The Two-Edge Connected Hop-Constrained Network Design Problem: Valid Inequalities and Branch-and-Cut

David Huygens and Martine Labbé
Département d'Informatique, Université Libre de Bruxelles, Boulevard du Triomphe, CP 210/01, 1050 Bruxelles, Belgium; e-mail: dhuygens@ulb.ac.be; mlabbe@ulb.ac.be

A. Ridha Mahjoub
LIMOS, CNRS, UMR 6158, Université Blaise Pascal, Clermont-Ferrand II, Complexe Scientifique des Cézeaux, 63177 Aubière Cedex, France; e-mail: Ridha.Mahjoub@math.univ-bpclermont.fr

Pierre Pesneau*
IAG/POMS, Université Catholique de Louvain, Place des Doyens, 1, 1348 Louvain-la-Neuve, Belgium; e-mail: pierre.pesneau@math.u-bordeaux1.fr

This article deals with the Two-edge connected Hop-constrained Network Design Problem (or THNDP for short). Given a weighted graph $G = (N, E)$, an integer $L \geq 2$, and a subset of pairs of nodes D, the problem consists of finding the minimum cost subgraph in G containing at least two edge-disjoint paths of at most L hops between all the pairs in D. First, we show that the THNDP is strongly *NP*-hard even when the demands in D are rooted at some node s and the costs are unitary. However, if the graph is complete, we prove that the problem in this case can be solved in polynomial time. We give an integer programming formulation of the problem in the space of the design variables when $L = 2, 3$. Then we study the associated polytope. In particular, we consider the case where all the pairs of nodes of D are rooted at a node s. We give several classes of valid inequalities along with necessary and/or sufficient conditions for these inequalities to be facet defining. We also derive separation routines for these inequalities. We finally develop a branch-and-cut algorithm based on these results and discuss some computational results for $L = 2, 3$. © 2006 Wiley Periodicals, Inc. NETWORKS, Vol. 49(1), 116–133 2007

Keywords: network design; edge connectivity; hop-constraints; NP-hardness; valid inequalities; facets; branch-and-cut

1. INTRODUCTION

Let $G = (N, E)$ be a graph. Let $D \subseteq N \times N$ be a set of pairs of nodes, called *demands*. If the pair $\{s, t\}$ is a demand in D, we will call s and t *demand nodes* or *terminal nodes*. In particular, when several demands $\{s, t_1\}, \ldots, \{s, t_d\}$ are rooted at the same node s, we will speak of s as a *source node* and of the t_is as the *destination nodes* of s. The nodes in N that do not belong to any demand of D will be called *Steiner nodes*.

Let $L \geq 2$ be a fixed integer. If s, t are two nodes of N, an *L-st-path* in G is a path between s and t of length at most L, where the length of a path is its number of edges (or *hops*). Given a function $c : E \to \mathbb{R}$, which associates a cost $c(e)$ to each edge $e \in E$, the *Two-edge connected Hop-constrained Network Design Problem* (THNDP) is to find a minimum cost subgraph such that, between each demand $\{s, t\} \in D$, there exist at least two edge-disjoint *L-st*-paths. If all the demands in D are rooted at some node s, then we will speak about the *rooted* THNDP.

In practice, the THNDP addresses the actual need of designing telecommunication networks of high survivability. Indeed, basic requirements, like 2-edge connectivity, for example, are often not sufficient to guarantee an effective survivable network. Further technical constraints must be added to guarantee the quality of the rerouting paths in the network. Actually, there are two types of rerouting strategies in telecommunications: the local, and the end-to-end reroutings. In the second one, in case of a link failure, the traffic is rerouted between its endnodes along an alternative path. This one has to be short enough so that this procedure can be accomplished in a minimum time. This strategy is used in the ATM and SDH/SONET networks and the Internet,

Received April 2005; accepted March 2006
*Present address: MAB, Université de Bordeaux 1, 351, Cours de la libération, 33405 Talence Cedex, France
Correspondence to: A. Ridha Mahjoub; e-mail: Ridha.Mahjoub@math.univ-bpclermont.fr
Research partially supported by FRIA fellowship and a cooperation agreement CGRI-FNRS-CNRS, projet 03/005.
DOI 10.1002/net.20146
Published online 12 October 2006 in Wiley InterScience (www.interscience.wiley.com).

for example. In such situations, hop-constrained paths offer exactly the reliability required.

In this article, we study the THNDP from a polyhedral point of view. We first show that the rooted THNDP is strongly NP-hard even for $L = 2$. We give an integer programming formulation for $L = 2, 3$. We introduce several classes of valid inequalities along with necessary conditions and sufficient conditions for these inequalities to be facet defining. We also discuss separation routines for these classes of inequalities. Using this, we propose a branch-and-cut algorithm for the THNDP when $L = 2, 3$, and present some computational results.

Despite its practical interest, we are not aware of previous work on the THNDP. However, the related problem where node-disjunction is required between the paths linking the demands has already been studied in [21]. The authors give three compact formulations for the problem and discuss computational results. Moreover, the particular case where the set of demands D is of cardinality 1 has been investigated by Huygens et al. [26] for two edge-disjoint L-st-paths with $L = 2, 3$, and by Dahl et al. [10] for k-edge-disjoint 2-st-paths with $k \geq 2$. For both problems, a complete and minimal linear description of the corresponding polytope is given.

Many related problems have been however investigated. In [7], Dahl considers the hop-constrained path problem, that is, the problem of finding between two distinguished nodes s and t a minimum cost path with no more than L edges when L is fixed. He gives a complete description of the dominant of the associated polytope when $L \leq 3$. Dahl and Gouveia [9] consider the directed hop-constrained path problem. They describe valid inequalities and characterize the associated polytope when $L \leq 3$. In [5], Coullard et al. investigate the structure of the polyhedron associated with the st-walks of length L of a graph, where a walk is a path that may go through the same node more than once. Dahl et al. [8] also consider the hop-constrained walk polytope.

The closely related problem of finding a minimum cost spanning tree with hop-constraints is considered in [16, 17, 22]. Here, the hop-constraints limit to a positive integer L the number of links between the root and any destination node in the network. Dahl [6] studies the problem for $L = 2$ from a polyhedral point of view, and gives a complete description of the associated polytope when the graph is a wheel. Gouveia and Janssen [18] discuss a generalization of the previous problem where two different cable technologies with different reliabilities are available. In [11], Dahl and Johannessen consider an extension of the shortest hop-constrained path problem with $L = 2$ called the 2-path network design problem. Given a set of pairs of terminal nodes, this problem consists of finding a minimum cost subgraph connecting each pair of terminal nodes by at least one path of length at most 2. The authors discuss a polyhedral approach for the problem and devise a cutting plane algorithm. Gouveia and Magnanti [19] consider the problem that consists of finding a minimum spanning tree such that the number of edges in the tree between any pair of nodes is limited to a given bound (diameter). In [20], Gouveia, et al. provide an alternative modeling approach for that problem when the tree diameter is odd. Further hop-constrained survivable network design problems are studied in [1–3, 29, 30, 32]. A survey of survivability, which also includes a section with hop-constraints, can be found in [28].

We assume familiarity with graphs and polyhedra. For specific details, the reader is referred to [4] and [33]. The graphs that we consider are finite, undirected, loopless, and may have multiple edges. A graph is denoted by $G = (N, E)$, where N is the *node set* and E is the *edge set*. If $V, W \subset N$, $[V, W]$ is the set of edges having one endnode in V and the other one in W. Note that we will write $[v, w]$ instead of $[\{v\}, \{w\}]$. Given two nodes u and v such that $[u, v]$ is not empty, we will denote by uv an arbitrary edge of $[u, v]$. If $W \subset N$ is a node subset of G, then the set of edges that have only one node in W is called a *cut* and denoted by $\delta(W)$. We will write $\delta(v)$ for $\delta(\{v\})$. A cut $\delta(W)$ such that $s \in W$ and $t \in N \backslash W$ will be called an *st-cut*. For a partition $\Pi = (V_0, V_1, \ldots, V_k)$ of N, the associated *multicut* in G, denoted by $\Delta_\Pi(G) = \delta(V_0, V_1, \ldots, V_k)$, is the set of edges having their endnodes in two different subsets. For a partition $\Pi = (V_0, V_1, \ldots, V_k)$ of N, we will denote by $E_\Pi^{p,q} = \bigcup_{i=p,p+1,\ldots,q}[V_i, V_{i+1}]$ the set of edges between the consecutive subsets $V_p, V_{p+1}, \ldots, V_{q+1}$ of Π. A *path* P of G is an alternating sequence of nodes and edges $(u_1, e_1, u_2, e_2, \ldots, u_{q-1}, e_{q-1}, u_q)$, where $e_i \in [u_i, u_{i+1}]$ for $i = 1, \ldots, q-1$. We will denote a path P by either its node sequence $(u_1, \ldots, u_q)$ or its edge sequence $(e_1, \ldots, e_{q-1})$. (The node sequence notation of a path will be used only if the edges of the path can be taken arbitrarily.) A *cycle* in G is a path whose endnodes coincide, that is, $u_0 = u_q$. It will be denoted in the same way as a path. We will define the *length* of a cycle as its number of edges. Finally, a *chord* is an edge between any two nonadjacent nodes of a path (cycle).

Given a graph $G = (N, E)$ and an edge subset $F \subseteq E$, the $0-1$ vector $x^F \in \mathbb{R}^E$, such that $x^F(e) = 1$ if $e \in F$ and $x^F(e) = 0$ otherwise, is called the *incidence vector* of F. Given a set of demands $D \subseteq N \times N$ and an integer $L \geq 2$, the convex hull of the incidence vectors of the solutions to the THNDP on G, denoted by $P_G(D, L)$, will be called the THNDP *polytope*. Given a vector $w \in \mathbb{R}^E$ and an edge subset $F \subseteq E$, we let $w(F) = \sum_{e \in F} w(e)$.

The article is organized as follows. In the next section, we investigate the complexity of the rooted THNDP. In Section 3, we give an integer programming formulation for the problem when $L = 2, 3$. In Section 4, we present some new classes of valid inequalities. Necessary and sufficient conditions for these inequalities to be facet defining are discussed in Section 5. In Section 6, we study the separation of these inequalities. In Section 7, we derive a branch-and-cut algorithm and present our experimental results. Finally, in Section 8, we give some concluding remarks.

2. COMPLEXITY OF THE ROOTED THNDP

It is easy to see that the rooted THNDP is a generalization of the two-edge connected subgraph problem. Given a

graph with weights on its edges, this latter problem consists of finding a minimum weight subgraph such that, between every pair of nodes, there exist at least two edge-disjoint paths. So the two-edge connected subgraph problem is nothing but the rooted THNDP for $D = \{s\} \times N$ and L sufficiently large. It is well known that this former problem is *NP*-hard, even when the weights are all equal to 1. This implies that the rooted THNDP is also *NP*-hard in this case. Moreover, because the input is then of polynomial length, we have that the rooted THNDP is strongly *NP*-hard.

In what follows, we are going to show that the rooted THNDP remains strongly *NP*-hard for every fixed $L \geq 2$.

Theorem 1. *For L fixed, the rooted THNDP is strongly NP-hard.*

Proof. We will show that any instance of the minimum cardinality dominating set problem can be polynomially transformed to an instance of the rooted THNDP with $L \geq 2$. As the former problem is *NP*-hard (see [13]), this will prove that the latter one is also *NP*-hard. Moreover, as the input data of the corresponding THNDP instance will always be of polynomial length, we obtain that this problem is strongly *NP*-hard.

Consider an instance $G = (N, E)$ of the dominating set problem. This problem consists of finding a subset N' of N of minimal cardinality such that every node in $N \backslash N'$ is adjacent to at least one node of N'.

Let us construct an instance of the rooted THNDP for a fixed value of L in the following way. We create a source node s, and two copies, N_1 and N_2, of N. Construct an edge sv for each node $v \in N_1 \cup N_2$, and an edge v_1v_2 between $v_1 \in N_1$ and $v_2 \in N_2$ if the corresponding nodes in N are either the same or adjacent to each other in the original graph G. Finally, insert $L - 2$ nodes of degree 2 on each edge between either s and N_2, or N_1 and N_2. Observe that the latter operation transforms these edges into paths of length $L - 1$, which we denote in the following way. For each $u \in \{s\} \cup N_1$ and $v \in N_2$, let us call P_{uv} the $(L-1)$-uv-path. Let us denote by $\overline{G} = (\overline{N}, \overline{E})$ the auxiliary graph. See Figure 1 for an illustration for $L = 3$. We consider the rooted THNDP on $\overline{G}$, with unit costs on all edges, and the set N_2 as destination nodes relative to s.

Let S^* be an optimal solution to the rooted THNDP in $\overline{G}$ with respect to s and N_2. We are going to show that an optimal solution of the THNDP in $\overline{G}$ corresponds to a minimum cardinality dominating set in G and conversely. Let $\Gamma_1 = \bigcup_{v \in N_2} P_{sv}$ and $\Gamma_2 = \bigcup_{u \in N_1, v \in N_2} P_{uv}$.

Claim 1. There exists an optimal solution S^* of the rooted THNDP in $\overline{G}$ such that

(i) S^* contains all the paths of Γ_1, and
(ii) S^* contains exactly $|N|$ paths from Γ_2.

Proof. First we remark that any feasible solution to the THNDP will contain at least $2|N|$ paths from $\Gamma_1 \cup \Gamma_2$, because such a path used for an L-sv_2-path, with $v_2 \in N_2$, cannot be used for an L-sv_2'-path, where $v_2' \in N_2 \backslash \{v_2\}$. Moreover, an optimal solution S^* for the THNDP in $\overline{G}$ can be considered so that it contains the $|N|$ paths of Γ_1. Indeed, if a path of Γ_1, say P_{sv_2} with $v_2 \in N_2$, is not taken in S^*, then there must exist two L-sv_2-paths in the solution, going through N_1. Therefore, the optimal solution contains two paths from Γ_2 incident to v_2. Now, by replacing one of these two by P_{sv_2}, we get a solution with the same cardinality, hence being still optimal, and containing P_{sv_2}. This shows (i).

