Two Node-Disjoint Hop-Constrained Survivable Network Design and Polyhedra

Ibrahima Diarrassouba
Laboratoire LMAH, Université du Havre, Le Havre, France

Hakan Kutucu
Department of Computer Engineering, Karabuk University, Karabuk, Turkey

A. Ridha Mahjoub
Laboratoire LAMSADE, Université Paris-Dauphine, Paris, France

Given a weighted undirected graph G with a set of pairs of terminals (s_i, t_i), $i = 1, \ldots, d$, and an integer $L \geq 2$, the two node-disjoint hop-constrained survivable network design problem is to find a minimum weight subgraph of G such that between every s_i and t_i there exist at least two node-disjoint paths of length at most L. This problem has applications in the design of survivable telecommunication networks with QoS-constraints. We discuss this problem from a polyhedral point of view. We present several classes of valid inequalities along with necessary and/or sufficient conditions for these inequalities to be facet defining. We also discuss separation routines for these classes of inequalities, and propose a Branch-and-Cut algorithm for the problem when $L = 3$, as well as some computational results.
NETWORKS, Vol. 67(4), 316–337 2016

Keywords: survivable network; node-disjoint paths; hop constraint; polyhedron; facet; Branch-and-Cut

1. INTRODUCTION

Consider an undirected graph $G = (N, E)$, an integer $L \geq 2$, and a set of demands $D \subseteq N \times N$. Each demand is an ordered pair (s, t) of nodes, $s \neq t$. Node s is referred to as the *source* of the demand and t its *destination*. The *Two Node-Disjoint Hop-Constrained Survivable Network Design Problem* (TNHNDP for short) consists of finding a minimum weight subgraph of G containing at least two node-disjoint L-st-paths, that is, paths between s and t with at most L edges (also called hops), between each pair of nodes $(s, t) \in D$. When the demand set consists of one source and several destinations, the problem is called the *Rooted Two Node-Disjoint Hop-Constrained Survivable Network Design Problem* (rooted TNHNDP).

In [13], Gouveia et al. discuss the TNHNDP within the context of an MPLS (Multi-Protocol Label Switching) network design model. They propose two extended formulations involving one set of variables associated with each path between each pair of demand nodes. The first model uses standard flow variables, and the second uses hop-indexed variables. Each subsystem of constraints associated with a path is a flow model with additional cardinality constraints. The authors also introduce a third model involving one set of hop-indexed variables for each pair of terminals. They show that this aggregated and more compact model produces the same linear programming bound as the multipath hop-indexed model. They also present computational results for $L \in \{4, 5, 6\}$ using these formulations. Unfortunately, as the number of variables of the resulting models grows with L (and the number of pairs of terminals), the size of the corresponding linear programming relaxation may lead to excessive computational time when more dense instances (or instances with a larger value of L or a larger number of nodes) are considered. As mentioned in [13], this points out the need for formulations using only natural variables.

The edge version of the problem (TEHNDP), when we look for two edge-disjoint hop-constrained paths instead of two node-disjoint paths, has already been investigated by several authors when $L \in \{2, 3\}$. In particular, Huygens et al. [15] study the TEHNDP when $|D| \geq 2$. They show that the TEHNDP is strongly NP-Hard for any $L \geq 2$ and $|D| \geq 2$ by reducing any instance of the dominating set problem to an instance of the TEHNDP. They also devise a Branch-and-Cut algorithm for the problem and present some computational results for $L \in \{2, 3\}$. Bendali et al. [3] and Diarrassouba

Received November 2013; revised March 2016; accepted March 2016
Correspondence to: I. Diarrassouba; e-mail: diarrasi@univ-lehavre.fr
Contract grant sponsor: TUBITAK-BIDEB fellowship
DOI 10.1002/net.21679
Published online 6 April 2016 in Wiley Online Library (wileyonlinelibrary.com).

[6] consider the more general case when k edge-disjoint hop-constrained paths are required, for any $k \geq 1$. Bendali et al. [3] give a complete and minimal description of the polytope associated with the problem when $L = 3$ and $|D| = 1$. Diarrassouba et al. [7] (see also [6]) consider the problem when $k \geq 1$, $|D| \geq 2$ and $L \in \{2, 3\}$. They present several integer programming formulations for the problem and devise a Branch-and-Cut algorithm for the problem in this latter case.

Huygens and Mahjoub [14] study the two versions of the problem (TNHNDP and TEHNDP) when $L = 4$. They give an integer programming formulation for the problem in the two cases. In [4], Dahl considers the hop-constrained path problem, namely the problem of finding a minimum weight path between two distinguished nodes s and t 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 [5] consider the directed hop-constrained path problem. They describe valid inequalities and characterize the associated polytope when $L \leq 3$.

As mentioned above, Huygens et al. [15] have shown that the TEHNDP is strongly NP-Hard for the case $L \geq 2$ and $|D| \geq 2$. They establish this result by reducing the vertex dominating set problem to the TEHNDP. The reduction, as well as the related results, is still valid for the case of node-disjoint paths. In fact, the solutions of the TEHNDP obtained from the reduction are at the same time solutions of the TNHNDP. Thus, as the TEHNDP, the TNHNDP is also strongly NP-Hard for $L \geq 2$ and $|D| \geq 2$.

One may notice that in the case where $L = 2$ and the graph is simple (it does not contain parallel edges), the TNHNDP reduces to the TEHNDP. Indeed, in this case, an edge set $F \subseteq E$ induces a solution for the TNHNDP if and only if it induces a solution for the TEHNDP. To see this, first notice that any two node-disjoint st-paths of G, for every $(s, t) \in D$, are also edge-disjoint. Thus any solution of the TNHNDP in G is also solution of the TEHNDP. Now as G is simple, any two edge-disjoint 2-st-paths of G are of the form (s, u, t) and (s, v, t), with $u \neq v$, for every $(s, t) \in D$. Clearly, these two paths are also node-disjoint and hence, any solution of the TEHNDP, when $L = 2$ and G is simple, is also solution of the TNHNDP.

