that $j\neq 0$ and $e=\{i,j\}$, inequality* (32) *defines a facet of $\mathcal{P}$.*

(iv) *Let $i\in V\backslash\{0\}$ and $e=\{i,0\}$. Then inequality* (34) *defines a facet of $\mathcal{P}$.*

Note that the inequalities $x_e\leq 1$ for $e\in E$ and $y_{ij}\leq 1$ for $(i,j)\in A$ are not facet defining as they are implied by constraints (32)–(34). Let $i\in V\backslash\{0\}$ and $e=\{i,0\}$. Inequality (33) is not facet defining either because all feasible solutions that satisfy $x_e+y_{i0}=1$ also satisfy $y_{i0}=\sum_{k\in V\backslash\{0,i\}}y_{ik}$. The next theorem gives necessary and sufficient conditions for the cut inequalities (35) to be facet defining for $\mathcal{P}$.

Theorem 7. *Let $S\subseteq V\backslash\{0\}$ such that $S\neq\varnothing$ and $i\in S$. The cut inequality* (35) *defines a facet of $\mathcal{P}$ if and only if the following conditions are satisfied*:

(i) $|V\backslash S|\geq 3$

(ii) $|S|\geq 4$ *or* $|S|=1$.

In §2, we derived two families of projection inequalities. These inequalities involve the variables t_i. Eliminating the t_i variables in the in-cut (19) and out-cut (20) inequalities, we obtain

$$x(\delta(S))+\sum_{k\in V\backslash\{i,j\}}y_{ik}-y_{ij}\geq 2 \tag{38}$$

and

$$x(\delta(S))+2\sum_{l\in V\backslash(S\cup\{j\})}y_{il}\geq 2, \tag{39}$$

respectively.

Theorem 8. (i) *Let $S\subseteq V\backslash\{0\}$ such that $S\neq\varnothing$, $i\in S$, and $j\in S\backslash\{i\}$. If $|S|\geq 3$ and $|V\backslash S|\geq 3$, then the in-cut inequality* (38) *defines a facet of $\mathcal{P}$.*

(ii) *Let $S\subseteq V\backslash\{0\}$ such that $S\neq\varnothing$, $i\in S$, and $j\in V\backslash S$. If $|S|\geq 4$ and $|V\backslash S|\geq 3$, then the out-cut inequality* (39) *defines a facet of $\mathcal{P}$.*

We finally give sufficient conditions for dual-homing and extended F-partition inequalities to be facet defining for $\mathcal{P}$.

Theorem 9. *For $(i,j)\in A$, the dual-homing inequality* (27) *defines a facet of $\mathcal{P}$.*

Theorem 10. *The extended F-partition inequality* (31) *defines a facet for $\mathcal{P}$ if*

(a) *$G(V_l)$ is 3-edge connected for $l=0,\ldots,p$,*

(b) *$|F\cap\delta(i_l)|\leq 1$ and $F\cap\delta(j)=\varnothing$ for $j\in V_l\backslash\{i_l\}$ for $l=1,\ldots,p$, and*

(c) *$|F\cap\delta(j)|\leq 1$ for $j\in V_0\backslash\{0\}$.*

4. Separation Algorithms

Because our mathematical model contains an exponential number of constraints and most of our valid inequalities are exponential in number, we propose a branch-and-cut algorithm for 2ECDHP. In this section,

we describe the separation algorithms that are used to identify violated inequalities.

Suppose that we are given a solution (x^*, y^*, t^*) of an LP relaxation. We define $V^* = \{i \in V\colon x^*(\delta(i)) > 0\}$ and $E^* = \{e \in E\colon x_e^* > 0\}$. By the conflict constraints, $t_i^* > 0$ for $i \in V^*\backslash\{0\}$. The graph $G^* = (V^*, E^*)$ is our support graph. This graph may be disconnected. Let $G^j = (V^j, E^j)$ for $j = 0, \ldots, r$ be the jth connected component of G^*. Without loss of generality, we assume that $0 \in V^0$. Clearly $G^0 = G^*$ if G^* is connected.

4.1. Cut Inequalities

For a fixed node $i \in V\backslash\{0\}$, the cut inequality (6) can be separated exactly by solving a minimum cut problem as explained by Labbé et al. (2004). Let $G_i^* = (V^* \cup \{i\}, E_i^*)$, where $E_i^* = E^* \cup \{\{i, j\}\colon j \in V^* \text{ and } (i, j) \in A\}$. If edge $e \in E^*$ is such that $i \notin e$, we let its capacity be equal to x_e^*. For an edge of the form $\{i, j\}$, the capacity is set to $x_{ij}^* + y_{ij}^*$. The most violated cut inequality with fixed node i can be found by solving a minimum cut problem between nodes i and 0 in graph G_i^*. If the minimum cut capacity is less than 2, then there is a violated cut inequality.

Our separation procedure is given in Algorithm 1. We have two remarks. Let $\bar{V}^* = \{i \in V\backslash V^*\colon y_{ij}^* + y_{ik}^* < 2 \ \forall \{j, k\} \in E_i\}$. First, if S and i define a violated cut inequality, then $S \cup \{k \in \bar{V}^*\backslash\{i\}\colon y_{ik}^* > 0\}$ and i also define a violated inequality with a violation that is at least as large as the one of S and i because $x^*(\delta(k)) = 0$ for all $k \in \bar{V}^*$. Second, the nodes in $V\backslash(V^* \cup \bar{V}^*) = \{i \in V\colon x^*(\delta(i)) = 0, \exists \{j, k\} \in E_i$ with $y_{ij}^* + y_{ik}^* = 2\}$ are not considered as fixed nodes in the separation algorithm. Let i be such a node and suppose that it is assigned to two nodes j and k. Also suppose that i and $S \subseteq V\backslash\{0\}$ with $i \in S$ define a violated cut inequality. For the inequality to be violated, at least one of j and k must be in S. Say j is in S; then $t_j^* = 1$. Hence, the cut inequality for S and j is also violated, and the violation is at least as large as the one for S and i.