Finally, by (i) and the first remark, we have that S^* must contain at least $|N|$ paths in Γ_2. As for each node $v_2 \in N_2$, only one path from Γ_2 is needed in S^*, it follows that S^* contains exactly $|N|$ paths from Γ_2, which establishes (ii). □

By Claim 1, all the paths of Γ_1 can be considered in S^* and S^* then contains exactly $|N|$ paths from Γ_2. As a consequence, determining such an optimal solution S^* reduces to finding a minimum number of edges between s and N_1 such that, for each node $v_2 \in N_2$, there is a path of length exactly L

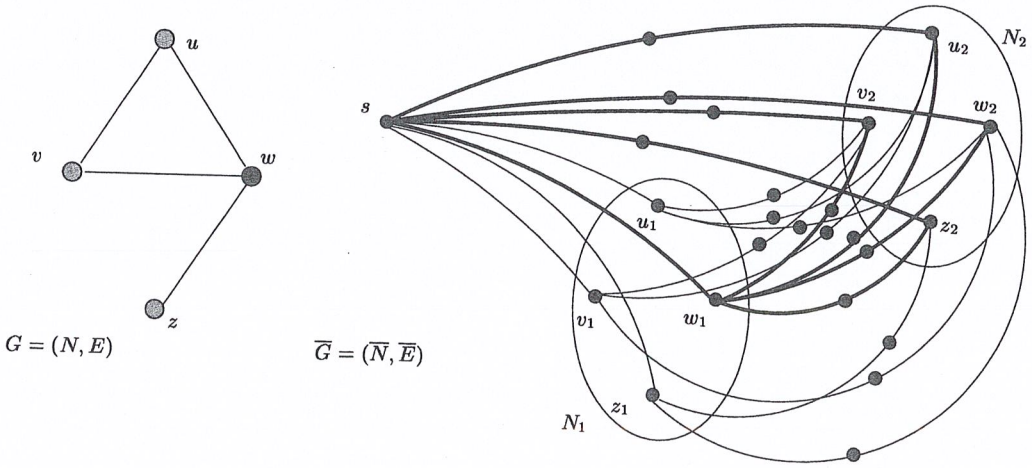

FIG. 1. An instance of the dominating set problem (graph G) and the corresponding instance of the rooted THNDP with $L = 3$ (graph $\overline{G}$).

between s and v_2, going through N_1 and using one of those $|N|$ paths of Γ_2. Because every node of N_1 is adjacent to s, this is equivalent to finding a minimum cardinality subset N_1' of N_1 that *covers* all the nodes of N_2, in the sense that all the nodes of N_2 are reachable by the paths of Γ_2 going out of N_1'. See, for instance, Figure 1, where S^* is represented by the bold edges.

Claim 2. A node subset N' of N is a dominating set of G if and only if the corresponding subset N_1' of N_1 covers all the nodes of N_2 in $\overline{G}$.

Proof. Suppose that we have a dominating set N' of G. By definition, each node of $N \backslash N'$ is adjacent to at least one node of N'. Let N_1' and N_2' be the sets of nodes corresponding to N' in N_1 and N_2, respectively. In $\overline{G}$, we have that each node in $N_2 \backslash N_2'$ is adjacent to at least one node of $N_1' \subseteq N_1$ by a path of Γ_2. Moreover, by construction, each node of N_2' is adjacent to its copy in N_1'. Hence, we have that N_2 is covered by the paths of Γ_2 going out of N_1'. The other way is similar. □

By Claim 2, and the above developments, the minimum cardinality subset $N_1' \subseteq N_1$ that covers N_2 corresponds to a minimum cardinality dominating set in G (e.g., the black node w in Fig. 1). Moreover, it is clear that any optimal solution in $\overline{G}$ can be transformed into a solution verifying the conditions of Claim 1 in polynomial time. ■

For rooted demands and unitary costs, along with the additional assumption that the underlying graph is complete, the THNDP can be solved in polynomial time, as shown in the following theorem. This result is based on Theorem 6, stated and proved later on.

Theorem 2. *If the graph G is complete, and all edge costs are equal to 1, the rooted THNDP can be solved in polynomial time for every $L \geq 2$.*

Proof. Let $D = \{\{s, t_1\}, \{s, t_2\}, \ldots, \{s, t_d\}\}$ be a set of d demands rooted at some node s. By Theorem 6, the minimum number of edges of a feasible solution to the THNDP is $\lceil (L+1)d/L \rceil$. Because the graph G is complete, it is easy to build a solution having exactly this number of edges, hence, optimal for the THNDP in G with unitary costs. This solution can be constructed in the following way. We consider the first L destination nodes and cover them, along with s, with a simple cycle of length $L+1$. Observe that each destination node t_i that is considered is covered by two edge-disjoint L-st_i-paths. This procedure is iterated for sets of L destination nodes until there remain $l < L$ destination nodes to cover. If $l = 0$, we are done. If $l = 1$, we link the last destination node t_d to s and to t_{d-1} (two additional edges). The resulting solution contains $\lceil (L+1)d/L \rceil$ edges. If $l > 1$, we cover the l remaining destination nodes, along with s, by a simple cycle of length $l+1$, and the obtained solution contains the desired number of edges. ■

3. INTEGER PROGRAMMING FORMULATION

It is clear that the incidence vector x^F of any solution (N, F) of the THNDP satisfies the following inequalities:

$$x(\delta(W)) \geq 2, \quad \text{for all } st\text{-cuts } \delta(W), \{s,t\} \in D,$$
$$1 \geq x(e) \geq 0, \quad \text{for all } e \in E.$$

The first inequalities are called *st-cut inequalities*, and the second ones are called *trivial inequalities*.

In [7], Dahl introduces a class of valid inequalities for the hop-constrained st-path problem as follows. Let $\Pi = (V_0, V_1, \ldots, V_{L+1})$ be a partition of N such that $s \in V_0$, $t \in V_{L+1}$ and $V_i \neq \emptyset$ for all $i = 1, \ldots, L$. Let T be the set of edges $e = uv$ where $u \in V_i$, $v \in V_j$ and $|i - j| > 1$, that is, $T = \Delta_\Pi(G) \backslash E_\Pi^{0,L}$. Then the inequality

$$x(T) \geq 1$$

is valid for the L-path polyhedron. The set T is called an *L-path-cut* (or *L-st-path-cut*). Figure 2 presents an example of an L-st-path-cut for $L = 3$.

Using the same kind of partitions, these inequalities can be generalized in a straightforward way to the THNDP polytope as

$$x(T) \geq 2,$$

for any L-(st-)path-cut T relative to any demand $\{s,t\} \in D$. Constraints of this type will be called *L-(st-)path-cut inequalities*.

In consequence, consider the following inequalities, which are valid for the THNDP for $L \geq 2$:

$$x(\delta(W)) \geq 2, \quad \text{for all } st\text{-cuts } \delta(W), \text{ for all } \{s,t\} \in D, \quad (1)$$
$$x(T) \geq 2, \quad \text{for all } L\text{-}st\text{-path-cuts } T, \text{ for all } \{s,t\} \in D, \quad (2)$$
$$x(e) \leq 1, \quad \text{for all } e \in E, \quad (3)$$
$$x(e) \geq 0, \quad \text{for all } e \in E. \quad (4)$$

We will show that the system of inequalities (1)–(4) along with integrality constraints formulates the THNDP as an integer program when $L = 2, 3$. For this, we need the following lemma, whose proof can be found in [26].

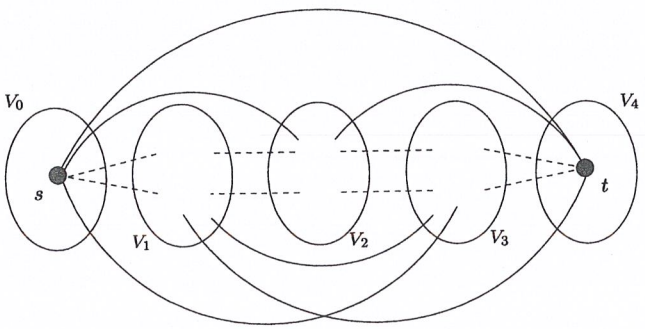

FIG. 2. Support graph of an L-st-path-cut inequality for $L = 3$.

Lemma 1. *Let $G = (N, E)$ be a graph, with s and t two nodes of N, and $L \in \{2, 3\}$. Suppose that two edge-disjoint L-st-paths in G do not exist. If G contains an L-st-path, then there exists an edge that belongs to every L-st-path.*

Theorem 3. *Let $G = (N, E)$ be a graph and $L \in \{2, 3\}$. Then the THNDP is equivalent to the integer program*

$$\min\{cx;\ \text{subject to (1)–(4)},\ x \in \{0,1\}^E\}.$$

Proof. First of all, note that as inequalities (1)–(4) are valid for the THNDP, the system (1)–(4) is satisfied by any feasible solution of the THNDP. Now, let x^F be the incidence vector of a network (V, F) of $G = (N, E)$ that is not feasible for the THNDP. Then, by definition of the problem, there must exist at least one demand $\{s, t\} \in D$ that is not linked by two edge-disjoint paths of length at most L. Let us suppose that x^F satisfies all the cut constraints (1) (and, because it is a boolean vector, all the trivial constraints (3) and (4)). We will show that there is a constraint of type (2) that is violated by x^F for the demand $\{s, t\}$.

By hypothesis, there do not exist in (V, F) two edge-disjoint L-st-paths. Consequently, by Lemma 1, there is at most one edge, say e_0, belonging to every L-st-path (if any) in (V, F). If e_0 exists, we consider the subgraph $H = (V, F\backslash\{e_0\})$. If not, we simply consider $H = (V, F)$. In H, it is clear that there are no L-st-paths. On the other hand, because all the constraints of type (1) are satisfied by x^F, there must exist at least two edge-disjoint paths between s and t in (V, F). Thus, there remains at least one st-path in H, of length at least $L + 1$.

Now we define a partition $(V_0, V_1, \ldots, V_{L+1})$ in H in the following way. The subset V_i, $i = 1, \ldots, L$, contains all the nodes at distance i from s in this subgraph (distance being defined as the minimum number of edges of a path from s to the node). The subset V_{L+1} contains all the other nodes. By the previous remarks, we have that $s \in V_0$, $t \in V_{L+1}$ and all the V_i's are nonempty. Moreover, by construction, there is no edge of H having its endnodes in two subsets V_i and V_j with $|i - j| > 1$. Therefore, by considering the L-st-path-cut associated with this partition, we get that no edge of H belongs to the L-st-path-cut, and hence $x^F(T) < 2$. ∎

Unfortunately, this result is no longer true when $L \geq 4$, because this is already the case when $|D| = 1$; see [26]. In fact, as shown in [25], in addition to inequalities (1)–(4), further valid inequalities are needed to formulate the problem as an integer linear programming problem when $L = 4$. We have, however, identified other classes of valid inequalities for the THNDP, for any $L \geq 2$. The next section is devoted to these classes.

4. VALID INEQUALITIES

In this section, we present new classes of valid inequalities for the THNDP for any $L \geq 2$ and set of demands $D = \{\{s_1, t_1\}, \ldots, \{s_d, t_d\}\}$ with $d \geq 2$. Note that some demands may have common nodes.

Theorem 4. *Let $G = (N, E)$ and $\Pi = (V_0, V_1, \ldots, V_{L+1})$ be a partition of N with $s_1 \in V_0$ (resp., $t_1 \in V_0$) and $t_1 \in V_{L+1}$ (resp., $s_1 \in V_{L+1}$). Let $i_0 \in \{0, \ldots, L\}$ be such that V_{i_0} and V_{i_0+1} induce a $s_{j_1}t_{j_1}$-cut and a $s_{j_2}t_{j_2}$-cut with $\{s_{j_1}, t_{j_1}\}$ and $\{s_{j_2}, t_{j_2}\}$ in D. (Note that j_1 and j_2 may be the same.) Let*

$$\overline{E} = [V_{i_0-1}, V_{i_0}] \cup [V_{i_0+1}, V_{i_0+2}] \cup \left(\bigcup_{k,l \notin \{i_0, i_0+1\},\ |k-l|>1} [V_k, V_l] \right),$$

and $F \subseteq \overline{E}$. Then the inequality

$$x(\Delta_\Pi(G)\backslash(F \cup E_\Pi^{0,i_0-2} \cup E_\Pi^{i_0+2,L})) \geq 3 - \lfloor |F|/2 \rfloor \qquad (5)$$

is valid for $P_G(D, L)$.

Proof. Observe that the following inequalities are valid for the THNDP:

$$\begin{aligned} x(\delta(V_{i_0})) &\geq 2, \\ x(\delta(V_{i_0+1})) &\geq 2, \\ x(\Delta_\Pi(G)\backslash E_\Pi^{0,L}) &\geq 2, \\ -x(f) &\geq -1, \quad \text{for all } f \in F, \\ x(f) &\geq 0, \quad \text{for all } f \in \overline{E}\backslash F. \end{aligned}$$

By summing these inequalities, dividing the sum by 2, and rounding up the right-hand side to the next integer, we obtain inequality (5). ∎

Inequality (5) will be called a *double cut inequality*, and the set of edges having a positive coefficient in (5) a *double cut*. The name double cut comes just from the fact that (exactly) two cuts are used for generating the inequalities by the Chvàtal-Gomory procedure.