When $L = 3$, the above observation is no longer valid. We may find, for some graphs, solutions of the TEHNDP with $L = 3$ that are not solutions of the TNHNDP. To see this, consider the graph G shown in Figure 1. In this graph, the paths (s, u, v, t) and (s, v, w, t) form two edge-disjoint st-paths of length 3, and hence, G induces a solution of the TEHNDP. However, these two paths are clearly not node-disjoint and G does not contain any pair of node-disjoint st-paths. Hence, G is not solution of the TNHNDP.

Following this observation, we focus in this work only on the case where $L = 3$. We consider the polytope associated with the TNHNDP when $L = 3$ and give some classes of valid inequalities along with necessary and/or sufficient conditions for these inequalities to define facets. We also devise separation procedures for these inequalities. Using these results, we develop a Branch-and-Cut algorithm for the problem and discuss some computational results.

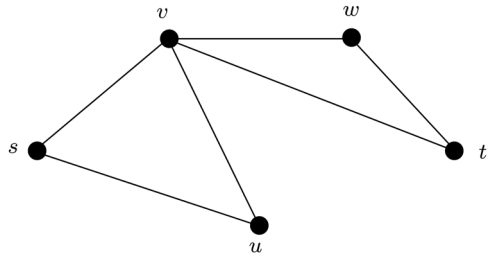

FIG. 1. A solution of the TEHNDP which is not solution of the TNHNDP for $L = 3$.

The paper is organized as follows. We first present in Section 2 the so-called natural formulation for the TNHNDP. Then, in Section 3, we give some classes of inequalities that are valid for the TNHNDP polytope. Necessary and sufficient conditions for these inequalities to be facet defining are discussed in Section 4. In Section 5, we discuss separation procedures for these inequalities and devise a Branch-and-Cut algorithm for solving the problem. In Section 6, we present some computational results for the problem and finally, give some concluding remarks in Section 7. The remainder of this section is devoted to more definitions and notation that will be used throughout the article.

Let $G = (N, E)$ be an undirected graph. Given a node $z \in N$, we denote by $G - z$ the subgraph obtained from G by deleting node z and all its incident edges (but not their other end nodes). If $W \subset N$ is a node subset of G, then the set of edges having only one end node in W is the *cut* induced by W and is denoted by $\delta(W)$. We write $\delta(v)$ for $\delta(\{v\})$. In the case where the graph is not clear from the context, we use $\delta_G(W)$, for a given node set $W \subset N$. For two nodes $s, t \in V$, an *st-cut* is a cut induced by a node set W such that $s \in W$ and $t \in N \setminus W$. For two disjoint node sets $U, V \subset N$, we denote by $[U, V]$ the set of edges of G having one node in U and the other in V. We write $[u, V]$ instead of $[\{u\}, V]$, where $u \in N$.

Given a partition $(V_0, \ldots, V_p)$, $p \geq 1$, of N, the set of edges having their end nodes in different sets of the partition is called the *multicut* induced by the partition and is denoted by $\delta(V_0, \ldots, V_p)$.

A directed graph will be denoted by $H = (U, A)$ with U the set of nodes and A the set of arcs. For any node subset $W \subset U$, the *st-dicut* induced by W, denoted by $\delta^+(W)$, is the set of arcs whose origin is in W and tail is in $U \setminus W$. As for the undirected case, if the digraph is not clear from the context, we write $\delta^+_H(W)$.

Recall that an *st-path* is a path between two nodes $s, t \in N$. An *L-st-path* is an st-path whose length, in terms of number of edges, is at most L.

Also, recall that a *terminal node* is a node involved in at least one demand. The set of terminal nodes of G will be denoted by R. Nonterminal nodes, those not involved in any demand, will be called *steiner nodes*.

Finally, given 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

the *incidence vector* of F and, given a vector $w \in \mathbb{R}^E$, we let $w(F) = \sum_{e \in F} w(e)$.

2. NATURAL FORMULATION FOR THE TNHNDP

In this section, we present an integer programming formulation for the TNHNDP for $L \in \{2, 3, 4\}$. As we will see later, this formulation is no longer valid for $L \geq 5$. Also, this formulation only uses the so-called design variables, where each variable corresponds to an edge of the input graph G.

It is clear that the incidence vector x^F of an edge set $F \subseteq E$ inducing a solution of the TNHNDP satisfies the following inequalities

$$x(\delta(W)) \geq 2, \quad \text{for all } st\text{-cuts } \delta(W) \text{ and } (s,t) \in D. \tag{2.1}$$

Inequalities (2.1) are the so-called *st-cut inequalities*. They ensure that the subgraph induced by F contains at least two edge-disjoint st-paths for all $(s,t) \in D$.

In [4], Dahl introduced a class of valid inequalities for the hop-constrained shortest st-path problem. For two nodes $s, t \in V$, $s \neq t$, let $(V_0, \ldots, V_{L+1})$ be a partition of N with $s \in V_0$, $t \in V_{L+1}$, and $V_i \neq \emptyset$ for all $i \in \{1, \ldots, L\}$. The set T of edges $uv \in E$ such that $u \in V_i$, $v \in V_j$, and $|i - j| > 1$, that is,

$$T = \delta(V_0, \ldots, V_{L+1}) \setminus \bigcup_{i=0}^{L} [V_i, V_{i+1}],$$

is called an *L-st-path-cut*. Figure 2 gives an illustration for $L = 3$. Notice that an L-st-path-cut intersects each L-st-path in at least one edge.

Dahl [4] showed that the inequality

$$x(T) \geq 1$$

is valid for the L-st-path polyhedron. These inequalities are called *L-st-path-cut inequalities*.