To speed up the separation of cut inequalities, we use heuristics similar to those presented by Fouilhoux et al. (2012). For a given connected component $G^j = (V^j, E^j)$ with $j \neq 0$ and a fixed node $i \in V^j$, let $S = V^j \cup \{k \in \bar{V}^*\colon y_{ik}^* > 0\}$. Because $x^*(\delta(S)) = 0$ and $t_i^* > 0$, the cut inequality defined by S and i is violated. For $i \in \bar{V}^*$ with $\sum_{k \in V\backslash(V^j \cup \{i\})} y_{ik}^* < 2$, the cut inequality defined by i and $S = V^j \cup \{k \in \bar{V}^*\backslash\{i\}\colon y_{ik}^* > 0\} \cup \{i\}$ is also violated because $x^*(\delta(S)) = 0$, $2t_i^* + \sum_{k \in S\backslash\{i\}} y_{ik}^* = 2 - \sum_{k \in V\backslash S} y_{ik}^* > 2 - \sum_{k \in V\backslash(V^j \cup \{i\})} y_{ik}^* > 0$.

For the connected component G^0, we use the global minimum cut algorithm of Hao and Orlin (1994) with the capacity of each edge $e \in E^0$ set to x_e^*. Given a capacitated graph, the global minimum cut is defined as the minimum cut in this graph among all possible cuts between any two source and destination pairs. The algorithm proposed by Hao and Orlin (1994) is as efficient as solving a single minimum cut problem. It reports $|V^0| - 1$ cut sets, say $S_1, S_2, \ldots, S_{|V^0|-1}$, one of which is a global minimum cut. Because the underlying graph is undirected, we may assume that $0 \in V^0\backslash S_l$ for $l = 1, \ldots, |V^0| - 1$. If the capacity of the global minimum cut is at least 2, then we can conclude that there exists no violated cut inequality. Otherwise, for each $l = 1, \ldots, |V^0| - 1$, we compute the violation of the cut inequality defined by $S_l \cup \{i\}$ and i for every $i \in S_l \cup \bar{V}^*$ and add a cut with the maximum violation.

Algorithm 1 (Cut inequality separation)

$C \leftarrow \bar{V}^* \cup (V^0\backslash\{0\})$
for $j = 1$ to r **do**
  $i' \leftarrow \arg\min_{i \in V^j \cup \bar{V}^*} \sum_{k \in V\backslash(V^j \cup \{i\})} y_{ik}^*$
  if $\sum_{k \in V\backslash V^j} y_{i'k}^* < 2$ **then**
    add the violated cut inequality for
      $S = V^j \cup \{k \in \bar{V}^*\backslash\{i'\}\colon y_{i'k}^* > 0\} \cup \{i'\}$ and i'
    $C \leftarrow C\backslash\{i'\}$
  end
end
use Hao–Orlin algorithm on G^0 and find $|V^0| - 1$
  cut sets denoted by $S_1, \ldots, S_{|V^0|-1}$
for $l = 1$ to $|V^0| - 1$ **do**
  if *capacity of* $[S_l, V^0\backslash S_l]$ *is less than* 2 **then**
    $i' \leftarrow \arg\min_{i \in S_l \cup \bar{V}^*} \sum_{k \in V\backslash(S_l \cup \{i\})} y_{ik}^*$
    if *the cut inequality defined by* $S_l \cup \{i'\}$ *and* i' *is violated* **then**
      add the violated cut inequality
      $C \leftarrow C\backslash\{i'\}$
    end
  end
end
forall $i \in C$ **do**
  find a minimum cut with cutset S between i
    and 0 on G_i^*
  if *capacity of the cut defined by* S *is less than* 2 **then**
  add the violated cut inequality defined by
    $S \cup \{k \in \bar{V}^*\backslash\{i\}\colon y_{ik}^* > 0\}$ and i
end.

Finally, we use exact separation for the nodes in $\bar{V}^* \cup (V^0\backslash\{0\})$ for which no violated cut inequality has been added in the heuristic part of the algorithm.

4.2. In-Cut Inequalities

In-cut inequalities (19) are defined by a node set $S \subseteq V\backslash\{0\}$ and two fixed nodes $i \in S$ and $j \in S\backslash\{i\}$. When i and j are fixed, the right side of the inequality is fixed. Hence, to find the most violated inequality for i and j, we need to compute a set S with $i, j \in S$ such that $x^*(\delta(S))$ is minimum. Let $G_{ij}^* = (V^* \cup \{i, j\}, E^* \cup \{\{i, j\}\})$. We set the capacity of edge $e \in E^*$ other than $\{i, j\}$ to x_e^* and capacity of $\{i, j\}$ to 2 so that

i and j are in the same set. We solve the minimum cut problem between i and 0 on G^*_{ij}. If the capacity of the minimum cut is less than $2(t^*_i + y^*_{ij})$, then a violated in-cut inequality is found. This way, in-cut inequalities can be separated exactly by solving $O(|V|^2)$ minimum cut problems.