Later on, we will discuss special cases of these inequalities that will be used in our experimental study. This will concern in particular the case where $L = 2, 3$ and $i_0 = 0$ (in that case V_{i_0-1} does not exist).

- If $L = 2$, then the partition Π is of the form (V_0, V_1, V_2, V_3) with $s_1 \in V_0$ (resp., $t_1 \in V_0$), $t_2 \in V_1$ and $t_1 \in V_3$ (resp., $s_1 \in V_3$). Consider $F = \{e\} \in [V_1, V_2]$. In this case, the double-cut inequality (5) can be written as

$$x(\delta(V_0)) + x([V_1, V_3]) + x([V_1, V_2]\backslash\{e\}) \geq 3. \qquad (6)$$

- If $L = 3$, then the partition Π is of the form $(V_0, \ldots, V_4)$ with $s_1 \in V_0$ (resp., $t_1 \in V_0$), $t_2 \in V_1$ and $t_1 \in V_3$ (resp., $s_1 \in V_3$). Consider $F = \{e\} \in [V_1, V_2] \cup [V_2, V_4]$. In this case, the double-cut inequality (5) (see Fig. 3) can be written as

$$x(\delta(V_0)) + x([V_1, V_3 \cup V_4]) + x([V_1 \cup V_4, V_2]\backslash\{e\}) \geq 3. \qquad (7)$$

Actually inequality (6) (or (7)) expresses the fact that we need two (jump) edges to satisfy the L-path-cut induced by the partition with respect to s_1 and t_1, and one more edge (with e from $\delta(V_1)$) to satisfy the s_2t_2-cut induced by V_1.

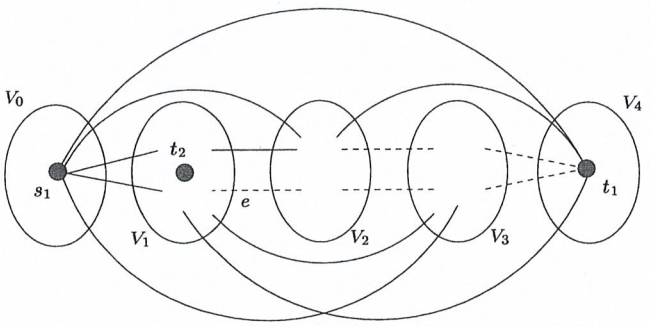

FIG. 3. Support graph of a double cut inequality for $L = 3$.

Our second class of valid inequalities are given for $L = 2, 3$ and are called *triple path-cut inequalities*. In a similar way as before, we will speak of a *triple path-cut* as the set of edges having a positive coefficient in such an inequality. They are defined as follows.

Theorem 5. *(i) Let $L = 2$ and $(V_0, V_1, \ldots, V_4)$ be a partition of N with $s_1, s_2 \in V_0$, $t_1 \in V_3$ and $t_2 \in V_4$. Then the triple path-cut inequality (see Fig. 4)*

$$2x([V_0, V_2]) + x([V_0, V_3]) + x([V_0, V_4]) + x([V_1, V_3]\backslash\{e\}) + x([V_1, V_4]) + x([V_3, V_4]) \geq 3, \quad (8)$$

where $e \in [V_1, V_3]$, is valid for $P_G(D, 2)$.

(ii) Let $L = 3$ and $(V_0, V_1, \ldots, V_5)$ be a partition of N with $s_1, s_2 \in V_0$, $t_1 \in V_4$ and $t_2 \in V_5$. Then the triple path-cut inequality

$$2x([V_0, V_2]) + 2x([V_0, V_3]) + 2x([V_1, V_3]) + x([V_0 \cup V_1 \cup V_2 \cup V_3, V_4 \cup V_5]\backslash\{e\}) + x([V_4, V_5]) \geq 3, \quad (9)$$

where $e \in [V_2 \cup V_3, V_4] \cup [V_3, V_5]$, is valid for $P_G(D, 3)$.

Proof. (i) Let T_1 be the L-s_1t_1-path-cut induced by the partition $(V_0, V_1 \cup V_4, V_2, V_3)$, and T_2 and T_3 be the L-s_2t_2-path-cuts induced by $(V_0, V_1 \cup V_3, V_2, V_4)$ and $(V_0, V_1, V_2 \cup V_3, V_4)$, respectively. Then the following inequalities are valid for the THNDP:

$$\begin{aligned} x(T_1) &\geq 2, \\ x(T_2) &\geq 2, \\ x(T_3) &\geq 2, \\ -x(e) &\geq -1, \\ x(f) &\geq 0, \qquad \text{for all } f \in [V_1, V_3]\backslash\{e\}. \end{aligned}$$

By summing these inequalities, dividing the sum by 2, and rounding up the right-hand side, we obtain inequality (8).

(ii) Let T_1 be the L-s_1t_1-path-cut induced by the partition $(V_0, V_1 \cup V_5, V_2, V_3, V_4)$, and T_2 and T_3 be the L-s_2t_2-path-cuts induced by $(V_0, V_1 \cup V_4, V_2, V_3, V_5)$ and $(V_0, V_1, V_2, V_3 \cup V_4, V_5)$, respectively. Then the following inequalities are valid for the THNDP:

$$\begin{aligned} x(T_1) &\geq 2, \\ x(T_2) &\geq 2, \\ x(T_3) &\geq 2, \\ -x(e) &\geq -1, \\ x(f) &\geq 0, \qquad \text{for all } f \in ([V_2 \cup V_3, V_4] \cup [V_3, V_5])\backslash\{e\}. \end{aligned}$$

By summing these inequalities, dividing the sum by 2, and rounding up the right-hand side, we obtain inequality (9). ∎

The name triple path-cut comes just from the fact that three path-cut inequalities are used for generating the inequalities by the Chvàtal-Gomory procedure.

Given a partition Π based on a subset of rooted demands, our next class of inequalities gives the minimum number of edges of the multicut $\Delta_\Pi(G)$, used by a feasible solution of the THNDP. We will call the set of edges with positive coefficient in such a multicut a *rooted-partition* and the associated inequality a *rooted-partition inequality*.

Theorem 6. *Let $L \geq 2$ and $T = \{t_1, \ldots, t_k\}$ be a subset of k destination nodes relative to node s. Let $\Pi = (V_0, V_1, \ldots, V_k)$ be a partition of N such that $s \in V_0$, and $t_i \in V_i$, for all $i = 1, \ldots, k$. See Figure 5 for an illustration. Then the inequality*

$$x(\Delta_\Pi(G)) \geq k + \lceil k/L \rceil \quad (= \lceil (L+1)k/L \rceil) \qquad (10)$$

is valid for $P_G(D, L)$.

Proof. The proof is by induction on k.

If $k = 1$, it is obvious that we need at least two edges in any feasible solution, and thus (10) is satisfied. Suppose that the statement holds for any partition based on at most $k - 1$ destination nodes relative to s. We will show that the statement remains true for the partition Π.

First notice that we can assume that the V_i's contain only one node, that is $V_0 = \{s\}$ and $V_i = \{t_i\}$ for $i = 1, \ldots, k$. If not, we could indeed consider the graph obtained from G by contracting the V_i's with $|V_i| \geq 2$. It is clear that neither

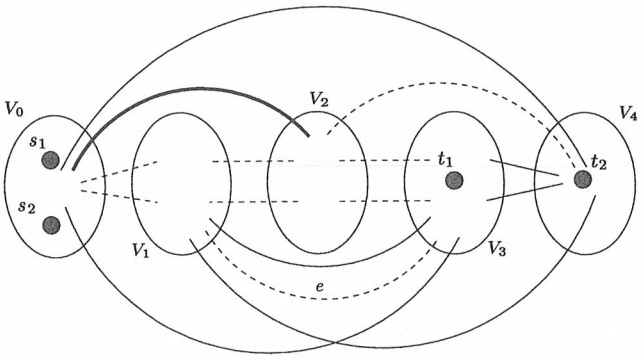

FIG. 4. Support graph of a triple path-cut inequality for $L = 2$.

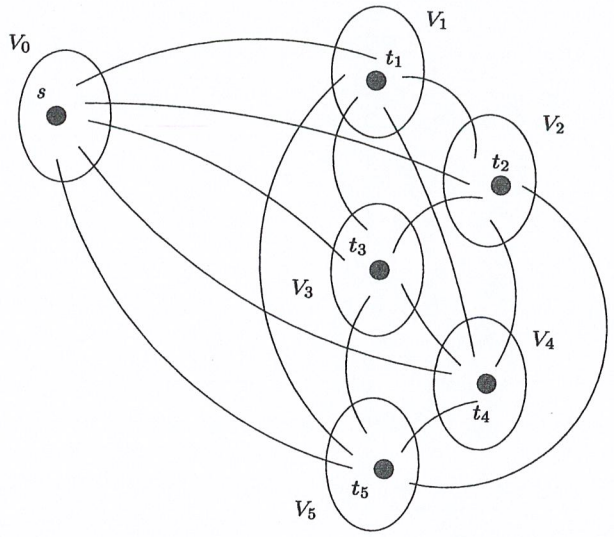

FIG. 5. Support graph of a rooted-partition inequality on $k = 5$ demands.

the number of edges in the multicut, nor the number k of distinguished destinations, would be changed by this operation, and hence, the inequality to prove remains the same.

Now let F be a feasible solution of the THNDP. Let T_1 (resp., T_2) be the subset of destinations adjacent to s (resp., not adjacent to s) in F. By the previous remark, we can also assume that $T_1 \neq \emptyset$. Let $t_1 \in T_1$. There must exist in F another L-path between s and t_1. Note that this path can use an edge that is parallel to the first edge between s and t_1. These two paths give us a cycle C of length c, with $2 \leq c \leq L+1$, going through at most $c-1$ destinations. Suppose, w.l.o.g., that $C = \{s, t_1, \ldots, t_{c-1}\}$. Let $\Pi' = (V'_0, V'_1, \ldots, V'_{k-c+1})$ be the partition given by $V'_0 = \{s, t_1, \ldots, t_{c-1}\}$ and $V'_i = V_{i+c-1}$ for $i = 1, \ldots, k-c+1$. By induction, we have

$$x(\Delta_{\Pi'}(G)) \geq (k-(c-1)) + \lceil (k-(c-1))/L \rceil.$$

This yields

$$x(\Delta_{\Pi}(G)) \geq (k-(c-1)) + \lceil (k-(c-1))/L \rceil + c.$$

Now, if $k = q + Lp$ for some integers p, q, we have $\lceil k/L \rceil = \lceil q/L \rceil + p$. Because $c \leq L+1$, we also have $k-(c-1) \geq k-L$. Hence, $\lceil (k-(c-1))/L \rceil \geq \lceil (k-L)/L \rceil = \lceil q/L \rceil + p - 1$. So,

$$\begin{aligned} x(\Delta_{\Pi}(G)) &\geq k + 1 + \lceil q/L \rceil + p - 1 \\ &= k + \lceil k/L \rceil. \end{aligned}$$

■

As it will turn out, the rooted-partition inequalities cannot define facets when k is a multiple of L or when the sets $V_1, \ldots, V_k$ are not all reduced to singletons. We will now present a strengthening of inequalities (10) for $L = 2$ when at least one set of $V_1, \ldots, V_k$ is not reduced to a single node.

Theorem 7. *Let $L = 2$ and $T = \{t_1, \ldots, t_k\}$ be a subset of $k \geq 2$ destination nodes of s among the demands of D. Let $\Pi = (V_0, \ldots, V_{k-1}, V_k, V_{k+1})$ be a partition of N such that $s \in V_0$, and $t_i \in V_i$, for all $i = 1, \ldots, k$. Let $e \in [V_{k-1}, V_{k+1}]$; see Figure 6 for an illustration. Then the inequality*

$$x(\Delta_{\Pi}(G) \setminus ([V_k, V_{k+1}] \cup \{e\})) \geq \lceil 3k/2 \rceil \qquad (11)$$

is valid for $P_G(D, 2)$.

Proof. First note that, as we did in Theorem 6, we can suppose that the sets V_i, $i = 0, \ldots, k+1$, are reduced to single nodes. Also, we may suppose that the (single) node of V_{k+1}, say v, is a Steiner node. Indeed, suppose that (11) is valid with v a Steiner node. If F is a solution of the problem with v a terminal, then, clearly, F is also a solution of the problem with v a Steiner node, and hence, x^F satisfies (11). This implies that (11) is also valid if v is a terminal node.

Consider the rooted-partition inequality induced by the partition $(V_0, \ldots, V_{k-1} \cup V_{k+1}, V_k)$,

$$x(\delta(V_0, \ldots, V_{k-1} \cup V_{k+1}, V_k)) \geq \lceil 3k/2 \rceil.$$

Similarly, by collapsing V_{k+1} and V_k, we get

$$x(\delta(V_0, \ldots, V_{k-1}, V_k \cup V_{k+1})) \geq \lceil 3k/2 \rceil.$$

Because $\Delta_{\Pi}(G) = \delta(V_0, \ldots, V_{k-1} \cup V_{k+1}, V_k) \cup [V_{k-1}, V_{k+1}] = \delta(V_0, \ldots, V_{k-1}, V_k \cup V_{k+1}) \cup [V_k, V_{k+1}]$, the previous two inequalities can be respectively written as

$$x(\Delta_{\Pi}(G)) - x([V_{k-1}, V_{k+1}]) \geq \lceil 3k/2 \rceil, \qquad (12)$$

and

$$x(\Delta_{\Pi}(G)) - x([V_k, V_{k+1}]) \geq \lceil 3k/2 \rceil. \qquad (13)$$

Let F be a solution to the THNDP. We distinguish two cases.