Using similar type of partitions, these inequalities can be generalized in a straightforward way to the TNHNDP polytope as

$$x(T) \geq 2, \text{ for every } L\text{-}st\text{-path-cut } T \text{ of } G, \quad \text{for any } (s,t) \in D. \tag{2.2}$$

Inequalities of type (2.2) will also be called *L-st-path-cut inequalities*. Inequalities (2.2) together with (2.1) ensure that the subgraph induced by a solution contains at least two edge-disjoint paths of length $\leq L$.

Inequalities (2.1) and (2.2) can be extended to the case of node-disjoint paths. For the node case, the following similar inequalities are also needed to ensure that the paths are node-disjoint

$$x(\delta_{G-z}(W)) \geq 1, \text{ for all } st\text{-cuts } \delta_{G-z}(W), \quad z \in N \setminus \{s,t\} \text{ and } (s,t) \in D, \tag{2.3}$$

$$x(T_{G-z}) \geq 1, \text{ for all } L\text{-}st\text{-path-cuts } T_{G-z} \text{ of } G - z, \quad z \in N \setminus \{s,t\} \text{ and } (s,t) \in D. \tag{2.4}$$

Inequalities (2.3) and (2.4) are called respectively *st-node-cut* and *L-st-node-path-cut inequalities*. Together with inequalities (2.1) and (2.2), they ensure that there exist at least two node-disjoint L-st-paths in the subgraph induced by an edge set F.

Finally, the incidence vector of an edge F inducing a solution of the TNHNDP satisfies

$$x(e) \leq 1, \text{ for all } e \in E, \tag{2.5}$$

$$x(e) \geq 0, \text{ for all } e \in E. \tag{2.6}$$

Inequalities (2.5) and (2.6) are called *trivial inequalities*.

Huygens and Mahjoub [14] showed that when $|D| = 1$ and $L \in \{2, 3, 4\}$, inequalities (2.1)–(2.6) together with the integrality constraints formulates the TNHNDP. This result can be easily extended to the case where $|D| \geq 2$. An edge set F induces a solution of the TNHNDP in this case if and only if x^F satisfies the following inequalities

$$\begin{aligned}
x(\delta(W)) &\geq 2, \text{ for all } st\text{-cuts } \delta(W) \text{ and } (s,t) \in D,\\
x(\delta_{G-z}(W)) &\geq 1, \text{ for all } st\text{-cuts } \delta_{G-z}(W), \quad z \in N \setminus \{s,t\} \text{ and } (s,t) \in D,\\
x(T) &\geq 2, \text{ for all } L\text{-}st\text{-path-cuts } T \text{ and } (s,t) \in D,\\
x(T_{G-z}) &\geq 1, \text{ for all } L\text{-}st\text{-path-cuts } T_{G-z} \text{ of } G - z, \quad z \in N \setminus \{s,t\} \text{ and } (s,t) \in D,\\
x(e) &\leq 1, \text{ for all } e \in E,\\
x(e) &\geq 0, \text{ for all } e \in E.
\end{aligned}$$

We thus have the following result.

Theorem 1. *The TNHNDP when $L \in \{2, 3, 4\}$ is equivalent to the integer program*

$$min \left\{ wx : x \text{ satisfies } (2.1) - (2.6),\ x \in \mathbb{Z}^E \right\}. \tag{2.7}$$

Formulation (2.7) is called *natural formulation*. As we can see, it only involves design variables. The *TNHNDP polytope*, denoted by TNHNDP(*G,L*), is the convex hull of the solutions to program (2.7).

We will also call inequalities (2.1)–(2.4) the *natural inequalities*. Here "natural" means that they are necessary in the natural formulation of the problem.

As mentioned before, formulation (2.7) is no longer valid for $L \geq 5$. Consider for example the graph shown in Figure 3.

As we can see, its incidence vector satisfies inequalities (2.1)–(2.4) but, unfortunately, the graph does not contain two node-disjoint st-paths of length at most $L = 5$. This example is given in [14].

Also, observe that the linear relaxation of formulation (2.7) is not integral when $L = 3$. To see this, consider the graph $G = K_5$ in Figure 4, where the variables corresponding to edges in solid lines have value 1, those corresponding to edges in dashed lines have value $\frac{1}{2}$ and those corresponding to the dotted edges have value zero.

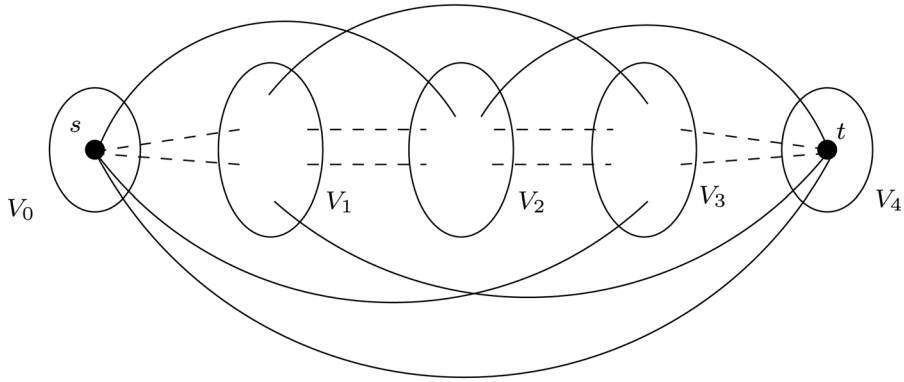

FIG. 2. Support graph of an L-st-path-cut with $L=3$ and T formed by the solid edges.

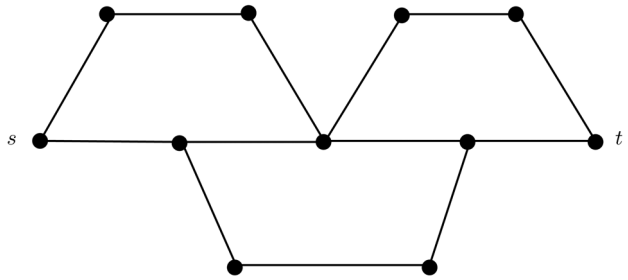

FIG. 3. Infeasible solution of the TNHNDP with $L=5$ and $D=\{(s,t)\}$.