We first apply a heuristic separation based on the cut sets $S_1, \ldots, S_{|V^0|-1}$ generated by the Hao and Orlin (1994) algorithm. If the capacity of a given cut set S is less than 2, we look for the node pair $i, j \in S$ that maximizes $t^*_i + y^*_{ij}$. We apply exact separation for the nodes in $\bar{V}^* \cup (V^0 \backslash \{0\})$ for which no violated in-cut inequality is found in the heuristic phase. The separation procedure is presented in Algorithm 2.

Algorithm 2 (In-cut inequality separation)
$C \leftarrow \bar{V}^* \cup (V^0 \backslash \{0\})$
use Hao–Orlin algorithm on G^0 and find $|V^0| - 1$ cut sets denoted by $S_1, \ldots, S_{|V^0|-1}$
for $l = 1$ to $|V^0| - 1$ **do**
  if *capacity of* $[S_l, V^0 \backslash S_l]$ *is less than* 2 **then**
    $S \leftarrow S_l \cup \bar{V}^*$
    $(i', j') \leftarrow \arg\max_{i \in S,\, j \in S \backslash \{i\}} 2(t^*_i + y^*_{ij})$
    if $2(t^*_{i'} + y^*_{i'j'}) > x^*(\delta(S))$ **then**
      add the violated in-cut inequality defined by S, i', and j'
      $C \leftarrow C \backslash \{i'\}$
    end
  end
end
forall $i \in C$ **do**
  forall $j \in V^* \backslash \{i\}$ **do**
    find a minimum cut with cut set S between i and 0 on G^*_{ij}
    if *capacity of the cut defined by S is less than* $2(t^*_i + y^*_{ij})$ **then**
    add the violated in-cut inequality defined by S, i, and j
  end
end.

4.3. Out-Cut Inequalities

An out-cut inequality (20) is defined by a node set $S \subseteq V \backslash \{0\}$ and two fixed nodes $i \in S$ and $j \in V \backslash S$. A violated out-cut inequality with fixed nodes i and j can be found by solving a minimum cut problem. Let $G^*_{ij} = (V^* \cup \{i, j\}, E^*_{ij})$, where $E^*_{ij} = E^* \cup \{\{i, k\}\colon k \in V^* \backslash \{j\} \text{ and } (i, k) \in A\} \cup \{\{0, j\}\}$. Let $e \in E^*_{ij}$ be different from $\{0, j\}$. We set the capacity of e to x^*_e if $i \notin e$ or $j \in e$, and to $x^*_{iv} + 2y^*_{iv}$ if $e = \{i, v\}$ otherwise. Note that node j must be in $V \backslash S$ to define the inequality. To enforce this, we set the capacity of edge $\{0, j\}$ to 2. Solving a minimum cut problem between i and 0 on G^*_{ij} will reveal a violated out-cut inequality with fixed nodes i and j if one exists. Consequently, the out-cut inequalities can be separated exactly in polynomial time by solving $O(|V|^2)$ minimum cut problems.

Consider the cut sets $S_1, \ldots, S_{|V^0|-1}$, attained by the application of the Hao and Orlin (1994) global cut algorithm to G_0. For a given cut set S_l with capacity less than 2, the set $S = S_l \cup \bar{V}^*$ and the node pair $i \in S$ and $j \in V \backslash S$ that minimizes $\sum_{k \in V \backslash (S \cup \{j\})} y^*_{ik}$ might define a potential violated out-cut inequality.

Similar to the cut inequality and in-cut inequality separation, we consider the nodes of $\bar{V}^* \cup (V^0 \backslash \{0\})$, first apply heuristic separation, and then resort to exact separation for the remaining nodes. We provide the separation procedure in Algorithm 3.

Algorithm 3 (Out-cut inequality separation)
$C \leftarrow \bar{V}^* \cup (V^0 \backslash \{0\})$
use Hao–Orlin algorithm on G^0 and find $|V^0| - 1$ cut sets denoted by $S_1, \ldots, S_{|V^0|-1}$
for $l = 1$ to $|V^0| - 1$ **do**
  if *capacity of* $[S_l, V^0 \backslash S_l]$ *is less than* 2 **then**
    $S \leftarrow S_l \cup \bar{V}^*$
    $(i', j') \leftarrow \arg\min_{i \in S,\, j \in V \backslash S} \sum_{k \in V \backslash (S \cup \{j\})} y^*_{ik}$
    if $x^*(\delta(S)) + 2\sum_{k \in V \backslash (S \cup \{j'\})} y^*_{i'k} < 2$ **then**
      add the violated out-cut inequality defined by S, i', and j'
      $C \leftarrow C \backslash \{i'\}$
    end
  end
end
forall $i \in C$ **do**
  forall $j \in V^* \backslash \{i\}$ **do**
    find a minimum cut with cut set S between i and 0 on G^*_{ij}
    if *capacity of the cut defined by S is less than* 2 **then**
    add the violated out-cut inequality defined by S, i, and j
  end
end.

4.4. Extended F-Partition Inequalities

We use the heuristic separation algorithm of Fouilhoux et al. (2012) for the extended F-partition inequalities. These inequalities are separated if the support graph is connected. We search for fractional odd cycles that do not use node 0. Let $\{v_1, \ldots, v_p\}$ be the set of nodes inducing a fractional odd cycle. Let $V_0 = V \backslash \{v_1, \ldots, v_p\}$. Edges from $\delta(V_0)$ with values greater than 1/2 are chosen for F in such a way that $|F|$ is odd. We check the corresponding inequality for violation. If no violated inequality is found in this stage, we set the capacity of each edge e to $1 - x^*_e$, and, using the algorithm of Hao and Orlin (1994) on the support graph, we find $|V^*| - 1$ cuts. Let S be a cut set obtained by this algorithm such that $0 \in S$ and $V_0 = S \cup \bar{V}^*$. Let $V \backslash V_0 = \{v_1, \ldots, v_p\}$. Then our

partition is $(V_0, v_1, \ldots, v_p)$. We construct F as previously explained. The extended F-partition inequality defined by this partition and F is added if it is violated. The algorithm is given in Algorithm 4.