CASE 1. $e \notin F$ or $F \cap [V_k, V_{k+1}] = \emptyset$. If $e \notin F$, then, by (13), we have that (11) is satisfied by x^F, the incidence vector

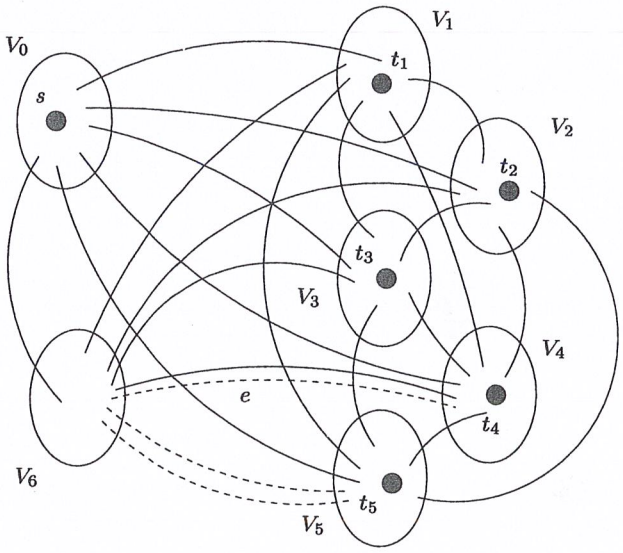

FIG. 6. Support graph of a rooted-opt-partition inequality on $k = 5$ demands.

of F. If F does not intersect $[V_k, V_{k+1}]$, then inequality (12) implies (11), and thus this latter inequality is satisfied by x^F.

CASE 2. $e \in F$ and $F \cap [V_k, V_{k+1}] \neq \emptyset$. First note that we may suppose that $F \cap [s, V_{k+1}] \neq \emptyset$, because otherwise $F \backslash \{e\}$ would be a solution of the THNDP, and Case 1 would apply. So let us consider an edge, say e_0, of $F \cap [s, V_{k+1}]$. Let also e' be an edge in $F \cap [V_k, V_{k+1}]$.

Suppose that there is a cycle C of length $c = 2$ or 3, linking s and $c-1$ destinations from $\{t_1, \ldots, t_{k-2}\}$. W.l.o.g., we may suppose that C is simple and $C = \{s, t_1, \ldots, t_{c-1}\}$. Let $\Pi' = (V'_0, \ldots, V'_{k-c+2})$ be such that $V'_0 = \bigcup_{i=0,\ldots,c-1} V_i$, and $V'_i = V_{i+c-1}$, for $i = 1, \ldots, k-c+2$. We have

$$x^F(\Delta_{\Pi'}(G)) = x^F(\Delta_{\Pi}(G)) - c. \tag{14}$$

Hence, if inequality (11) is satisfied by x^F with respect to Π' and e, we get

$$x^F(\Delta_{\Pi'}(G) \backslash ([V'_{k-c+1}, V'_{k-c+2}] \cup \{e\})) \geq \lceil 3(k-c+1)/2 \rceil.$$

As $c \leq 3$, by (14), it follows that inequality (11) is also satisfied by x^F with respect to Π and e. Thus, we may suppose that the subgraph, say $\widetilde{G} = (\widetilde{V}, \widetilde{E})$, induced by F on the nodes $s, t_1, \ldots, t_{k-2}$ does not contain any cycle of length 2 or 3.

Let T_1 (resp., T_2) be the set of nodes among $\{t_1, \ldots, t_{k-2}\}$ that are linked (resp., not linked) to s by edges in F. Note that, for every node $t_i \in T_1$, we already have one 2-st_i-path, namely the edge st_i. Because $\widetilde{G}$ does not contain cycles of length 2 or 3, the second 2-st_i-path for $t_i \in T_1$ must use one edge of $[t_i, \{t_{k-1}, t_k, v\}]$. Moreover, all those edges are different. Therefore, to cover T_1, F uses at least $2|T_1|$ edges.

Next, each node in T_2 must be linked to s by two 2-paths, each one going through either T_1 or $\{t_{k-1}, t_k, v\}$. Hence, F must use two additional edges for every node of T_2. In consequence, in the 2-paths between s and the nodes $\{t_1, \ldots, t_{k-2}\}$, F uses at least $2(|T_1| + |T_2|) = 2(k-2)$ edges.

Let $\widetilde{F}$ be the set of these edges, and $\widetilde{F}'$ be the set of edges of F different from those in $\widetilde{F} \cup [V_k, V_{k+1}] \cup \{e\}$. We have the following claim.

Claim. $|\widetilde{F}'| \geq 3$.

Proof. First note that $e_0 \in \widetilde{F}'$. Also observe that, to cover t_{k-1}, t_k, besides e_0, e, e', we still need one 2-path from s to t_{k-1} and another from s to t_k. (Recall that v is a Steiner node.) We consider two cases.

CASE 2.1. None of the edges of $\widetilde{F}$ is incident to t_{k-1} or t_k. This implies that the edges of $\widetilde{F}$ used in 2-st_i-paths, with $t_i \in T_1 \cup T_2$, going through the node set $\{t_{k-1}, t_k, v\}$ are all incident to v. So, to cover t_{k-1}, we need at least one more edge, say h, from $\delta(t_{k-1}) \backslash \{e\}$. Moreover, we note that $h \in \widetilde{F}'$. If $h \notin [t_{k-1}, t_k]$, then, to cover t_k, one more edge in $\widetilde{F}'$ is needed, and hence, $|\widetilde{F}'| \geq 3$. If $h \in [t_{k-1}, t_k]$, again it is not hard to see that one more edge from $\widetilde{F}'$ is necessary to cover both t_{k-1} and t_k.

CASE 2.2. Some of the edges of $\widetilde{F}$ are incident to t_{k-1} or t_k. If there is an edge of $\widetilde{F}$ incident to t_{k-1} (resp., t_k), then $\widetilde{F}'$ must contain the edge st_{k-1} (resp., st_k). If not, the edge of $\widetilde{F}$ would not belong to a 2-st_i-path, where $t_i \in T_1 \cup T_2$, which is a contradiction. Therefore, if there are edges of $\widetilde{F}$ incident to both t_{k-1} and t_k, then clearly $\widetilde{F}'$ contains at least the edges e_0, st_{k-1}, st_k and thus $|\widetilde{F}'| \geq 3$. Now suppose that there is only one node t_i among t_{k-1}, t_k that is incident to some edge of $\widetilde{F}$. Thus $st_i \in \widetilde{F}'$. Moreover, to cover $t_j, j \in \{k-1, k\} \backslash \{i\}$, we need one more edge in $\widetilde{F}'$, which implies that $|\widetilde{F}'| \geq 3$. □

In consequence, by the claim above, and as $\widetilde{F} \cup \widetilde{F}' \subseteq F$, we have

$$|F| \geq 2(k-2) + 3 \geq \lceil 3k/2 \rceil,$$

for all $k \geq 2$. ■

Inequalities (11) will be called *rooted-opt-partition inequalities*. The *rooted-opt-partition* will then be the set of edges with positive coefficient in the corresponding inequality.

Consider a rooted-partition $(V_0, \ldots, V_k)$. If $|V_i| \geq 2$ for some i in $\{1, \ldots, k\}$, then one can move one of the nodes of V_i into V_{k+1}. In consequence, the resulting rooted-opt-partition inequality dominates the first one. As mentioned before, this implies that if $V_1, \ldots . V_k$ are not all reduced to singletons, the corresponding rooted-partition inequality does not define a facet.

Note that the extension of the rooted-opt-partition inequalities to any L leads to nonvalid inequalities for the THNDP. Indeed, consider a rooted instance where s is the source node, t_1 and t_2 are the destination nodes of s, and v is a Steiner node. A cycle of length four spanning s, t_1, v, t_2 in this order is feasible for $L = 3$. However, if we take $V_0 = \{s\}$, $V_1 = \{t_1\}$, $V_2 = \{t_2\}$ and $V_3 = \{v\}$ (hence, $k = 2$), the corresponding rooted-opt-partition inequality would be

$$x_{st_1} + x_{st_2} + x_{sv} + x_{t_1t_2} \geq \lceil 4k/3 \rceil = 3,$$

which is violated by this point.

5. FACETS OF THE ROOTED THNDP POLYTOPE

In this section, we will consider the rooted case, where all the demands are rooted at a unique source node s. So, let $G = (N, E)$ be a graph and $D = \{\{s, t_1\}, \ldots, \{s, t_d\}\}$ be a set of rooted demands. We will describe necessary conditions and sufficient conditions for the previous inequalities to be facet defining. Besides their theoretical interest, these conditions will be used in the next section to devise efficient separation procedures.

First, we discuss the dimension of $P_G(D, L)$. If $D = \{\{s, t\}\}$, then $dim(P_G(D, L)) = |E| - |E^*_{st}|$ where E^*_{st} is the set of essential edges of G, relative to $\{s, t\}$; see [26]. An edge e is essential relative to $\{s, t\}$ if and only if e belongs to an st-cut or L-st-path-cut of cardinality 2. If we extend this definition of E^*_{st} to any demand $\{s, t\} \in D$, we get that

$dim(P_G(D,L)) = |E| - |\bigcup_{\{s,t\}\in D} E^*_{st}|$. In the following, we will always suppose that $G = (N,E)$ is a complete graph with $|N|$ large enough to have $\bigcup_{\{s,t\}\in D} E^*_{st} = \emptyset$, and hence, $P_G(D,L)$ full dimensional. This assumption can always be achieved by adding nodes and edges with sufficiently high weights.

Let

$$S(G) = \{F \subseteq E \mid (N,F) \text{ is a solution of the THNDP}\}.$$

Given an inequality $ax \geq \alpha$ valid for $P_G(D,L)$, we will denote by

$$S_a(G) = \{F \in S(G) \mid ax^F = \alpha\}.$$

We first give two lemmas that will be frequently used in the sequel, sometimes without explicit reference. Their proofs being obvious, they are omitted.

Lemma 2. *Let $ax \geq \alpha$ be a facet defining inequality of $P_G(D,L)$, different from a trivial inequality. Then for every edge $e \in E$, there exists an edge subset in $S_a(G)$ that contains e and another one that does not.*

Lemma 3. *Let $ax \geq \alpha$, where $a = (a(e), e \in E)$, be a facet defining inequality of $P_G(D,L)$, different from a trivial inequality. Then $a(e) \geq 0$, for all $e \in E$ and $\alpha > 0$.*

We now present necessary and sufficient conditions for the double cut inequalities to be facet defining when $L = 2$. We consider first the case where $s \in V_0$.

Theorem 8. *Let $L = 2$ and $G = (N,E)$ be a complete graph with $|N| \geq 4$, one source node s and d destination nodes $T = \{t_1, \ldots, t_d\}$. Let (V_0, V_1, V_2, V_3) be a partition of N such that $s \in V_0$, $t_1 \in V_1$ and $t_2 \in V_3$. Let $e \in [V_1, V_2]$. Then the double cut inequality (6) is facet inducing for $P_G(D,2)$ if and only if all the following hold:*

(i) $|V_0| = 1$,
(ii) $|V_1| = 1$,
(iii) $d = 2$,
(iv) if v_2 is the endnode of e in V_2 and $|[t_1, v_2]| \geq 2$, then $|[t_2, v_2]| \geq 2$ and $|[s, v_2]| \geq 2$.

Proof. Let us denote inequality (6) by $ax \geq \alpha$ and let $S_a(G)$ be the induced face.

Necessity

1. Suppose, on the contrary, that inequality (6) is facet defining while $|V_0| \geq 2$. Let $v_0 \in V_0 \backslash \{s\}$. We are going to show that any feasible solution $F \in S_a(G)$ does not intersect $[v_0, V_2]$, which contradicts Lemma 2. Suppose that this is not the case, and that there exists a feasible solution F containing $f \in [v_0, V_2]$ and such that x^F satisfies (6) as an equality. First, note that, as f belongs to the double cut and $ax^F = \alpha = 3$, F contains exactly two more edges of it (recall that the double cut is the set of edges such that $a(e) > 0$). On the other hand, because f is incident neither to s, nor to t_1 (resp. t_2), f is not useful to the two 2-paths between s and t_1 (resp., t_2) that F must contain. However, it is not possible to build these four paths with only two more edges from the double cut.
2. Suppose, on the contrary, that inequality (6) is facet defining while $|V_1| \geq 2$. Hence, because the graph G is complete, we have $|[V_1, V_2]| \geq 2$. Suppose first that $e \in [t_1, V_2]$. Then, we have that any edge $f \in [V_1 \backslash \{t_1\}, V_2]$ belongs to the double cut. However, f is not useful for building 2-paths between s and t_1, or s and t_2. Therefore, any feasible solution $F \in S_a(G)$ cannot contain f. Thus, by Lemma 2, we have a contradiction.