It is easy to check that this solution is a fractional extreme point of the polyhedron given by the linear relaxation of formulation (2.7) with $L=3$ and $D=\{(1,4)\}$. Moreover, this solution violates the following inequality

$$x(e_1)+2x(e_2)+x(e_3)+2x(e_4)+x(e_5)+2x(e_8)\geq 3. \tag{2.8}$$

Furthermore, and as we will see in Section 4, inequality (2.8) defines a facet of the TNHNDP polytope.

Therefore, the natural inequalities together with the trivial inequalities are not sufficient to describe TNHNDP(G,3) and further inequalities are necessary for its description. In the next section, we present several classes of additional valid inequalities for TNHNDP(G,L), for $L\geq 2$ in general, and more specifically for $L=3$.

3. VALID INEQUALITIES OF TNHNDP(G,L)

In this section, we present several classes of valid inequalities for the TNHNDP polytope. We may notice that since any two node-disjoint paths in G are also edge-disjoint, it follows that any solution of the TNHNDP is also a solution of the TEHNDP. Thus, every valid inequality for the TEHNDP polytope is also valid for the TNHNDP polytope. In the following, we describe some classes of valid inequalities for the TEHNDP, which are, consequently, valid for the TNHNDP polytope, and introduce a class of inequalities which is specific to the TNHNDP.

3.1. *Valid Inequalities from the TEHNDP Polytope*

The first inequalities are the so-called generalized L-st-path-cut inequalities. They have been introduced by Dahl and Gouveia [5] for the problem of finding an L-st-path between two nodes s and t. They are defined as follows. Let $(s,t)\in D$

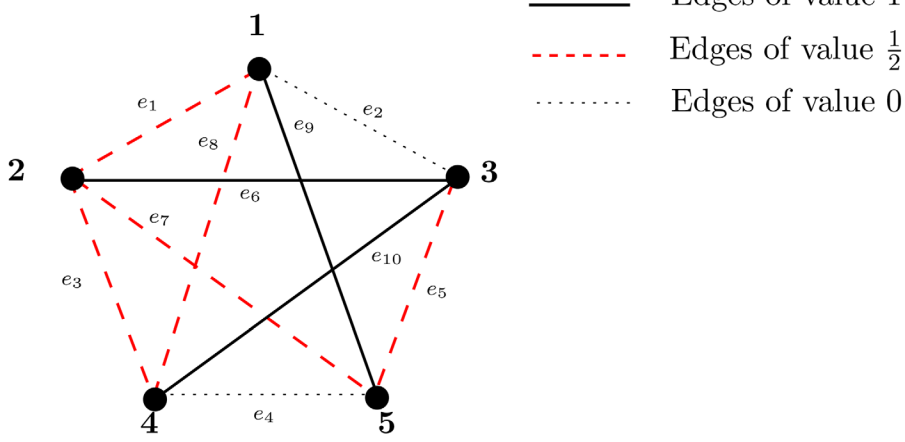

FIG. 4. A fractional extreme point of the linear relaxation of the TNHNDP with $L=3$ and $D=\{(1,4)\}$.[Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

and $\pi = (V_0, \ldots, V_{L+r})$, $r \geq 1$, be a partition of N such that $s \in V_0$ and $t \in V_{L+r}$. Then, the *generalized L-st-path-cut inequality* induced by (s, t) and π is

$$\sum_{e \in [V_i, V_j], i \neq j} \min(|i-j|-1, r) x(e) \geq r. \tag{3.9}$$

These inequalities can be easily extended to the TNHNDP by replacing the right-hand side of inequality (3.9) by $2r$, yielding

$$\sum_{e \in [V_i, V_j], i \neq j} \min(|i-j|-1, r) x(e) \geq 2r. \tag{3.10}$$

Inequalities (3.10) give, in some way, the minimum number of the so-called jumps involved in a partition $\pi = (V_0, \ldots, V_{L+r})$, with $s \in V_0$ and $t \in V_{L+r}$, for some $r \geq 1$. A *jump* is an edge between two nonconsecutive sets of π. A jump between two sets V_i and V_j of π will be said to be of length $|i-j|-1$. An inequality of type (3.10) for π expresses the fact that a set of *two st*-paths of length at most L in G contains a set of jumps of total length $\geq 2r$, and that these jumps may be completed by non-jump edges to form two *st*-paths of length $\leq L$.

Inequalities of type (3.10) will also be called generalized *L-st*-path-cut inequalities. They generalize the *L-st*-path-cut inequalities (2.2). In fact, these latter inequalities are obtained from the former ones by setting $r = 1$.

Theorem 2 ([5]). *For two nodes $s, t \in N$ with $(s, t) \in D$, the generalized L-st-path-cut inequalities (3.10) are valid for the TNHNDP polytope.*

Another family of valid inequalities is that of the so-called rooted partition inequalities, introduced by Huygens et al. [15] for the TEHNDP. Suppose that there are p demands, $|D| \geq p \geq 2$, of the form (s, t_i), $i = 1, \ldots, p$, for some $s \in N$ and $t_i \in N \setminus \{s\}$. These demands are said to be *rooted* in node s and s is the *root* for these demands. Let $(V_0, V_1, \ldots, V_p)$ be a partition of N such that $s \in V_0$ and $t_i \in V_i$, for all $i \in \{1, \ldots, p\}$. Such a partition is called a *rooted partition*. Huygens et al. [15] showed that, for any $L \geq 2$, the following inequality is valid for the TEHNDP polytope

$$x(\delta(V_0, V_1, \ldots, V_p)) \geq \left\lceil \frac{(L+1)p}{L} \right\rceil. \tag{3.11}$$

They called this the *rooted partition inequality*. It indicates that in the subgraph induced by any solution of the TEHNDP, the multicut induced by a rooted partition contains at least $\left\lceil \frac{(L+1)p}{L} \right\rceil$ edges.