Algorithm 4 (Extended F-partition inequality separation)

repeat
  find a fractional odd cycle $v_1, \ldots, v_p$ in G^*
    such that $v_i \neq 0$ for $i = 1, \ldots, p$
  $V_0 \leftarrow V \backslash \{v_1, \ldots, v_p\}$
  construct $F \subseteq \{e \in \delta(V_0)\colon x^*_e > 0.5\}$ so that $|F|$ is odd
  add the extended F-partition inequality for
    $(V_0, v_1, \ldots, v_p)$ if violated
until *no fractional odd cycle is found*;
if *no violated extended F-partition inequality is found above* **then**
  use algorithm of Hao and Orlin on G^* with edge capacities set to $1 - x^*_e$
  foreach *cut* $[S, V^* \backslash S]$ *such that* $0 \in S$ *found in the algorithm* **do**
    $V_0 \leftarrow S \cup \bar{V}^*$
    construct $F \subseteq \{e \in \delta(V_0)\colon x^*_e > 0.5\}$ so that $|F|$ is odd
    add the extended F-partition inequality for $(V_0, v_1, \ldots, v_p)$ where $V \backslash V_0 = \{v_1, \ldots, v_p\}$ if violated
  end
end.

5. Variable Fixing

In this section, we propose rules to fix some of the variables. These rules can be grouped into two classes. The first class of fixing rules only eliminates fractional solutions. The second class, however, cuts off integer solutions provided that they are not potentially uniquely optimal. The variable fixing rules we propose are as follows:

(1) Let $(i, j) \in A$. If $d_{ij} > d_{ik}$ for every $(i, k) \in A$, then we can fix $y_{ij} = 0$.

(2) Let $\bar{z}$ be the objective function value of a feasible solution, and let $\underline{z}$ be the objective function value of the current linear program. Let $\bar{x}$ and $\bar{u}$ denote the solution and reduced cost vectors, respectively.

(a) If variable x_i is a nonbasic variable at its lower bound ($\bar{x}_i = 0$), and if $\underline{z} + \bar{u}_i > \bar{z}$, then we can fix $x_i = 0$.

(b) If variable x_i is a nonbasic variable at its upper bound ($\bar{x}_i = 1$), and if $\underline{z} - \bar{u}_i > \bar{z}$, then we can fix $x_i = 1$.

(3) Let $H = \{i \in V\colon t_i = 1\}$ be the set of nodes that are fixed to be hub nodes at some particular node of the branch-and-cut tree. If $|H| \geq 2$ then at least two of the nodes that will be hubs in the optimal solution corresponding to this subtree are known. Let $i \in V \backslash H$ and $u, v \in H$ be distinct nodes such that $d_{iu} \leq d_{iv}$. Then for every $j \in V \backslash \{i\}$ such that $d_{ij} > d_{iv}$ we can fix $y_{ij} = 0$.

(4) Let $e = \{i, j\} \in E$. If x_e is fixed to 1, then we can fix $y_{ik} = 0$ for every $k \in V \backslash \{i\}$, $y_{jk} = 0$ for every $k \in V \backslash \{j\}$, and $t_i = t_j = 1$.

(5) Let $(i, j) \in A$. If y_{ij} is fixed to 1, then we can fix $y_{jk} = 0$ for every $k \in V \backslash \{j\}$, $y_{ki} = 0$ for every $k \in V \backslash \{i, 0\}$, $x_e = 0$ for every $e \in \delta(i)$, $t_i = 0$, and $t_j = 1$.

(6) Let $i \in V \backslash \{0\}$. If t_i is fixed to 1, then we can fix $y_{ij} = 0$ for every $j \in V \backslash \{i\}$.

(7) Let $i \in V \backslash \{0\}$. If t_i is fixed to 0, then we can fix $y_{ji} = 0$ for every $j \in V \backslash \{i, 0\}$ and $x_e = 0$ for every $e \in \delta(i)$.

The first rule is based on the fact that a user will not be assigned to a hub with the highest assignment cost because there are at least three hubs in a feasible solution. The second rule is a well-known one for variable fixing and uses the reduced cost information (see, e.g., Wolsey 1998). A feasible solution is necessary to apply this rule, and clearly better feasible solutions will possibly allow more fixing. Rule 3 uses the fact that the local access network design problem, i.e., the problem of assigning the users to hubs, is trivial when the hubs are known. Users are assigned to hubs with the least assignment costs. According to rule 3, if we know the locations of at least two hubs, then we can find two open hubs that offer the least cost and conclude that a node will not be assigned to another hub with a higher assignment cost. The first three rules are based on optimality conditions; however, the remaining ones, rules 4–7, are based on the conflict constraints. Actually, they are implied by the formulation. However, as we explain next, we do not include all conflict inequalities in the subproblems. Therefore, with these rules we can fix some variables before corresponding constraints are added to the model.

The first fixing rule is applied once at the beginning of the algorithm. Rule 2 is used at each step after solving a subproblem, and we keep applying rules 3–7 until we cannot fix a new variable.