   Suppose now that $e \in [V_1 \backslash \{t_1\}, V_2]$. In that case, any edge $f \in [t_1, V_2]$ is in the double cut. Consider a solution $F \in S_a(G)$ that contains f. If f is not used in a 2-st_1-path of F, then f does not belong to any of the 2-paths between s and t_1, and s and t_2, and therefore, we get the same contradiction as before. As a consequence, f must belong to a 2-st_1-path of F of the form (s, v_2, t_1) with $v_2 \in V_2$. As this path uses two edges of the double cut and $ax^F = \alpha = 3$, F can only contain one more edge of it for the other 2-st_1-path and for at least one of the two 2-st_2-paths (given that sv_2 can be used together with an edge of $[v_2, t_2]$ to form a 2-st_2-path). But this is also impossible.
3. By definition of the double cut, we have already that $d \geq 2$ holds. Let us show now that $d \leq 2$ when the corresponding inequality is facet defining. Suppose, on the contrary, that the double cut inequality is facet defining while there exists a third destination of s, say t_3. By (i) and (ii), we have $t_3 \in V_2 \cup V_3$. Consider a solution $F \in S_a(G)$ not containing e. Therefore, the two 2-st_1-paths of F are only constituted of edges in the double cut. First, it is clear that these paths cannot be both of length 2. If one of the paths is of length 1 and the other is of length 2, we have already three edges taken by F in the double cut, while we still need to construct at least one 2-path for one of the two destination nodes t_2, t_3. Clearly, this is impossible. Finally, if both 2-st_1-paths are of length 1, we can still take one more edge in the double cut. But, this time, we need to link both t_2 and t_3 to s by two 2-paths. Once again, this is not possible. We, therefore, get that any solution $F \in S_a(G)$ does contain e, which contradicts Lemma 2.
4. Suppose that the double cut inequality is facet defining and that $|[t_1, v_2]| \geq 2$. Therefore, any edge $e' \in [t_1, v_2] \backslash \{e\}$ belongs to the double cut. By Lemma 2, there must exist a feasible solution F in $S_a(G)$ containing e'. Moreover, e' must belong to a 2-st_1-path of F. This path is then of the form (sv_2, e'). As sv_2 and e' belong to the double cut, only one more edge of it must be used to form a second 2-st_1-path and one 2-st_2-path (the other one using the edge sv_2 plus an edge in $[v_2, t_2]$). Clearly, the only possibility to do that is to go once again through v_2. As a consequence, we obtain that both $[s, v_2]$ and $[v_2, t_2]$ contain parallel edges.

Sufficiency

Suppose that (1), (2), (3), and (4) hold. Let v_2 be the endnode of e in V_2. Let $bx \geq \beta$ be a facet defining inequality of $P_G(D,2)$ such that

$$\{x^F \in P_G(D,2) \mid ax^F = \alpha\} \subseteq \{x^F \in P_G(D,2) \mid bx^F = \beta\}.$$

As before, we will show that $b = \rho a$ for some $\rho > 0$.

Consider the solution $F = \{st_1, st_2, t_1t_2\}$. Clearly, its incidence vector x^F satisfies $ax = \alpha$. Note that we can add to F all the edges in $E(V_2 \cup V_3) \cup \{e\}$ while x^F still satisfies $ax = \alpha$. Consequently, we get

$$b(f) = 0, \text{ for all } f \in E(V_2 \cup V_3) \cup \{e\}. \quad (15)$$

Now, let $F^* = (F \backslash \{t_1t_2\}) \cup \{sv_2, e, v_2t_2\}$. Clearly, $F^* \in S_a(G)$. Hence, $bx^{F^*} = \beta$. As by (15) $b(e) = b(v_2t_2) = 0$, this yields

$$b(t_1t_2) = b(sv_2). \quad (16)$$

The same holds if in F^* we replace st_1 (or st_2) by t_1t_2. Therefore,

$$b(st_1) = b(t_1t_2) = b(st_2). \quad (17)$$

From (16) and (17), we get

$$b(st_1) = b(st_2) = b(t_1t_2) = b(sv_2).$$

As these edges are arbitrary edges from $[s, t_1]$, $[s, t_2]$, $[t_1, t_2]$, $[sv_2]$, respectively, we obtain

$$b(f) = \rho, \text{ for all } e \in [s, t_1] \cup [s, t_2] \cup [t_1, t_2] \cup [s, v_2], \quad (18)$$

for some scalar ρ.

Now consider the solution $F_v = \{sv_2, e, v_2t_2, sv, vt_1, vt_2\}$ for $v \in (V_2 \cup V_3) \backslash \{v_2, t_2\}$. It is not hard to see that $F \in S_a(G)$. Therefore, $bx^{F_v} = \beta$. As $bx^{F^*} = \beta$, using (15) we obtain

$$b(st_1) + b(st_2) = b(sv) + b(vt_1). \quad (19)$$

Also consider $F'_v = (F_v \backslash \{vt_1\}) \cup \{st_1\}$. Because $F'_v \in S_a(G)$, and hence $bx^{F'_v} = \beta$, we get

$$b(st_1) = b(vt_1). \quad (20)$$

From (19) and (20), it follows that

$$b(st_2) = b(sv). \quad (21)$$

From (20) and (21), together with the fact that sv (resp., vt_1) is an arbitrary edge between s and v (resp., v and t_1), by (18) it follows that

$$b(f) = \rho, \text{ for all } f \in [\{s, t_1\}, V_2 \cup V_3] \backslash [v_2, t_1]. \quad (22)$$

If there are edges in $[v_2, t_1] \backslash \{e\}$, we still need to prove that their coefficient in b is equal to ρ. Suppose this is the case, and let v_2t_1 be an edge of $[v_2, t_1] \backslash \{e\}$. Then, by (iv), the edge set $\widetilde{F} = \{sv_2, g, e, v_2t_1, v_2t_2, h\}$ exists. Here, g and h are edges parallel to sv_2 and v_2t_2, respectively. Clearly, its incidence vector $x^{\widetilde{F}}$ satisfies $ax = \alpha$. Thus, $bx^{\widetilde{F}} = \beta$. As $bx^{F^*} = \beta$, we obtain

$$b(g) + b(v_2t_1) = 2b(st_1). \quad (23)$$

From (18) and the fact that v_2t_1 is an arbitrary edge of $[v_2, t_1] \backslash \{e\}$, we get

$$b(f) = \rho, \text{ for all } f \in [v_2, t_1] \backslash \{e\}. \quad (24)$$

From (15), (18), (22), (24), we have $b = \rho a$. As $bx \geq \beta$ is a facet defining inequality different from a trivial one, by Lemma 3, it follows that $\rho > 0$. ∎

The following theorem gives a similar result for the double-cut inequalities based on partitions with $s \in V_3$. Its proof is along the same line as that of Theorem 8.

Theorem 9. *Let $L = 2$ and $G = (N, E)$ be a complete graph with $|N| \geq 6$, one source node s and d destination nodes $T = \{t_1, \ldots, t_d\}$. Let (V_0, V_1, V_2, V_3) be a partition of N such that $t_2 \in V_0$, $t_1 \in V_1$ and $s \in V_3$. Let $e \in [V_1, V_2]$. Then the double cut inequality (6) is facet inducing for $P_G(D, 2)$ if and only if all of the following hold:*

(i) $|V_0| = 1$,
(ii) $|V_1| = 1$,
(iii) *if v_2 is the endnode of e in V_2 and $|[t_1, v_2]| \geq 2$, then* $|[s, v_2]| \geq 2$.

For $L = 3$, we have also investigated necessary conditions for the double cut inequalities (7) to be facet defining. We have shown the following result, given here without proof because it is similar to that of Theorem 8.

Theorem 10. *If $L = 3$, inequality (7) defines a facet of $P_G(D, 3)$ only if*

(i) $|V_0| = 1$,
(ii) $|V_1| = 1$, *and*
(iii) $|V_4| \leq 2$.

We have also investigated the conditions under which the triple path-cut inequalities are facet defining when $L = 2$. Note that, in the case of rooted demands, we simply pose $s_1 = s_2$.

Theorem 11. *Let $L = 2$ and $G = (N, E)$ be a complete graph with $|N| \geq 5$, one source node s and d destination nodes $T = \{t_1, \ldots, t_d\}$. Let $(V_0, V_1, V_2, V_3, V_4)$ be a partition of N such that $s \in V_0$, $t_1 \in V_3$ and $t_2 \in V_4$. Let $e \in [V_1, V_3]$. Then the triple path-cut inequality (8) is facet defining for $P_G(D, 2)$ if and only if all of the following hold:*

(i) $|V_0| = 1$,
(ii) $|V_3| = 1$,
(iii) $|V_4| = 1$,
(iv) *if v_1 is the endnode of e in V_1 and $|[v_1, t_1]| \geq 2$, then* $|[s, v_1]| \geq 2$.
(v) *if $|V_1| \leq 2$ and $V_1 \subset T$, then there exists $t \in V_1 \cap T$ such that* $|[s, t]| \geq 2$.
(vi) *if $|V_1| = 2$ and $V_1 \cap T = \{t_3\}$, then at least one of the following holds:*
— *e is incident to t_3,*
— $|[s, V_1 \backslash \{t_3\}]| \geq 2$,
— $|[s, t_3]| \geq 2$.

Proof. See [24]. ∎

Note that triple path-cut inequalities never define facets of $P_G(D, L)$ when $L = 3$. It is indeed impossible to find a feasible solution satisfying such an inequality with equality while containing an edge in $[V_1, V_3]$ (of coefficient 2 in the triple path-cut). However, let us remark that these inequalities will be useful in our branch-and-cut algorithm. In fact, although they are not facet defining, they permit us to cut off fractional solutions.

The following two theorems give conditions for inequalities (10) to be facet defining.

Theorem 12. *Let $G = (N, E)$ be a complete graph and let $t_1, \ldots, t_k$ be k destination nodes associated with s. Let $\Pi = (V_0, V_1, \ldots, V_k)$ be a partition of N such that $s \in V_0$ and $t_i \in V_i$ for $i = 1, \ldots, k$. If the rooted-partition inequality (10) is facet defining, then k is not a multiple of L.*

Proof. Suppose that k is a multiple of L. We will show that, for any feasible solution F whose incidence vector x^F satisfies (10) with equality, we have $|F \cap [s, T]| = 2k/L$, where $T = \{t_1, \ldots, t_k\}$. But this will imply that every solution of the face defined by (10) satisfies the equation

$$x([s, T]) = 2k/L.$$

As (10) is not a positive multiple of this equation, it cannot define a facet.

For this, first note that, as we did before, we may suppose that the V_i's are reduced to single nodes, that is $V_0 = \{s\}$, $V_i = \{t_i\}$, $i = 1, \ldots, k$. It is easy to see that if the statement holds in this case, it also holds when the elements of the partition are not necessarily singletons.

Claim 1. The solution F does not contain any chordless cycle containing s of length $\leq L$.

Proof. Assume to the contrary that there exists in F a chordless cycle C spanning s and at most $L - 1$ destination nodes of s. Suppose, w.l.o.g., that C spans $V_0, V_1, \ldots, V_{c-1}$, $c \leq L$. Let $\Pi' = (V'_0, V'_1, \ldots, V'_{k-c+1})$ be the partition given by $V'_0 = \bigcup_{i=0,\ldots,c-1} V_i$, and $V'_i = V_{i+c-1}$, for $i = 1, \ldots, k - c + 1$. By the validity of the rooted-partition inequality induced by Π', we have

$$|\Delta_{\Pi'}(G) \cap F| \geq \lceil (L + 1)(k - c + 1)/L \rceil.$$

On the other hand, we have $|\Delta_{\Pi}(G) \cap F| = (L + 1)k/L$. As $|\Delta_{\Pi}(G) \cap F| = |\Delta_{\Pi'}(G) \cap F| + c$, it follows that

$$\begin{aligned}(L + 1)k/L &\geq \lceil (L + 1)(k - c + 1)/L \rceil + c \\ &= \lceil (k - c + 1)/L \rceil + k + 1.\end{aligned}$$

As $c \leq L$, we have

$$k/L \geq \lceil (k - L + 1)/L \rceil + 1 = k/L + 1,$$

where the last equality comes from the fact that k is a multiple of L. But this is a contradiction. □

Consequently, by Claim 1, F does not contain any cycle that spans V_0 and the sets V_is, and whose length is less than or equal to L.

Claim 2. Any two cycles $C_1, C_2 \subseteq F, C_1 \neq C_2$, of length $L + 1$ and going through s, cannot have a destination node in common.

Proof. Assume, to the contrary, that C_1 and C_2 intersect in d destination nodes, $L \geq d \geq 1$. Let p be the number of destination nodes covered by $C_1 \cup C_2$. First, we show that $p < k$. In fact, suppose, by contradiction, that $p = k$. If $k = L$, then $|F| = L + 1$. As $|C_1| = L + 1$ and $|C_2 \backslash C_1| \geq 1$, we have

$$|F| \geq |C_1| + |C_2 \backslash C_1| \geq L + 2,$$

a contradiction.

So suppose that $k > L$. Then

$$p = 2L - d > L.$$

Therefore, $d < L$. But this implies that $p = k$ is not a multiple of L, a contradiction.

Consequently, $p < k$. Also, as $d \geq 1$, we have $p \leq 2L - 1$. Moreover, observe that $|C_1 \cup C_2| \geq p + 2$. This follows from the fact that $C_1 \cup C_2$ covers $p + 1$ nodes and $C_1 \cup C_2$ is not a simple cycle.

Suppose, w.l.o.g., that $t_1, \ldots, t_p$ are the destination nodes covered by $C_1 \cup C_2$. Let $\widetilde{\Pi} = (\widetilde{V}_0, \widetilde{V}_1, \ldots, \widetilde{V}_{k-p})$ be such that $\widetilde{V}_0 = \{s, t_1, \ldots, t_p\}$, and $\widetilde{V}_i = \{t_{i+p}\}$, for $i = 1, \ldots, k - p$.

By the validity of the rooted-partition inequality corresponding to $\widetilde{\Pi}$, we have

$$|\Delta_{\widetilde{\Pi}}(G) \cap F| \geq \lceil (L + 1)(k - p)/L \rceil.$$

Because $|\Delta_{\Pi}(G) \cap F| \geq |\Delta_{\widetilde{\Pi}}(G) \cap F| + p + 2$, we get

$$\begin{aligned}|\Delta_{\Pi}(G) \cap F| &\geq \lceil (L + 1)(k - p)/L \rceil + p + 2, \\ &= k + 2 + \lceil (k - p)/L \rceil, \\ &\geq k + 2 + \lceil (k - (2L - 1))/L \rceil, \\ &= k + k/L + \lceil 1/L \rceil, \\ &= k + k/L + 1.\end{aligned}$$

But this contradicts the fact that x^F satisfies (10) with equality, and the claim is proved. □

Claim 3. The solution F does not contain any chordless cycle containing s of length $\geq L + 2$.