As mentioned before, rooted partition inequalities (3.11) are also valid for the TNHNDP. Moreover, as we will see in Section 4, they define, under some conditions, facets of TNHNDP(G,3).

Theorem 3 ([15]). *The rooted partition inequalities (3.11) are valid for the TNHNDP polytope.*

We now present the last class of inequalities coming from the TEHNDP. In [15], Huygens et al. introduced the following class of inequalities for the TEHNDP. Let (s, t) and (s_1, t_1) be two demands, with $s_1 \neq s$ (t_1 and t may be the same node). Let $(V_0^1, V_0^2, V_1, V_2, V_3, V_4)$ be a partition of N such that $(V_0^1, V_0^2 \cup V_1, V_2, V_3, V_4)$ induces a 3-*st*-path-cut T and V_1 induces a s_1t_1-cut in G. Such a partition is called a *double cut*. For an edge set $F \subseteq [V_0^2 \cup V_1 \cup V_4, V_2]$ such that $|F|$ is odd, the inequality

$$\begin{aligned} & x([V_0^1, V_1 \cup V_2 \cup V_3 \cup V_4]) \\ & \quad + x([V_0^2, V_1 \cup V_3 \cup V_4]) + x([V_1, V_3 \cup V_4]) \\ & \quad + x([V_0^2 \cup V_1 \cup V_4, V_2] \setminus F) \geq 3 - \left\lfloor \frac{|F|}{2} \right\rfloor \end{aligned} \tag{3.12}$$

is called *double cut inequality* and is valid for the TEHNDP when $L = 3$. Figure 5 gives an illustration.

As before, double cut inequalities are also valid for the TNHNDP polytope.

Theorem 4 ([15]). *The double cut inequalities (3.12) are valid for the TNHNDP polytope when $L = 3$.*

Double cut inequalities (3.12) express the fact that if an edge set $F \subseteq [V_0^2 \cup V_1 \cup V_4, V_2]$ is taken in a solution of the TEHNDP, then the two edge-disjoint 3-*st*-paths and the two edge-disjoint 3- s_1t_1-paths of this solution must use at least $3 - \left\lfloor \frac{|F|}{2} \right\rfloor$ edges from $(T \cup [V_0^2, V_1] \cup [V_1, V_2]) \setminus F$.

Huygens et al. [15] also described necessary conditions for the double cut inequalities to define facets of the TEHNDP polytope.

Theorem 5 ([15]). *Consider the rooted TEHNDP with source node s, $L = 3$ and t_1 and t_2 two destination nodes. Let $\pi = (V_0^1, V_0^2, V_1, V_2, V_3, V_4)$ be a partition of N with $s \in V_0^1$, $t_1 \in V_1$ and $t_2 \in V_4$, and $F \subseteq [V_0^2 \cup V_1 \cup V_4, V_2]$. Then, the double cut inequality (3.12) induced by π and F defines a facet of the TEHNDP polytope only if*

1. $|V_0^1| = 1$ *and* $V_0^2 = \emptyset$,
2. $|V_1| = 1$,
3. $|V_4| \leq 2$.

3.2. Further Valid Inequalities

Next, we will introduce a new class of inequalities that are valid for the TNHNDP polytope. By contrast to those presented before, these inequalities are specific to the TNHNDP. We call them *st-jump inequalities* and they are presented in the following theorem.

Theorem 6. *Assume that $|N| \geq 5$, $(s, t) \in D$, $L = 3$ and let $(V_0, V_1, \ldots, V_4)$ be a partition of N such that $s \in V_0$ and $t \in V_4$. Let u_i be a node of V_i, $i = 1, 2, 3$. Then, the st-jump inequality*

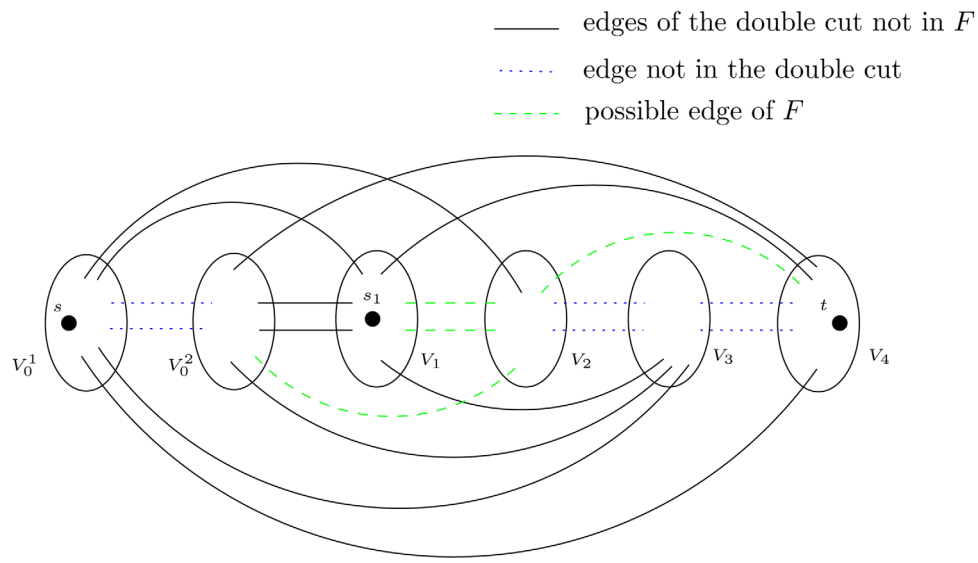

FIG. 5. A double cut with $L = 3$ and $t_1 = t$.[Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

$$\sum_{i=0}^{2} x([V_i, V_{i+2}]) + \sum_{i=0}^{1} \sum_{j\geq i+3}^{4} 2x([V_i, V_j]) + \sum_{i=0}^{1} x([V_i, V_{i+1} \setminus \{u_{i+1}\}]) + \sum_{i=2}^{3} x([V_i \setminus \{u_i\}, V_{i+1}]) \geq 3, \quad (3.13)$$

is valid for the TNHNDP polytope.