Variable fixing provides several advantages. The first is the reduction of the size of the model. The second is that we can identify some constraints that become redundant after fixing a particular variable. Let $S \subset V \backslash \{0\}$ be a node set, and $i \in S$. Clearly, S and i define a cut inequality. Note that for every $j \in S \backslash \{i\}$ there is another cut inequality defined by S and j. So there are $|S|$ cut inequalities that are induced by the same node set, and a feasible solution must satisfy all of them. Now suppose that the variable t_i is fixed to 1 during the solution algorithm. Clearly, the cut inequalities involving S are not required anymore as $x(\delta(S)) \geq 2$ becomes valid. All the cut inequalities associated with set S that were added until this point can be removed.

In the next section, we present the results of a computational experiment to investigate the effect of variable fixing.

6. Branch-and-Cut Algorithm and Computational Results

We present a branch-and-cut algorithm that uses the valid inequalities, the separation algorithms, and the variable fixing rules described in §§3–5. We implemented our algorithm in C++ using Concert Technology 29 as the framework and CPLEX 12.1 as the LP solver. The computational analysis is performed on a workstation with 2.66 GHz Xeon processor and 8 GB of memory. We use the default strategies of CPLEX in searching the branch-and-cut tree. Tailing off control is used; we branch if the improvement in the objective function value is small in 10 subsequent iterations.

A construction heuristic is used to find an initial solution at the beginning of the algorithm. We start from node 0 and apply the nearest-neighbor traveling salesman problem (TSP) heuristic to obtain a cycle that includes all nodes of the graph. Because a cycle is two-edge connected, it is a feasible solution. Throughout the branch-and-cut algorithm, we also use an LP-based heuristic. The edges are ranked in a nonincreasing order of their x_e values in the fractional solution. We start with an empty set and add the edges one by one until we obtain a two-edge-connected subgraph.

As done by Labbé et al. (2004) and Fouilhoux et al. (2012), we generate our test problems using TSP instances from TSPLIB 2.1 (Reinelt 1991). The number in the name of the instance is the number of nodes. We compute the costs of installing backbone and access links as $c_{ij} = \lceil \alpha l_{ij} \rceil$ and $d_{ij} = \lceil (10-\alpha) l_{ij} \rceil / 2$, where l_{ij} denotes the distance between nodes i and j in the TSPLIB instances and $\alpha \in \{3, 5, 7, 9\}$. We set $d_{ii} = 0$ for all $i \in V\backslash\{0\}$. We note here that as α decreases, access link costs increase, whereas backbone link costs decrease and 2ECDHP gets closer to the two-edge-connected subgraph problem.

6.1. Results for Small Instances

First, we solve our original formulation using a branch-and-cut algorithm and report the results for small size problems (up to 105 nodes). Because our formulation has a large number of constraints, we start our branch-and-cut algorithm by solving the following relaxed linear program:

$$\min \left\{ \sum_{e\in E} c_e x_e + \sum_{(i,j)\in A} d_{ij} y_{ij} + \sum_{i\in V\backslash\{0\}} d_{i0} t_i \right\}$$

$$\text{s.t. } 2t_i + \sum_{j\in V\backslash\{i\}} y_{ij} = 2 \quad \forall i \in V\backslash\{0\},$$

$$x_e + y_{i0} \le 1 \quad \forall i \in V\backslash\{0\},\ e = \{i, 0\},$$

$$x_e \le t_i \quad \forall i \in V\backslash\{0\},\ e = \{i, 0\},$$

$$x(\delta(i)) \ge 2t_i \quad \forall i \in V\backslash\{0\},$$

$$x(\delta(0)) \ge 2,$$

$$x(\delta(\{0, i\})) \ge 2 \quad \forall i \in V\backslash\{0\},$$

$$0 \le x_e \le 1 \quad \forall e \in E,$$

$$0 \le y_{ij} \le 1 \quad \forall (i, j) \in A,$$

$$0 \le t_i \le 1 \quad \forall (i) \in V\backslash\{0\}.$$

A solution $(\bar{x}, \bar{y}, \bar{t})$ of this initial subproblem is feasible for 2ECDHP if it satisfies the relaxed conflict constraints (3), cut inequalities (6), and the integrality constraints (7)–(9). The cut inequalities are separated as explained in §4. The conflict constraints are separated by enumeration.

The results are reported in Table 1. Here, the first two columns give the name of the instance and the α value. We report the percentage root gap, the number of branch-and-cut nodes explored, and the CPU time in seconds in the columns "Gap," "Nodes," and "CPU," respectively. Finally, in columns "No conflict" and "No cut," we report the number of conflict constraints and the number of cut inequalities that are added in the course of the algorithm.

We observe that the LP relaxations give strong bounds, and not many nodes are enumerated in the branch-and-cut tree. The largest gap is 1.53%, and 402 nodes are enumerated for the corresponding instances. Two instances are solved at the root node without branching. All problems are solved to optimality in less than five minutes.

We also observe that even though the root gaps are higher with smaller α values, the CPU times tend to increase as α increases. More conflict constraints are added for larger α values. This is expected because as α increases, fewer nodes are chosen as hubs.