Proof. Suppose there is a cycle $C \subseteq F$ of length $c \geq L + 2$. We assume that c is minimum. Suppose w.l.o.g. that C goes through $s, t_1, \ldots, t_{c-1}$ in this order. Because F is a feasible solution for the THNDP and, by Claim 1, F does not contain cycles of length less than or equal to L, there must exist two paths P_1 and P_2 of length L joining s to t_1 and t_{c-1}, respectively. Let $C_1 = \{st_1\} \cup P_1$ and $C_2 = \{st_{c-1}\} \cup P_2$.

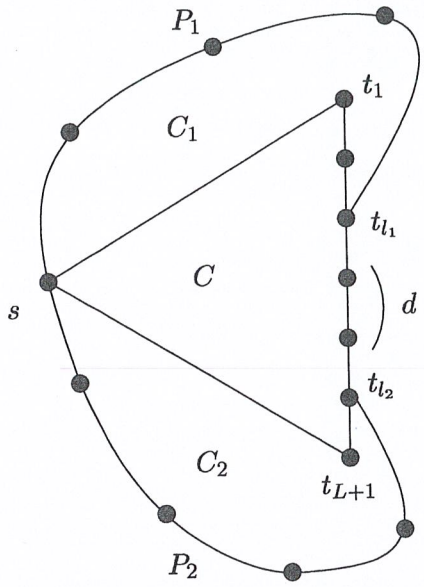

FIG. 7. An example of an $(L+2)$-cycle C and the two associated $(L+1)$-cycles C_1 and C_2.

If $c \geq L+3$, then there must also exist a chordless path P from s to t_2 of length either $L-1$ or L. If P is of length $L-1$, then the cycles C_1 and $P \cup \{st_1, t_1t_2\}$ are both of length $L+1$. Because these cycles intersect at t_1, this contradicts Claim 2. If P is of length L, then the cycle $P \cup \{st_1, t_1t_2\}$ is chordless and of length $L+2$. But this contradicts the minimality of C.

Consequently, we have $c = L+2$. Now we are going to show that P_1 (P_2) cannot go through two nodes t_i, t_j, $i,j \in \{1,\ldots,c-1\}$, $i < j$, such that the subpaths of P_1 (P_2) and C between t_i and t_j are edge-disjoint. In fact, suppose, for instance, that the subpaths $\widetilde{P}_1$ of P_1 and $\widetilde{C}$ of C between t_i and t_j are edge-disjoint. Let $\tilde{p}_1$ and $\tilde{c}$ be the lengths of $\widetilde{P}_1$ and $\widetilde{C}$, respectively. We claim that $\tilde{p}_1 = \tilde{c}$. Indeed, if $\tilde{p}_1 > \tilde{c}$, then, by replacing $\widetilde{P}_1$ by $\widetilde{C}$ in P_1, we get a path of length $\leq L-1$ between s and t_1, a contradiction. On the other hand, if $\tilde{c} > \tilde{p}_1$, then, by replacing $\widetilde{C}$ by $\widetilde{P}_1$ in C, we obtain a cycle, say C', of length $\leq L+1$. Because, by Claim 1, F does not contain a cycle of length $\leq L$, it follows that C' is of length exactly $L+1$. Because C_1 and C' have t_1 in common, this contradicts Claim 2.

Therefore $\tilde{p}_1 = \tilde{c}$. But this is still a contradiction because C_1 and the cycle obtained from C_1 by replacing $\widetilde{P}_1$ by $\widetilde{C}$ are both of length $L+1$ and have destination nodes in common.

In consequence, if t_{l_1} (resp., t_{l_2}) is the first node of C met by P_1 (resp., P_2), then P_1 (resp., P_2) contains the subpath of C between t_{l_1} and t_1 (resp., t_{l_2} and t_{c-1}). See Figure 7 for an illustration. Moreover, we have $l_1 < l_2$, for otherwise, the cycles C_1 and C_2 would intersect in some destination nodes, which by Claim 2 is impossible.

Let $d = l_2 - l_1 - 1$. Observe that d is the number of internal nodes of the subpath of C between t_{l_1} and t_{l_2}. Also note that $d < L$. Now let $\overline{\Pi} = (\overline{V}_0, \overline{V}_1, \ldots, \overline{V}_{k-2L-d})$ be the partition of N obtained from Π by gathering the nodes of $C_1 \cup C_2 \cup C$ into $\overline{V}_0$. Note that $|E(\overline{V}_0)| \geq 2(L+1)+d+1$. By the validity of the rooted-partition induced by $\overline{\Pi}$, we have

$$\begin{aligned} |\Delta_{\overline{\Pi}}(G) \cap F| &\geq k - 2L - d + \lceil k - 2L - d/L \rceil, \\ &= k - 2L - d + k/L - 2. \end{aligned}$$

Notice that this remains true even if $k - 2L - d = 0$, that is if $\overline{V}_0 = N$, because in this case $\Delta_{\overline{\Pi}}(G) \cap F$ is empty and the right-hand side of (10) is equal to 0. So,

$$|\Delta_{\Pi}(G) \cap F| \geq |\Delta_{\overline{\Pi}}(G) \cap F| + 2(L+1) + d + 1 \geq k + k/L + 1,$$

a contradiction, which finishes the proof of the claim. □

From Claims 1 and 3, it follows that the only cycles induced by F are of length exactly $L+1$. By Claim 2, these cycles do not contain destination nodes in common. Therefore, F consists of k/L cycles of length $L+1$ and having only s in common. Because each of these cycles uses exactly two edges of $[s,T]$, this yields $|F \cap [s,T]| = 2k/L$, and the proof of the theorem is complete. ∎

Given a graph $G = (N,E)$ and a node subset $V \subseteq N$, we say that V satisfies Property (π) if one of the following holds:

(i) $|V| \geq 4$,
(ii) $|V| = 2,3$ and V contains at least one Steiner node,
(iii) $|V| = 3$, V does not contain Steiner node, and there are parallel edges between at least two pairs of nodes in V,
(iv) $|V| = 2$, V does not contain Steiner node, and $E(V)$ contains parallel edges,
(v) $|V| = 1$.

Theorem 13. *Let $L = 2$, and $G = (N,E)$ be a complete graph with one source node s and at least k destination nodes associated with s. Let $\Pi = (V_0, V_1, \ldots, V_k)$ be a partition of N such that $s \in V_0$ and $t_i \in V_i$ for $i = 1,\ldots,k$.*

Then the rooted-partition inequality (10) defines a facet if and only if

(i) k is odd,
(ii) $|V_i| = 1$ for all $i = 1,\ldots,k$, and
(iii) V_0 satisfies Property (π).

Proof. See [24]. ∎

6. SEPARATION ROUTINES

In this section, we discuss the separation problem for the facet defining inequalities introduced in the previous section. Given a system of inequalities $Ax \leq b$ and a vector $x^* \in \mathbb{R}^{|E|}$, the *separation problem* associated with $Ax \leq b$ consists of verifying whether x^* satisfies $Ax \leq b$, and, if not, in finding an inequality of $Ax \leq b$ violated by x^*. In the sequel, we will denote by $G_{x^*} = (N, E_{x^*})$ the *support graph* of x^*, that is the graph induced by the edges e such that $x^*(e) > 0$.

The st-cut constraints (1) can be separated exactly using the Gomory-Hu algorithm [15]. This produces the so-called

Gomory-Hu tree, which has the property that for all pairs of nodes $s, t \in N$ the minimum st-cut in the tree is also a minimum st-cut in G_{x^*}. To do this, we use the algorithm developed by Gusfield [23], which requires $|N|-1$ maximum flow computations. In practice, to speed up the computation, we first use a simple heuristic to try to quickly find a violated st-cut inequality. This goes as follows. We contract iteratively the edges by decreasing x^* values until either the total value of the shrunk graph $\widetilde{G} = (\widetilde{V}, \widetilde{E})$ is less than its number of nodes, or there only remain two nodes. In the first case, at least one of the nodes of $\widetilde{G}$ induces a cut violated by the restriction of x^* on $\widetilde{E}$. By expanding this node, we obtain a cut in G violated by x^*. In the second case, we check if the cut between the two nodes of $\widetilde{G}$ is violated or not. Of course, in both cases, we verify if the cut obtained separates two nodes of the same demand. If this heuristic is unsuccessful, we then generate the Gomory-Hu tree, using the Gusfield algorithm, to separate the cut constraints exactly.

The L-path-cut inequalities (2) can also be separated in polynomial time when $L = 2, 3$. In fact, for a fixed demand $\{s, t\} \in D$, the separation problem reduces to find a minimum weight edge subset that intersects all L-st-paths. This has been shown to be polynomially solvable in [12]. In practice, when $L = 2$, we do the following for each demand $\{s, t\} \in D$. We consider the partition $\Pi = (V_0, V_1, V_2, V_3)$ with $V_0 = \{s\}$, $V_3 = \{t\}$, and where V_1 and V_2 are constructed as follows. For each node $u \in N \backslash \{s, t\}$, we put u in V_1 if $x(su) \geq x(ut)$, and u in V_2 if not. We then test the violation of the corresponding 2-st-path-cut inequality. This exact separation routine takes into account the particular structure of L-path-cuts when $L = 2$. When $L = 3$, for each demand $\{s, t\} \in D$, we first check with the algorithm given in [12], if the minimum weight edge set cutting all the 3-st-paths of G_{x^*} has size less than 2. If yes, then there is a 3-st-path-cut inequality violated by x^*. That one is then built through a breadth first search from s in G_{x^*}.

Let us now turn our attention to the separation problem for the rooted-partition inequalities (10). We shall prove that this problem for $L = 2$, when k is odd, and the partition sets, except V_0, are singletons, can be solved in polynomial time. Note that these two conditions are necessary for the rooted-partition inequalities to be facet defining; see Theorem 13. As it will turn out, that problem will reduce to minimizing a submodular function on a parity subfamily of a lattice family.

Let M be a finite set. A family C of subsets of elements of M is called a *lattice family* if

$$X \cup Y, X \cap Y \in C, \text{ for all } X, Y \in C.$$

A function $f : C \to \mathbb{R}$ is said to be *submodular* if

$$f(X \cup Y) + f(X \cap Y) \leq f(X) + f(Y), \forall X, Y \in C.$$

A subcollection D of C is called a *parity family* if

$$X \cap Y \in D \iff X \cup Y \in D, \text{ for all } X, Y \in C \backslash D.$$

In [14], Goemans and Ramakrishnan have shown the following result.

Theorem 14 ([14]). *Given a submodular function f on a lattice family C, and a parity family D of C, a set U minimizing $f(U)$ over $U \in D$ can be found in polynomial time.*

Theorem 15. *The separation problem for the rooted-partition inequalities*

$$x(\Delta_\Pi(G)) \geq \lceil 3k/2 \rceil, \tag{25}$$

when $L = 2$, k is odd, and $V_i = \{t_i\}$ for $i = 1, \ldots, k$, can be solved in polynomial time.

Proof. Let $x \in \mathbb{R}^{|E|}$. As k is odd, the separation problem for inequalities (25) is equivalent to

$$\min x(\Delta_\Pi(G)) - 3k/2 - 1/2, \tag{26}$$

over the partitions $\Pi = (V_0, V_1, \ldots, V_k)$ of N such that $V_i = \{t_i\}$ for $i = 1, \ldots, k$. Problem (26) can also be written as

$$\min_{S \subseteq \bigcup_{i=1}^{d} \{t_i\},\, |S| \text{ odd}} x(E(S)) + x(\delta(S)) - 3|S|/2 - 1/2. \tag{27}$$

Now let M be the set N of nodes of G, C be the set of subsets of N contained in $\bigcup_{i=1}^{d} \{t_i\}$, $f(S) = x(E(S)) + x(\delta(S)) - 3|S|/2 - 1/2$, and $D = \{X \in C : |X| \text{ odd }\}$.

It is clear that C is a lattice family, and that D is a parity family. Also, it can be easily seen that f is submodular on C. Consequently, by Theorem 14, the result follows. ∎

As the complexity of the minimization of a submodular function is around $o(n^5)$ [34], using the above algorithm to separate exactly inequalities (10) would be time consuming. In our branch-and-cut algorithm, we will rather use a heuristic separation for the rooted-partition inequalities (10). The heuristic can be described as follows for each source node s with at least two destinations. We consider the nodes that are not destinations of s as Steiner nodes. If $L = 2$, we first look for triangles formed by edges with value 1. If C is such a triangle, then we contract C into a pseudo-node w. If C contains s (resp., a destination node but not s) (resp., only Steiner nodes), then w will take the role of s (resp., a destination node) (resp., a Steiner node) in the new graph. After this eventual step, we consider the Steiner nodes. If a Steiner node u is adjacent to at least one destination node, then we contract the edge that has the highest value, between u and a destination node. The new node is considered as a destination node. If not, then we contract u and s and consider the new node as s. At the end of this procedure, we get a graph without Steiner nodes, and containing, say k, destination nodes $t_1, \ldots, t_k$. This graph gives rise to a partition $V_0, V_1, \ldots, V_k$ of N such that $s \in V_0$ and V_i contains at least a destination node of s, for $i = 1, \ldots, k$. If $L = 3$, then we check whether the rooted-partition induced by this partition is violated. If $L = 2$, then we consider a rooted-opt-partition obtained from $V_0, V_1, \ldots, V_k$ as follows. We consider an edge uv with maximum value such that $u \in V_i$ and $v \in V_j$ for some $i, j \in \{1, \ldots, k\}$, $i < j$, and at least one of the sets $V_i \backslash \{u\}$ and $V_j \backslash \{v\}$ contains at least a destination node.