Proof. We shall use the Chvátal-Gomory procedure to show the validity of *st*-jump inequalities. Let $u_1 \in V_1$, $u_2 \in V_2$, and $u_3 \in V_3$ and let T_1, T_2, T_3, and T_4 be the *L-st-path-cuts* induced by $(V_0, \{u_1\}, V_2 \cup V_1 \setminus \{u_1\}, V_3, V_4)$, $(V_0, V_1, \{u_2\}, V_3 \cup V_2 \setminus \{u_2\}, V_4)$, $(V_0, V_1 \cup V_2 \setminus \{u_2\}, \{u_2\}, V_3, V_4)$, and $(V_0, V_1, V_2 \cup V_3 \setminus \{u_3\}, \{u_3\}, V_4)$, respectively. Then by summing the 3-*st*-path-cut inequalities induced by T_i, $i = 1, \ldots, 4$, and the *st*-node-cut inequalities induced by V_0 and u_1, by $V_0 \cup V_1$ and u_2, and by V_4 and u_3, together with

$$x(e) \geq 0, \text{for all } e \in \delta(V_1 \setminus \{u_1\}, V_3) \cup \delta(V_2 \setminus \{u_2\}, V_4) \cup \delta(V_1, V_3 \setminus \{u_3\}),$$

$$3x(e) \geq 0, \text{for all } e \in \delta(V_0, V_1 \setminus \{u_1\} \cup V_4) \cup \delta(V_1, V_2 \setminus \{u_2\}) \cup \delta(V_3 \setminus \{u_3\}, V_4),$$

$$4x(e) \geq 0, \text{for all } e \in \delta(V_0 \cup V_2 \setminus \{u_2\}, V_3) \cup \delta(V_1, V_4),$$

dividing the resulting inequality by 5, and rounding up the right-hand side, we obtain inequality (3.13). ∎

An illustration of *st*-jump inequalities is given by inequality (2.8) obtained from the graph of Figure 4 and the partition $\pi = (V_0, V_1, V_2, V_3, V_4) = (\{1\}, \{5\}, \{2\}, \{3\}, \{4\})$ with $u_1 = 5$, $u_2 = 2$, $u_3 = 3$. Here, $s = 1$, $t = 4$, and T will denote the 3-*st*-path-cut induced by π. Inequality (2.8) indicates that any feasible solution $F \subseteq E$ intersects T in at least two edges if F contains an edge of $[V_0, V_4] \cup [V_0, V_3] \cup [V_1, V_4]$ and at least three edges if not. Notice that any feasible solution may contain one or more edge from the *st*-path (s, u_1, u_2, u_3, t). For a more general partition (that is, with $|V_i| \geq 2$ for some $i \in \{1, 2, 3\}$), if F does not contain an edge of the path (s, u_1, u_2, u_3, t), then it may contain at least one edge from $[V_0, V_1 \setminus \{u_1\}] \cup [V_1, V_2 \setminus \{u_2\}] \cup [V_2 \setminus \{2\}, V_3] \cup [V_3 \setminus \{3\}, V_4]$.

As we will see in Section 4, *st*-jump inequalities may define facets of the TNHNDP polytope.

4. FACETS OF THE TNHNDP POLYTOPE

In this section, we investigate the conditions under which the inequalities presented in the previous section define facets of TNHNDP(G,3). These conditions will be used later to devise efficient separation procedures for these inequalities. First, we discuss the dimension of TNHNDP(G,3).

An edge $e \in E$ is said to be *essential* if the TNHNDP on the graph obtained by deleting the edge e from G has no solution. This clearly means that e is essential if and only if it belongs to either an *st*-cut or a 3-*st*-path-cut of cardinality 2, or, to an *st*-node-cut or a 3-*st*-path-node-cut of cardinality 1. We thus have the following theorem.

Theorem 7. *$dim(TNHNDP(G, 3)) = |E| - |E^*|$, where $|E^*|$ is the set of essential edges.*

Proof. Note first that since the edges of E^* belong to every solution of the problem, that is, $x^F(e) = 1$, for all $e \in E^*$, and every solution $F \subseteq E$ of the problem, we have $dim(\text{TNHNDP}(G, 3)) \leq |E| - |E^*|$. Now we show that $dim(\text{TNHNDP}(G, 3)) \geq |E| - |E^*|$ by showing that there

exists $|E| - |E^*| + 1$ solutions of the problem whose incidence vectors are affinely independent. Consider the edge sets $E_f = E \setminus \{f\}$, for every $f \in E \setminus E^*$. Clearly, the edge sets E_f, for every $f \in E \setminus \{f\}$, and E induce solutions of the TNHNDP with $L = 3$. Let $x_\emptyset$ and x_f be the incidence vectors of the solutions E and $E_f, f \in E \setminus E^*$, respectively. The matrix whose rows are $x_\emptyset - x_f$, $f \in E \setminus E^*$, is given by

$$
|E \setminus E^*| \left\{ \begin{array}{c} \\ \\ \\ \\ \end{array} \right.
\overbrace{\begin{array}{cccc} \\ \\ \\ \\ \end{array}}^{|E \setminus E^*|}
\overbrace{\begin{array}{ccc} \\ \\ \\ \\ \end{array}}^{|E^*|}
\begin{pmatrix}
1 & 0 & \cdots & 0 & 0 & \cdots & 0 \\
0 & \ddots & \ddots & \vdots & \vdots & \vdots & \vdots \\
\vdots & \ddots & \ddots & 0 & \vdots & \vdots & \vdots \\
0 & \cdots & 0 & 1 & 0 & \cdots & 0
\end{pmatrix}
$$

Clearly, this matrix contains an identity submatrix of size $|E| - |E^*|$, which shows that vectors $x_\emptyset - x_f$, $f \in E \setminus E^*$, are linearly independent. Thus, vectors $x_\emptyset$, $x_f, f \in E \setminus E^*$, form a family of $|E| - |E^*| + 1$ affinely independent solutions of the TNHNDP, which establishes the theorem. ∎

An immediate consequence of Theorem 7 is the following.