These results are further improved by incorporating the variable fixing scheme explained in §5. The results are reported in the last two columns of Table 1. Here the "Fix rate" refers to the percentage of the number of variables fixed, and "CPU imp" refers to the percentage improvement in the CPU time due to variable fixing. We observe that variable fixing has improved the computation times for all instances considered. The smallest improvement is 1.4%, whereas the largest improvement is 95.1%. In general, more variables are fixed for small α values. Also, the effect of fixing variables is higher for smaller values of α. For the instance KroA100, 71.6% of variables are fixed, and this results in an improvement of 86.3% in the CPU time when α is 3. On the other hand, when α is 9, the fix rate is 93.9% but the improvement in the CPU time is only 28.7%. The average fix rate is 50.3%, and the average improvement in the CPU times is

Table 1 Results of the Branch-and-Cut Algorithm with Conflict Constraints and Cut Inequalities for Small Instances

Instance	α	Gap	Nodes	CPU	No conflict	No cut	Fix rate	CPU imp
eil101	3	0.24	5	6	10	389	87.6	89.1
eil101	5	0.02	2	22	156	1,167	99.5	95.1
eil101	7	0.72	103	110	597	2,865	3.2	1.4
eil101	9	0.06	3	79	1,755	1,857	87.1	52.1
gr96	3	1.13	128	14	51	1,247	79.2	90.8
gr96	5	0.85	82	27	177	2,475	50.8	81.7
gr96	7	0.43	13	27	455	1,282	10.7	28.5
gr96	9	0.27	11	159	1,507	2,106	68.1	40.9
kroA100	3	1.53	402	16	45	869	71.6	86.3
kroA100	5	0.87	79	6	164	429	76.5	72.9
kroA100	7	0.26	18	12	445	748	19.2	13.6
kroA100	9	0.01	2	61	1,525	1,800	93.9	28.7
kroB100	3	1.02	63	11	32	697	62.5	81.5
kroB100	5	0.87	243	61	173	3,613	35.4	67.6
kroB100	7	0.09	8	25	466	1,308	10.3	34.1
kroB100	9	0.28	19	232	1,657	2,003	17.9	42.3
kroC100	3	1.33	92	15	52	1,162	69.2	77.6
kroC100	5	1.09	76	26	166	1,878	37.7	73.5
kroC100	7	0.28	16	21	467	1,334	33.3	43.4
kroC100	9	0.00	1	50	1,466	1,671	10.8	30.5
kroD100	3	0.72	18	15	38	926	73.8	88.8
kroD100	5	0.09	4	13	161	968	88.7	87.0
kroD100	7	0.21	15	26	469	1,312	3.1	6.5
kroD100	9	0.26	11	142	1,514	2,194	24.5	2.4
kroE100	3	0.56	39	5	48	791	89.3	87.9
kroE100	5	0.96	76	73	174	5,867	47.0	76.6
kroE100	7	0.08	3	16	466	848	17.2	42.4
kroE100	9	0.92	41	262	1,552	2,897	25.8	6.4
lin105	3	0.06	2	11	56	777	93.0	90.3
lin105	5	0.09	3	5	160	471	61.7	71.5
lin105	7	0.00	1	25	447	1,743	99.8	76.4
lin105	9	0.05	2	109	1,666	2,334	19.1	9.0
rat99	3	0.41	13	8	12	536	74.5	90.7
rat99	5	0.21	27	23	169	1,525	32.6	81.8
rat99	7	0.11	11	38	539	1,894	22.4	43.2
rat99	9	0.20	10	206	1,702	2,646	13.9	18.5

55.9%. Based on these results, we decided to use variable fixing for larger instances.

6.2. Results for Large Instances and the Effect of Valid Inequalities

In this experiment, we investigate the effect of adding valid inequalities. Here, we use larger instances; we solve 20 instances where the number of nodes ranges from 150 to 198. In Table 2, we report the results obtained by solving the original formulation using variable fixing.

All instances are solved to optimality in less than 2.5 hours. The largest gap is 1.5%. As in the case of small instances, here we also observe that in most cases the CPU times and the number of conflict constraints added increase and the fix rate decreases as α increases.

In Table 3, we report the results with valid inequalities. In our preliminary experiment, we observed that the in-cut inequalities are not effective in the solution of our problem. Therefore we do not include the in-cut inequalities in this analysis. In Table 3 column "Dual-homing ineqs," we report the results obtained when dual-homing inequalities are used together with the conflict constraints and the cut-inequalities. The results under the heading "Out-cut ineqs" are obtained when out-cut inequalities are also incorporated over dual-homing inequalities. Finally, we also use the extended F-partition inequalities together with dual-homing and out-cut inequalities and give the results in the columns under "Extended F-partition ineqs." The best values are shown in bold.

The dual-homing inequalities are separated using enumeration, and the out-cut and extended F-partition inequalities are separated using the algorithms given in §4. The separation procedures for different classes of inequalities are performed in the following order: conflict, cut, dual homing, out-cut,

Table 2 Results of the Branch-and-Cut Algorithm with Conflict Constraints and Cut Inequalities for Larger Instances

Instance	α	Gap	Nodes	CPU	No conflict	No cut	Fix rate
kroA150	3	0.85	494	10	59	1,701	79.2
kroA150	5	0.42	85	19	262	2,004	46.4
kroA150	7	0.30	18	72	694	1,979	13.9
kroA150	9	0.07	7	541	2,193	3,693	22.4
kroB150	3	1.26	1,061	37	90	2,362	53.4
kroB150	5	1.50	17,888	1,862	322	8,213	30.0
kroB150	7	0.13	4	36	742	3,331	48.3
kroB150	9	0.19	18	712	2,347	4,353	17.5
pr152	3	0.64	104	12	82	2,132	84.8
pr152	5	0.71	1,084	144	298	3,995	40.1
pr152	7	1.07	22,550	2,778	793	4,132	9.5
pr152	9	0.47	924	6,774	2,330	6,651	6.3
u159	3	0.37	51	5	50	1,195	79.9
u159	5	0.34	62	37	259	8,130	70.9
u159	7	0.22	19	92	770	3,348	16.3
u159	9	0.35	184	1,553	2,493	5,622	20.8
d198	3	0.43	554	154	44	4,835	47.5
d198	5	0.10	20	212	343	10,887	37.2
d198	7	0.22	503	1,311	1,080	9,589	18.9
d198	9	0.06	54	8,522	3,320	15,857	2.7

and extended F-partition. At most, 200 cuts are added at an iteration. Out-cut inequalities are separated if no conflict constraints or cut inequalities are added.