If $V_i\backslash\{u\}$ (resp., $V_j\backslash\{v\}$) contains a destination, then we consider the rooted-opt-partition induced by $e = uv$ together with the partition $\Pi' = (V'_0, V'_1, \ldots, V'_{k+1})$ given by

$$\begin{aligned}
V'_l &= V_l && \text{for } l = 0, \ldots, i-1,\\
V'_{l-1} &= V_l && \text{for } l = i+1, \ldots, j-1,\\
V'_{l-2} &= V_l && \text{for } l = j+1, \ldots, k,\\
V'_{k-1} &= V_j && (\text{resp., } V_i),\\
V'_k &= V_i\backslash\{u\} && (\text{resp., } V_j\backslash\{v\}),\\
V'_{k+1} &= \{u\} && (\text{resp., } \{v\}).
\end{aligned}$$

We notice that, as shown in the proof of Theorem 13 (ii), this rooted-opt-partition inequality is stronger than the rooted-partition one. We then test the violation of this rooted-opt-partition inequality. Moreover, in both cases ($L = 2$ or $L = 3$), if the tested inequality is not violated, we then contract two sets V_i, V_j such that $[V_i, V_j]$ contains an edge having the largest value in $\Delta_\Pi(G)$, and test this smaller partition. This procedure is stopped whenever the number of partition subsets becomes 2, because the corresponding rooted-partition is then an st-cut.

When $L = 2, 3$, we also separate heuristically the double cut inequalities (6)–(7) and triple path-cut inequalities (8)–(9). In what follows, we present their separation procedures for $L = 2$. Those for $L = 3$ are similar. For the former class, we apply the following for every demand $\{s_1, t_1\} \in D$ and every terminal node t_2 different from s_1, t_1. We consider the partition $\Pi = (V_0, \ldots, V_3)$, where $V_0 = \{s_1\}$, $V_1 = \{t_2\}$, $V_2 = N\backslash\{s_1, t_1, t_2\}$, $V_3 = \{t_1\}$. The idea behind this is to get a double-cut inequality that, by Theorems 8–9, may define a facet of $P_G(D, 2)$. We select $e \in [V_1, V_2]$ having the largest value and then test the violation of the double-cut inequality corresponding to Π and e. (We also test the partition obtained by exchanging the roles of s_1 and t_1.)

For the latter class, the separation procedure can be presented as follows for any source node s with at least two destinations. We look for two destination nodes t_1, t_2 of s such that the triangle s, t_1, t_2 has minimum x^* value. We then consider the partition $\Pi = (V_0, V_1, \ldots, V_4)$ such that $V_0 = \{s\}$, $V_1 = N\backslash\{s, t_1, t_2\}$, $V_2 = \emptyset$, $V_3 = \{t_1\}$, $V_4 = \{t_2\}$. Note that this partition satisfies the necessary conditions (i), (ii), (iii) of Theorem 11. Also, because we consider graphs with at least five nodes, we have $|V_1| \geq 2$. For every node u of V_1, if $x^*(su) \leq 1/2(x^*(ut_1) + x^*(ut_2))$, then we move u from V_1 to V_2, and consider the new partition, still denoted by Π. The motivation behind this is, as before, to reduce as much as possible the left-hand side of the generated inequality. This process is stopped if V_1 has only one node left. If, after this step, V_2 is still empty, then we take a node u from V_1 such that $x^*(su) = \min_{v \in V_1}\{x^*(sv)\}$, and we transfer it to V_2. We then test if the triple path-cut inequality corresponding to Π is violated. The edge e to be removed is chosen as the edge with the maximum x^* value between V_1 and $\{t_1, t_2\}$. If e is incident to t_2, then we exchange the role of t_1 and t_2 for e to belong to $[V_1, V_3]$.

7. BRANCH-AND-CUT AND COMPUTATIONAL RESULTS

Based upon the previous theoretical results, we have developed a branch-and-cut algorithm to solve efficiently the THNDP when $L = 2, 3$. The algorithm has been implemented in C++, using BCP (the Branch, Cut and Price package of the library COIN-OR [31]) to manage the branching tree and CPLEX 8.11 [27] as the linear solver, and tested on a Pentium III at 933 MHz with 384 Mb of RAM under Linux. The maximum run time has been fixed at 5 hours. The results presented here essentially concern the case where the demands are rooted at a node s, because we have focussed our polyhedral study on this case. Nevertheless, our algorithm is adapted to any set of demands, and we also present some computational results in that case. For each instance tested, we have run the algorithm twice, once with the constraints (1)–(4) only, and a second time with also inequalities (5)–(11) depending on their respective validity for $L = 2, 3$.

To begin the optimization, we consider the linear program consisting of the cut inequalities associated with the demand nodes, and the trivial inequalities. Moreover, in the second run of the algorithm, for each source node s with at least two destinations, we add to this basic program the rooted-partition inequality where each destination of s corresponds to an element of the partition and all the other nodes are put in V_0.

In the branch-and-cut algorithm, we have to check whether or not an optimal solution of a relaxation to the THNDP is feasible. An optimal solution x^* of a relaxation is feasible for the THNDP if it is an integer vector satisfying the st-cut and L-st-path-cut inequalities. Verifying if an integer solution x^* is feasible for the THNDP can be done in an efficient way. For every edge e of G_{x^*} and every demand $\{s, t\} \in D$, we check if the shortest st-path in $G_{x^*} - e$ is of length $\leq L$. If this is the case, then by Lemma 1 x^* is feasible. If not, this means there is no L-st-path not containing e. This implies that x^* is not feasible.

Another important issue in the effectiveness of the branch-and-cut algorithm is to compute a good upper bound. For this, in the solution of the current linear program, we first round up to 1 all the variables with a value ≥ 0.3 and round down to 0 those with value < 0.3. We try to improve this solution by deleting all the edges incident to a Steiner node whose total value is less than 2. We then verify if the resulting integer solution is feasible. If not, we apply the same procedure for the solution obtained by rounding up to 1 all the fractional values. This latter solution is certainly a feasible solution.

If an optimal solution x^* of the linear relaxation of the THNDP is not feasible, the algorithm generates additional valid inequalities of $P_G(D, L)$ violated by x^*. Their separation is realized in the following order:

1. st-cut inequalities,
2. L-path-cut inequalities,
3. double cut inequalities,
4. rooted-(opt-)partition inequalities (if $L = 2$),
5. triple path-cut inequalities.

We remark that all inequalities are global (i.e., valid at every node of the branch-and-cut tree) and several of them can be added at each iteration. Moreover, we go to the next class only if we do not find any violated inequalities in the current class.

To separate the different inequalities, we use the algorithms described in Section 6. All our separation procedures are applied on the support of x^*, that is G_{x^*}, where x^* is the solution of the current relaxation. When solving instances of the THNDP, we observed that the exact separation of the st-cut inequalities is time consuming. Therefore, we decided to perform this exact separation after that of the L-path-cut inequalities.

To store the generated inequalities, we use a pool whose size increases dynamically. Inequalities in the pool can be removed from the current linear program when they are not active. Also, they are the first inequalities to be separated. If all the inequalities in the pool are satisfied by the current solution, we then separate the classes of inequalities according to the order given above.

The computational results presented here concern randomly generated instances and instances coming from real applications. The instances consist of complete graphs with edge costs equal to rounded Euclidean distances. The tests were performed for $L = 2, 3$. In practice, note that the bound on the routing paths does not usually exceed 4. The second set of instances comes from the network of the Belgian telecommunications operator, Belgacom, on 52 cities and subsets of these (the demands related to these insatnces are randomly generated). The random problems were generated with $n = 10$ to $n = 40$ nodes, with different number d of demands. For each couple (n, d), five instances were tested.

In the various tables, the entries are:

- n : the number of nodes of the problem,
- d : the number of demands,
- Cu : the number of generated st-cut inequalities (run 2),
- Pc : the number of generated L-path-cut inequalities (run 2),
- Dc : the number of generated double cut inequalities (run 2),
- Ro : the number of generated rooted-(opt-)partition (if $L = 2$) inequalities (run 2),
- Tp : the number of generated triple path-cut inequalities (run 2),
- o1 : the number of problems solved to optimality (run 1) over the five instances tested,
- o2 : the number of problems solved to optimality (run 2) over the five instances tested,
- Gap1 : the gap between the best upper bound and the lower bound obtained at the root node of the branch-and-cut tree in the first run,
- Gap2 : the gap between the best upper bound and the lower bound obtained at the root node of the branch-and-cut tree in the second run,
- Gt1 : the gap between the best upper bound and the best lower bound obtained in the first run,
- Gt2 : the gap between the best upper bound and the best lower bound obtained in the second run,
- Tree1 : the number of nodes in the branch-and-cut tree for the first run,
- Tree2 : the number of nodes in the branch-and-cut tree for the second run,
- CPU1 : the total time of the first run in seconds,
- CPU2 : the total time of the second run in seconds.

The first two tables report the average results for the random instances, obtained in the case of rooted demands, for $L = 2$ and $L = 3$, respectively.

In Table 1, we remark that, up to 20 nodes and 15 demands, all problems have been solved to optimality within the time limit. With one exception, this is also the case for 30 (resp., 40) nodes when there are 15 (resp., 10) destinations or fewer. When we consider 20 demands or more, only one instance has been solved in less than 5 hours. We remark that not many st-cut inequalities are obtained for the different instances. This, in fact, is because, when $L = 2$, the L-st-path-cut inequalities dominate the st-cut constraints induced by nonsingletons [26]. Note that, in the second run, a significant number of double cut, rooted-opt-partition, and triple path-cut inequalities have been generated. If these yield little impact on the number of instances solved to optimality, or on the CPU time, we remark that the gaps (Gap2, Gt2) and, in particular, the size

TABLE 1. Results for random instances with $L = 2$ and rooted demands.

n	d	Cu	Pc	Dc	Ro	Tp	o1	o2	Gap1	Gap2	Gt1	Gt2	Tree1	Tree2	CPU1	CPU2
10	2	0	4	2	2	2	5	5	0.0	0.0	0.0	0.0	1	1	0.01	0.01
10	5	1	24	5	9	5	5	5	8.0	1.1	0.0	0.0	17	5	0.10	0.06
10	8	3	70	20	26	8	5	5	12.0	5.0	0.0	0.0	156	34	1.30	0.54
20	5	0	36	5	10	10	5	5	6.6	0.9	0.0	0.0	10	2	0.12	0.09
20	10	5	480	65	311	25	5	5	13.5	6.8	0.0	0.0	1143	547	32.02	40.88
20	15	13	3077	371	973	59	5	5	15.1	9.2	0.0	0.0	79,054	35,770	6711.73	7081.31
30	8	2	249	22	273	26	5	5	8.9	3.3	0.0	0.0	152	51	5.05	15.19
30	15	8	2465	207	1491	82	4	4	12.8	7.4	0.2	0.7	28,501	7694	5023.44	5215.93
30	22	17	2982	327	1554	40	1	1	30.5	20.2	22.6	15.7	73,564	26,235	14,591.42	14,432.96
40	10	3	1714	87	2972	84	4	4	13.2	7.4	0.1	1.0	10,458	2139	3696.43	4426.79
40	20	13	3252	213	3379	56	0	0	33.4	24.2	25.6	20.7	60,744	10,978	18,000.00	18,000.00
40	30	26	3191	234	1626	19	0	0	43.3	36.3	39.1	34.1	53,753	12,859	18,000.00	18,000.00

TABLE 2. Results for random instances with $L = 3$ and rooted demands.

n	d	Cu	Pc	Dc	Ro	Tp	o1	o2	Gap1	Gap2	Gt1	Gt2	Tree1	Tree2	CPU1	CPU2
10	2	5	0	2	3	1	5	5	0.0	0.0	0.0	0.0	1	1	0.01	0.01
10	5	11	53	10	14	3	5	5	11.3	5.6	0.0	0.0	58	17	0.70	0.21
10	8	9	198	20	8	2	5	5	12.3	6.4	0.0	0.0	278	89	7.26	1.50
20	5	28	146	13	45	2	5	5	8.3	5.2	0.0	0.0	37	17	2.25	1.35
20	10	30	3407	52	312	9	5	5	13.9	6.8	0.0	0.0	3681	1247	2403.97	437.00
20	15	38	24,953	170	422	21	0	0	30.6	16.1	21.8	8.6	10,049	11,971	18,000.00	18,000.00
30	8	30	2250	42	384	1	5	5	10.3	5.3	0.0	0.0	996	336	1927.48	468.81
30	15	60	18,509	122	1071	5	0	1	29.9	16.4	21.9	9.9	5326	5087	18,000.00	16,647.14
30	22	39	15,703	105	318	4	0	0	42.8	30.4	38.5	27.7	3859	5753	18,000.00	18,000.00
40	10	58	9351	71	1087	4	2	3	16.7	7.4	6.6	1.7	1965	1099	11,925.39	8710.00
40	20	61	11,442	76	940	1	0	0	46.0	36.7	42.4	34.7	1808	1930	18,000.00	18,000.00
40	30	49	11,344	85	274	4	0	0	57.3	45.7	55.1	44.4	1525	1547	18,000.00	18,000.00

TABLE 3. Results for random instances with $L = 2$ and arbitrary demands.