Corollary 8. *If $G = (N, E)$ is a complete graph and $|N| \geq 4$, then TNHNDP(G,3) is full dimensional.*

In the remainder of this article, the graph $G = (N, E)$ is assumed to be complete and simple (that is, it has no parallel edges), and has $|N| \geq 4$. By Corollary 8, TNHNDP(G,3) is full dimensional under these conditions. Notice that if G is not complete, one can obtain an equivalent TNHNDP by adding missing edges to G to make it complete, and giving the new edges a sufficiently large weight.

Now we investigate the conditions for the trivial and natural inequalities to define facets. The next two theorems give necessary and sufficient conditions for the trivial inequalities to be facet defining.

Theorem 9. *Inequality $x(e) \leq 1$ defines a facet of TNHNDP(G,3), for every $e \in E$.*

Proof. Note that as G is complete and $|N| \geq 4$, by Corollary 8, TNHNDP(G,3) is full dimensional. Thus, to prove the theorem, it suffices to show that there exist $|E|$ solutions of the problem satisfying $x(e) = 1$ and which are affinely independent.

For all $f \in E \setminus \{e\}$, consider the edge set $E_f = E \setminus \{f\}$. As $|N| \geq 4$ and G is complete, E_f induces a solution of the TNHNDP, for all $f \in E \setminus \{e\}$. Hence, E and the edge sets E_f, for all $f \in E \setminus \{e\}$, constitute a set of $|E|$ solutions of the TNHNDP. Moreover, their incidence vectors satisfy $x(e) = 1$. Finally, using a similar proof as in Theorem 7, we can show that these vectors are affinely independent which proves the theorem. ∎

Theorem 10. *Inequality $x(e) \geq 0$ defines a facet of TNHNDP(G,3) if and only if either $|N| \geq 5$ or $|N| = 4$, $D = \{(s,t)\}$ and $e = uv$ with $\{u, v\} = N \setminus \{s, t\}$.*

Proof. Note that G being a complete graph with $|N| \geq 4$ implies that TNHNDP(G,3) is full dimensional.

Suppose that $|N| \geq 5$. Then, for all $(s,t) \in D$, G contains four node-disjoint st-paths (an edge of $[s,t]$ and three paths of the form $(s,u,t), u \in N \setminus \{s,t\}$). Hence, any edge set $E \setminus \{f,g\}, f,g \in E$, contains at least two node-disjoint 3-st-paths. The sets $E \setminus \{e\}$ and $E_f = E \setminus \{e,f\}$, for all $f \in E \setminus \{e\}$, constitute a set of $|E|$ solutions of the TNHNDP. Moreover, their incidence vectors satisfy $x(e) = 0$ and are affinely independent, yielding that $x(e) \geq 0$ defines a facet of TNHNDP(G,3).

Now suppose that $|N| = 4$, so $N = \{s,t,u,v\}$. If $|D| \geq 2$, then G has at least three terminal nodes. Without loss of generality, we assume that these three terminals are s, t, and u. In this case, any edge $e \in E$ is adjacent to at least one terminal $z \in \{s,t,u\}$. Therefore, the inequality $x(e) \geq 0$ can be obtained by summing the following inequalities which are valid for TNHNDP(G,3)

$$
\begin{aligned}
x(\delta(z)) &\geq 2, \\
-x(f) &\geq -1, \text{ for all } f \in \delta(z) \setminus \{e\}.
\end{aligned}
$$

Thus, $x(e) \geq 0$, for every $e \in E$, is redundant with respect to the st-cut inequalities (2.1) and the trivial inequalities (2.5).

Finally, suppose that $|N| = 4$, $|D| = 1$, so $D = \{(s,t)\}$, and $e = uv$ with $\{u, v\} = N \setminus \{s,t\}$. Then, the edge sets $E \setminus \{e\}$ and E_f, for all $f \in E \setminus \{e\}$, introduced above, are still solutions of TNHNDP. Moreover, their incidence vectors satisfy $x(e) = 0$ and are affinely independent, implying that $x(e) \geq 0$ defines a facet. ∎

In what follows, we investigate the conditions for the st-cut and the st-node-cut inequalities to define facets of TNHNDP(G,3). Observe that it may not be easy to give necessary and sufficient conditions for these inequalities to define facets in the general case when $|D| \geq 2$. Consider for example the instance of the TNHNDP given by the graph G shown in Figure 6, and demand set $D = \{(s,t_1), (s,t_2), (s,t_3)\}$.

We can see that the node set $W = \{s, t_1\}$ induces a st_2-node-cut with respect to node z, and that every solution $F \subseteq E$ of the TNHNDP on G such that $x^F(\delta_{G-z}(W)) = 1$ satisfies $x^F(st_1) = 1$. Indeed, if we consider a solution $F \subseteq E$ of the TNHNDP on G such that $st_1 \notin F$, then as $(s,t_1) \in D$ and, hence, the subgraph induced by F contains two node-disjoint paths between s and t_1, those paths are of the form $P_1 = (s, u_1, \ldots, t_1)$ and $P_2 = (s, u_2, \ldots, t_1)$, where $u_1, u_2 \in \{z, t_2, t_3\}$. As $st_1 \notin F$, we have that u_1 and u_2 are in $\overline{W}$. Moreover, as P_1 and P_2 are node-disjoint, it follows that $u_1 \neq u_2$, and hence, one of them, say u_1, is different from z. Therefore, the path P_1 uses at least two edges of $\delta_{G-z}(W)$, and $x^F(\delta_{G-z}(W)) \geq 2 > 1$. Consequently, the st_1-node-cut inequality induced by W and z does not define a facet of TNHNDP(G,3).