We observe that the dual-homing inequalities are not very useful when $\alpha = 3$. We recall that these are the instances where all nodes become hubs at optimality. For larger α values, the LP bounds are improved significantly, and one instance is solved to optimality at the root node. In general, the number of explored branch-and-cut nodes also decreases. The results are mixed for computation times; we observe significant improvements for 10 instances, whereas in 8 instances the CPU times increase. In the remaining two instances, the changes are minimal.

After adding the out-cut inequalities, the CPU times improved significantly for six instances, whereas in three instances we obtained worse results. In the remaining 11 instances the differences are not significant. No out-cut inequalities are added for the instances with α equal to 3 and 5. The out-cut inequalities are useful for improving the LP relaxation values especially when $\alpha = 9$. Three more instances are solved to optimality without branching by using the out-cut inequalities. The number of branch-and-cut nodes also decreased in general; however, in two instances this number increased.

Finally, we observe that the extended F-partition inequalities significantly improve the LP relaxation bounds and the solution times. Although the number of branch-and-cut nodes also decreased in most of the instances, there are significant increases in some instances. Two more instances are solved to optimality without branching.

We note that the time spent in separation is insignificant compared to the total computation time. We observed that more dual-homing inequalities are added for larger α values, especially for $\alpha = 7$. The number of out-cut inequalities tends to increase, whereas the number of extended F-partition inequalities tends to decrease as α increases.

Overall, in all instances except one, the solution times reduced, showing that the addition of valid

Table 3 Effect of Adding Valid Inequalities for Larger Instances

		Dual-homing ineqs			Out-cut ineqs			Extended F-partition ineqs		
Instance	α	Gap	Nodes	CPU	Gap	Nodes	CPU	Gap	Nodes	CPU
kroA150	3	0.85	489	10	0.85	489	10	**0.51**	**74**	**8**
kroA150	5	0.42	64	9	0.42	64	9	**0.00**	**2**	**8**
kroA150	7	0.16	**2**	**49**	0.15	8	54	**0.08**	**2**	54
kroA150	9	0.03	2	495	**0.00**	**1**	**408**	**0.00**	**1**	**408**
kroB150	3	1.26	1,002	37	1.26	1,002	38	**0.82**	**95**	**8**
kroB150	5	1.37	17,045	2,822	1.37	17,045	2,789	**0.83**	**797**	**232**
kroB150	7	**0.00**	**1**	22	**0.00**	**1**	22	**0.00**	**1**	**15**
kroB150	9	0.14	12	910	**0.00**	**1**	**458**	**0.00**	**1**	573
pr152	3	0.64	**116**	12	0.64	**116**	13	**0.60**	164	**11**
pr152	5	0.61	**242**	71	0.61	**242**	71	**0.52**	265	**46**
pr152	7	0.93	4,431	1,923	0.90	7,061	2,863	**0.81**	**1,411**	**1,137**
pr152	9	0.40	478	**4,593**	**0.11**	7	6,429	**0.11**	**4**	6,219
u159	3	0.37	46	8	0.37	46	8	**0.20**	**3**	**4**
u159	5	0.31	37	56	0.31	37	55	**0.00**	**1**	**14**
u159	7	0.10	9	59	0.04	**2**	**45**	**0.00**	3	51
u159	9	0.34	161	1,787	**0.11**	35	1,413	**0.11**	**15**	**1,219**
d198	3	0.43	470	118	0.43	470	120	**0.28**	**63**	**106**
d198	5	0.09	20	1,758	0.09	20	1,759	**0.00**	**1**	**706**
d198	7	0.10	205	1,434	0.09	**59**	**785**	**0.08**	88	829
d198	9	0.02	15	5,629	**0.00**	**1**	**3,510**	**0.00**	**1**	3,622

inequalities improves the performance of the branch-and-cut algorithm. In instance d198 with α equal to 5, the CPU time increased by 233%, but the problem is solved at the root node. Overall, six problems are solved without branching when valid inequalities are used, and significant savings in the computation times are obtained for the other problems. In particular, the instance with the longest computation time, d198 with $\alpha = 9$, is solved in one hour instead of more than two hours. Another difficult instance, pr152 with $\alpha = 9$, still requires 6,219 seconds, and the improvement in the CPU time is approximately 8%. However, the number of nodes decreases from 924 to 4 after the addition of valid inequalities. The average improvements in the number of nodes and in the CPU times are 82.9% and 27.8%, respectively.

7. Conclusion

We analyzed a hierarchical network design problem with survivability requirements in both levels of the design. The resulting design has an access network that meets the two-edge-connected backbone network in a dual-homing manner and thus the network is wholly protected against single link failures. This work extends the literature because it tackles survivability and design in both levels of the network in an exact manner. To this end, we proposed two formulations and compared them in terms of the LP relaxation bounds and the difficulty of the separation problems associated with their cut inequalities. We performed a polyhedral analysis based on the small size formulation and proposed exact and heuristic separation algorithms for the valid inequalities. To improve the performance of the branch-and-cut algorithm, we developed several variable fixing rules. The effect of the valid inequalities and the variable fixing rules were tested on a range of problem instances involving nearly 200 nodes. The computational analysis depicted that the valid inequalities and the variable fixing rules have significantly improved the performance of the proposed algorithm.