n	d	Cu	Pc	Dc	Ro	Tp	o1	o2	Gap1	Gap2	Gt1	Gt2	Tree1	Tree2	CPU1	CPU2
10	5	3	59	21	10	20	5	5	10.2	8.8	0.0	0.0	21	21	0.16	0.24
20	5	3	76	14	7	20	5	5	4.7	3.8	0.0	0.0	7	9	0.22	0.40
20	10	7	761	52	20	102	5	5	8.3	7.8	0.0	0.0	625	358	50.58	45.19
20	15	11	5898	191	60	949	3	3	15.8	14.3	5.9	4.8	13,178	9346	8536.60	8314.82
30	8	5	650	66	56	154	5	5	8.5	8.5	0.0	0.0	459	489	28.61	43.40
30	15	13	5397	412	119	910	2	2	12.5	12.7	3.9	4.6	23,291	15,357	11,748.40	11,613.36
30	22	16	6426	164	5	266	0	0	41.0	45.8	38.1	43.2	4159	3668	18,000.00	18,000.00
40	10	6	519	50	15	73	5	5	5.4	5.0	0.0	0.0	107	63	15.28	16.44
40	20	13	6565	363	15	736	0	0	17.0	19.3	11.4	14.0	10,859	8349	18,000.00	18,000.00
40	30	17	5273	69	1	118	0	0	46.8	46.8	45.1	45.1	913	888	18,000.00	18,000.00

TABLE 4. Results for random instances with $L = 3$ and arbitrary demands.

n	d	Cu	Pc	Dc	Ro	Tp	o1	o2	Gap1	Gap2	Gt1	Gt2	Tree1	Tree2	CPU1	CPU2
10	5	9	68	21	4	2	5	5	7.0	5.8	0.0	0.0	29	22	0.35	0.34
20	5	20	565	46	6	0	5	5	8.5	7.7	0.0	0.0	103	91	12.12	10.01
20	10	31	20,280	395	7	3	2	2	20.7	18.5	8.0	6.7	5042	4369	12,555.41	11,183.45
20	15	27	28,657	584	15	5	0	0	44.6	41.7	38.0	34.9	4326	4484	18,000.00	18,000.00
30	8	50	14,251	282	40	4	3	3	15.7	14.0	3.1	2.1	1948	1816	8489.24	8181.75
30	15	29	19,982	318	4	1	0	0	56.7	56.4	53.7	53.5	1126	1190	18,000.00	18,000.00
30	22	30	16,097	236	2	0	0	0	66.1	67.0	64.6	65.5	601	626	18,000.00	18,000.00
40	10	64	16,501	279	9	1	0	0	30.1	33.5	23.8	28.2	1605	1709	18,000.00	18,000.00
40	20	29	12,519	148	4	1	0	0	61.5	61.4	60.2	60.0	274	301	18,000.00	18,000.00
40	30	23	9313	60	2	0	0	0	69.3	69.3	68.7	68.7	79	79	18,000.00	18,000.00

TABLE 5. Results for real instances when $L = 2$.

n	d	Cu	Pc	Dc	Ro	Tp	Gap1	Gap2	Gt1	Gt2	Tree1	Tree2	CPU1	CPU2
11	10	0	37	25	28	3	9.9	3.3	0.0	0.0	201	29	1.83	0.59
11	30	0	793	4	48	49	17.3	8.9	0.0	0.0	135	95	7.86	10.67
11	55	1	2376	8	51	51	20.9	10.1	0.0	0.0	245	235	118.30	88.88
30	10	4	241	35	302	27	9.9	3.4	0.0	0.0	215	101	5.19	18.58
30	30	2	1862	7	347	67	17.3	8.9	0.0	0.0	103	87	35.29	94.78
30	55	0	8767	4	610	98	20.9	10.1	0.0	0.0	227	321	463.32	859.75
52	10	1	613	48	1041	59	9.9	4.5	0.0	0.0	641	315	35.30	226.05
52	30	0	4184	13	1239	156	17.3	9.2	0.0	0.0	161	235	241.0	1130.67
52	55	0	13,540	3	6920	90	20.9	11.1	0.0	7.4	221	1051	3119.40	18,000.00

TABLE 6. Results for real instances when $L = 3$.

n	d	Cu	Pc	Dc	Ro	Tp	Gap1	Gap2	Gt1	Gt2	Tree1	Tree2	CPU1	CPU2
11	10	10	135	19	3	0	11.2	2.4	0.0	0.0	869	81	21.82	1.13
11	30	11	2887	234	14	5	16.2	8.8	0.0	0.0	4627	891	1655.84	126.39
11	55	11	3485	351	9	0	16.1	7.3	0.0	0.0	3767	721	2741.18	196.13
30	10	3	43	3194	50	339	11.2	3.8	0.0	0.0	3235	603	3678.49	348.73
30	30	14	24,111	423	659	44	17.1	8.9	5.3	1.8	2233	3115	18,000.00	18,000.00
30	55	16	25,376	512	576	45	53.2	7.6	46.8	1.0	1605	2361	18,000.00	18,000.00
52	10	51	14,309	68	1328	9	11.3	7.3	3.2	2.6	1643	2221	18,000.00	18,000.00
52	30	31	14,685	80	535	4	21.5	12.2	13.9	6.6	449	407	18,000.00	18,000.00
52	55	42	11,950	103	1234	11	53.5	13.7	49.1	10.6	383	387	18,000.00	18,000.00

of the branch-and-cut tree (Tree2) are significantly reduced with respect to Gap1, Gt1, and Tree1. We may also observe that for the instances with 20 demands or more, the gap at the root node in the second run (Gap2) is better than the final gap obtained after 5 hours in the first run (Gt1).

Table 2 gives the computational results for the same instances as those of Table 1, when $L = 3$.

Similar observations as for Table 1 can be made for Table 2. However, the improvement between the first and second runs is even more important. Indeed, one can remark that, for the instances with $(n, d) = (10, 8)$, $(20, 10)$, or $(30, 8)$, the CPU time is almost divided by 5. This can be explained by the fact that the *st*-cut inequalities when $L = 2$ have been all generated by our heuristic, while, for $L = 3$, the exact separation, which takes more time, has been used. For this, it is natural to come with more exciting results in run 1 when $L = 2$ than in run 1 when $L = 3$. Moreover, for the instances with $(n, d) = (30, 15)$ or $(40, 10)$, we can see that one more instance has been solved to optimality in run 2. Finally, the gap Gt2 has been reduced for the big instances by almost 10% with respect to Gt1.

Tables 3 and 4 summarize the computational results obtained for the same instances, but with arbitrary sets of demands.

Unfortunately, for Tables 3 and 4, no significant improvement between the first and second runs is observed. We believe that this is because the valid inequalities presented in this paper are more adapted to rooted demands. However, many instances in both cases have been solved to optimality. For $L = 2$, all the instances with up to 10 demands are solved to optimality. For $L = 3$, this is the case for up to five demands only. Also note that, as the triple-path-cut inequalities never define facets for $L = 3$, they are generated only a few times in this case. However, as we may see, when $L = 2$, they are generated more.

In Tables 5 and 6, we report the computational results obtained for real instances when $L = 2$ and $L = 3$, respectively. Here, the instances with 10 demands are rooted, while the others have multiples sources.

Consider first Table 6. We can remark that, for the instances solved to optimality, the CPU time in the first run has been divided by more than 10 in the second run, and for the other instances, the final gap Gt2 has been considerably reduced compared to Gt1. The most significant case is the instance with 30 nodes and 55 demands. The final gap decreases from 46.8 to 1%. However, quite surprisingly, in Table 5, for the same instances with $L = 2$, the second run does not permit us to improve either the CPU time, or the final gap. Only the gap at the root node decreases between the two runs.

8. CONCLUDING REMARKS

In this article, we have studied the two-edge connected hop-constrained network design problem (THNDP). We have proved that the problem remains strongly *NP*-hard even in the rooted case and $L = 2$. We have given an integer programming formulation of the THNDP when $L = 2, 3$, and described various families of valid inequalities. We have then focussed on the rooted case and $L = 2, 3$, where we have studied necessary conditions and sufficient conditions for these inequalities to be facet defining. We have also discussed separation routines for the different classes of inequalities. In particular, for the rooted-partition inequalities, when the elements of the partition, different from the one containing the source node, are singletons, we have shown that the associated separation problem can be reduced to the minimization of a submodular function, and hence, can be solved in polynomial time.

Using our polyhedral results, we have devised a branch-and-cut algorithm for $L = 2, 3$, and presented some computational results. We could estimate the effect of the double cut, the rooted-(opt-)partition and the triple path-cut inequalities in the branch-and-cut algorithm. This is particularly strong for the rooted case because these inequalities are more adapted to this case. For arbitrary demands, the best results have been realized for the real instances when $L = 3$. We could also measure the performance of our separation techniques.

It would be interesting to investigate the polyhedral aspect of the THNDP when the demand set contains multiple sources. The case where $L = 4$ is of particular interest and also merits study.

Acknowledgments

Part of this work was done while the third author was visiting GOM, Free University of Brussels in 2004. We thank the anonymous referees and the Guest Editor for their constructive comments.

REFERENCES

[1] A. Balakrishnan and K. Altinkemer, Using a hop-constrained model to generate alternative communication network design, ORSA J Comput 4 (1992), 192–205.

[2] W. Ben-Ameur, Constrained length connectivity and survivable networks, Networks 36 (2000), 17–33.

[3] W. Ben-Ameur and E. Gourdin, Internet routing and related topology issues, SIAM J Discrete Math 17 (2003), 18–49.

[4] J.A. Bondy and U.S.R. Murty, Graph theory with applications, Universities Press, Belfast, 1976.

[5] C.R. Coullard, A.B. Gamble, and J. Liu, "The k-walk polyhedron," Advances in optimization and approximation, D.-Z. Du and J. Sun (Editors), Kluwer Academic Publishers, New York, 1994, pp. 9–29.

[6] G. Dahl, The 2-hop spanning tree problem, Oper Res Lett 23 (1998), 21–26.

[7] G. Dahl, Notes on polyhedra associated with hop-constrained paths, Oper Res Lett 25 (1999), 97–100.

[8] G. Dahl, N. Foldnes, and L. Gouveia, A note on hop-constrained walk polytopes, Oper Res Lett 32 (2004), 345–349.

[9] G. Dahl and L. Gouveia, On the directed hop-constrained shortest path problem, Oper Res Lett 32 (2004), 15–22.

[10] G. Dahl, D. Huygens, A.R. Mahjoub, and P. Pesneau, On the k-edge disjoint 2-hop-constrained paths polytope, 2005, Oper Res Lett 34 (2006), 577–582.

[11] G. Dahl and B. Johannessen, The 2-path network problem, Networks 43 (2004), 190–199.

[12] B. Fortz, A.R. Mahjoub, S.T. McCormick, and P. Pesneau, Two-edge connected subgraphs with bounded rings: Polyhedral results and branch-and-cut, Math Prog 105 (2006), 85–111.

[13] M.R. Garey and D.S. Johnson, Computers and intractability: A guide to the theory of NP-completeness, Freeman, New York, 1979.

[14] M.X. Goemans and V.S. Ramakrishnan, Minimizing submodular functions over families of sets, Combinatorica 15 (1995), 499–513.

[15] R.E. Gomory and T.C. Hu, Multi-terminal network flows, SIAM J Appl Math 9 (1961), 551–570.

[16] L. Gouveia, Multicommodity flow models for spanning trees with hop constraints, Eu J Oper Res 95 (1996), 178–190.

[17] L. Gouveia, Using variable redefinition for computing lower bounds for minimum spanning and Steiner trees with hop constraints, INFORMS J Comput 10 (1998), 180–188.

[18] L. Gouveia and E. Janssen, Designing reliable tree networks with two cable technologies, Eur J Oper Res 105 (1998), 552–568.

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

[20] L. Gouveia, T. Magnanti, and C. Requejo, A 2-path approach for odd-diameter-constrained minimum spanning and Steiner trees, Networks 44 (2004), 254–265.

[21] L. Gouveia, P. Patricio, and A. de Sousa, "Compact models for hop-constrained node survivable network design, an application to MPLS," Telecommunications planning: Innovations in pricing, network design and management, S. Raghavan and G. Anandalingam (Editors), Vol. 33, Springer, Berlin, 2006, pp. 167–180.

[22] L. Gouveia and C. Requejo, A new Lagrangean relaxation approach for the hop-constrained minimum spanning tree problem, Eur J Oper Res 132 (2001), 539–552.

[23] D. Gusfield, Very simple methods for all pairs network flow analysis, SIAM J Comput 19 (1990), 143–155.

[24] D. Huygens, M. Labbé, A. Mahjoub, and P. Pesneau, The 2-edge connected hop-constrained network design problem: Valid inequalities and branch-and-cut, Tech. Rep. RR-06:02, LIMOS, Université Blaise Pascal, Clermont-Ferrand, France, 2006.

[25] D. Huygens and A.R. Mahjoub, Integer programming formulations for the two 4-hop-constrained paths problem, Preprint, 2005.

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

[27] ILOG, S.A., ILOG Cplex 8.11: User's Manual and Reference Manuals, 2003. http://www.ilog.com/.

[28] H. Kerivin and A.R. Mahjoub, Design of survivable networks: A survey, Networks 46 (2005), 1–21.

[29] L.J. Leblanc and R. Reddoch, Reliable link topology/capacity and routing in backbone telecommunication networks, Working paper 90-08, Owen Graduate School of Management, Vanderbilt University, Nashville, TN, 1990.

[30] L.J. Leblanc, R. Reddoch, J. Chifflet, and P. Mahey, Packet routing in telecommunications networks with path and flow restrictions, INFORMS J Comput 11 (1999), 188–197.

[31] R. Lougee-Heimer, The common optimization interface for operations research, IBM J Res Dev 47 (2003), 57–66.

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

[33] W.R. Pulleyblank, "Polyhedral combinatorics," Optimization, Handbooks Oper Res Management Sci 1, G.L. Nemhauser, A.H.G. Rinnooy Kan, and M.J. Todd (Editors), North-Holland, Amsterdam, 1989, pp. 371–446.

[34] A. Schrijver, Combinatorial optimization: Polyhedra and efficiency, Volume B, Matroids, trees, stable sets, Chapters 39–69, Springer, Berlin, 2003.