In our study, the terminal nodes can communicate with each other only through direct connections with the hub nodes. In some applications, it is feasible to communicate through other terminal nodes as well. The ring/ring designs are examples of such networks. One potential future research direction could be the study of access networks different from star topologies. An interesting hierarchical network that generalizes ring/ring designs is a two-edge-connected/two-edge-connected network.

In some applications, technological limitations may bound the maximum number of hops a signal can traverse in the network. Thus, limiting the diameter of the resulting networks even in case edge failures could lead to more realistic and applicable designs. A study that has a similar flavor is that of Baldacci et al. (2007). They consider the design of a collection of rings satisfying certain limitations, one of which is an upper bound on the number of hub and terminal nodes being served by each ring.

Another interesting line of extension may be to consider capacity installations for demand routing in the existing survivable networks.

Finally, a further extension line of research could be the consideration of Steiner nodes in the designs.

The models as well as the valid inequalities developed and used in this paper can potentially be extended to answer the noted technological restrictions.

Supplemental Material

Supplemental material to this paper is available at http://dx.doi.org/10.1287/ijoc.1120.0541.

Acknowledgments

The authors greatly appreciate the contribution of two anonymous referees. This research was supported by TUBITAK [Project 107M247].

References

Balakrishnan A, Magnanti TL, Mirchandani P (1998) Designing hierarchical survivable networks. *Oper. Res.* 46(1):116–136.

Balakrishnan A, Mirchandani P, Natarajan HP (2009) Connectivity upgrade models for survivable network design. *Oper. Res.* 57(1):170–186.

Baldacci R, Dell'Amico M, Gonzalez JS (2007) The capacitated *m*-ring-star problem. *Oper. Res.* 55(6):1142–1162.

Din DR, Tseng SS (2002) A genetic algorithm for solving dual-homing cell assignment problem of the two-level wireless ATM network. *Comput. Comm.* 25(17):1536–1547.

Fouilhoux P, Karasan OE, Mahjoub AR, Ozkok O, Yaman H (2012) Survivability in hierarchical telecommunications networks. *Networks* 59(1):37–58.

Garey MR, Johnson DS (1979) *Computers and Intractability: A Guide to the Theory of NP-Completeness* (W. H. Freeman, New York).

Gourdin E, Labbé M, Yaman H (2002) Telecommunication and location. Drezner Z, Hamacher HW, eds. *Facility Location: Applications and Theory* (Springer, Berlin), 275–305.

Grötschel M, Monma CL, Stoer M (1995) Design of survivable communications networks. Ball MO, Magnanti TL, Monma CL, Nemhauser GL, eds. *Network Models* (North-Holland, Amsterdam), 617–671.

Hao J, Orlin JB (1994) A faster algorithm for finding the minimum cut in a directed graph. *J. Algorithms* 17(3):424–446.

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

Kerivin H, Mahjoub AR (2005b) On survivable network polyhedra. *Discrete Math.* 290(2–3):183–210.

Klincewicz JG (1998) Hub location in backbone/tributary network design: A review. *Location Sci.* 6(1–4):307–335.

Kraushaar J (1999) Fiber deployment update-end of year 1998. Report, Industrial Analysis Division, Common Carrier Bureau, Federal Communications Commission, Washington, DC.

Labbé M, Laporte G, Martin IR, Gonzalez JJS (2004) The ring star problem: Polyhedral analysis and exact algorithm. *Networks* 43(3):177–189.

Lee CY, Koh SJ (1997) A design of the minimum cost ring-chain network with dual-homing survivability: A tabu search approach. *Comp. Oper. Res.* 24(9):883–897.

Magnanti TL, Raghavan S (2005) Strong formulations for network design problems with connectivity requirements. *Networks* 45(2):61–79.

Mahjoub AR (1994) Two-edge connected spanning subgraphs and polyhedra. *Math. Prog.* 64(1–3):199–208.

Mahjoub AR, Pesneau P (2008) On the Steiner 2-edge connected subgraph polytope. *RAIRO Oper. Res.* 42(3):259–283.

Proestaki A, Sinclair MC (2000) Design and dimensioning of dual-homing hierarchical multi-ring networks. *IEEE Proc. Comm.* 147(2):96–104.

Reinelt G (1991) TSPLIB—A traveling salesman problem library. *ORSA J. Comp.* 3(4):376–384.

Schrijver A (1986) *Theory of Linear and Integer Programming* (Wiley, Hoboken, NJ).

Shi JJ, Fonseka JP (1997) Analysis and design of survivable communications networks. *IEE Proc. Comm.* 144(5):322–330.

Stoer M (1992) Design of Survivable Networks. Lecture Notes in Mathematics, Vol. 1531 (Springer-Verlag, Berlin).

Thomadsen T, Stidsen T (2005) Hierarchical ring network design using branch-and-price. *Telecomm. Sys.* 29:61–76.

Vandenbussche D, Nemhauser GL (2005) The 2-edge-connected subgraph polyhedron. *J. Combin. Optim.* 9(4):357–379.

Wolsey LA (1998) *Integer Programming* (Wiley, Hoboken, NJ).

Zhang J, Mukherjee B (2004) A review of fault management in WDM mesh networks: Basic concepts and research challenges. *IEEE Network* 18(2):41–48.