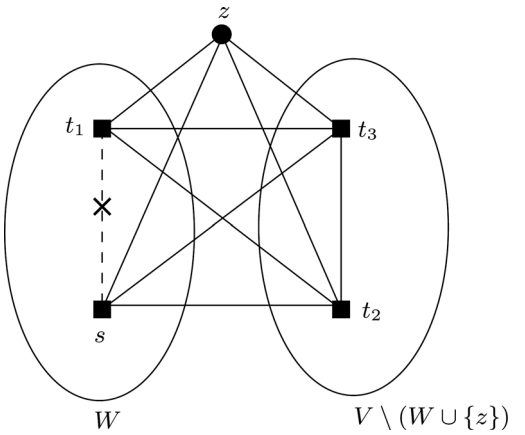

FIG. 6. A graph G, a node set W and node z such that $x(\delta_{G-z}(W)) \geq 1$ does not induce a facet.

If we consider now the graph G' obtained by adding to G a steiner node u adjacent to all the nodes of G (in this way, G' is also complete as G is complete) and let $W = \{s, t_1, u\}$, then $x(\delta_{G'-z}(W)) \geq 1$ defines a facet of TNHNDP (G',3). In fact, a deeper observation of the above example shows that the fact that an *st*-node-cut inequality defines a facet depends on the disposition of the demands and steiner nodes over W and $\overline{W}$. There may exist several types of configurations in which *st*-node-cut inequalities define facets. In the next two theorems, we show that the *st*-cut and the *st*-node-cut inequalities define facets of TNHNDP(G,3) when $|D| = 1$.

Theorem 11. *When $|D| = 1$, every st-cut inequality defines a facet of TNHNDP(G,3).*

Proof. Similar to that used to prove Theorem 3.3 of [3]. ∎

Theorem 12. *If $|N| \geq 5$ and $|D| = 1$ with $D = \{(s,t)\}$, then every st-node-cut inequality induced by a node set $W \subseteq N$ such that $s \in W$, $t \notin W$ and $W \setminus \{s\} \neq \emptyset \neq \overline{W} \setminus \{t\}$ defines a facet of TNHNDP(G,3).*

Proof. Let $W \subseteq V$ be such that $s \in W$, $t \in \overline{W}$, and $W \setminus \{s\} \neq \emptyset \neq \overline{W} \setminus \{t\}$, and let $z \in N \setminus \{s,t\}$. As G is complete and $|N| \geq 5$, TNHNDP(G,3) is full dimensional. Let us denote by $ax \geq \alpha$ the inequality (2.3) induced by W and z, and let $bx \geq \beta$ be a facet defining inequality of TNHNDP(G,3) such that $\{x \in \text{TNHNDP}(G,3) : ax = \alpha\} \subseteq \{x \in \text{TNHNDP}(G,3) : bx = \beta\}$. To prove that inequality $x(\delta_{G-z}(W)) \geq 1$, induced by W and z, defines a facet, it then suffices to show that there exist $\rho \in \mathbb{R}$ such that $b = \rho a$. To do this, we use a result proven by Bendali et al. [3] on the so-called k edge-disjoint 3-hop-constrained paths problem (kHPP for short). This latter problem consists of finding, given two nodes s and t, a minimum weight subgraph of G containing k edge-disjoint paths of length at most 3 between s and t. Here k is a fixed integer. Bendali et al. [3] studied the polytope associated with this problem, denoted by kHPP(G), and showed that for any $k \geq 1$, the *st*-cut inequalities (2.1) are valid for kHPP(G) and define facets. They also showed that kHPP(G) is full dimensional if G is complete and $N \geq 4$.

Our idea is to use the fact that $x(\delta_{G-z}(W)) \geq 1$ is a valid *st*-cut inequality for 1HPP($G - z$), and hence, by Bendali et al. [3], defines a facet of 1HPP ($G - z$). As $x(\delta_{G-z}(W)) \geq 1$ defines a facet of 1HPP($G - z$), there exist $dim($ 1 HPP($G - z$)) + 1 solutions of the 1HPP on $G - z$ whose incidence vectors satisfy $x(\delta_{G-z}(W)) \geq 1$ with equality and are affinely independent. In what follows, we shall use these solutions to build $|E|$ solutions of the TNHNDP on G satisfying $x(\delta_{G-z}(W)) \geq 1$ with equality and which are still affinely independent. This will yield the existence of $\rho \in \mathbb{R}$ such that $b = \rho a$. Notice that as G is complete and $|N| \geq 5$, $G - z$ is also complete with $|N \setminus \{z\}| \geq 4$. Thus, by Bendali et al. [3], 1HPP($G - z$) is full dimensional and $dim(\text{1HPP}(G-z)) = |E| - |\delta(z)| = |E| - |N| + 1$. The idea of the proof is summarized in Figure 7 below.

As $x(\delta_{G-z}(W)) \geq 1$ defines a facet of 1HPP(G), there must exist $m' = |E| - |N| + 1$ solutions of the 1HPP on $G - z$, denoted by F'_i, $i = 1, \ldots, m'$, whose incidence vectors are affinely independent and satisfy $x(\delta_{G-z}(W)) = 1$. As the edge sets F'_i, where $i \in \{1, \ldots, m'\}$, induce affinely independent vectors, one of them contains the edge st. We will denote by F'_1 such an edge set.

We denote the edges of $\delta(z) \setminus \{sz, zt\}$ by $e_j, j = 1, \ldots, |N| - 3$, and the edges sz and zt by $e_{|N|-2}$ and $e_{|N|-1}$, respectively. Also, observe that as $W \setminus \{s\} \neq \emptyset \neq \overline{W} \setminus \{t\}$, there exist two nodes, say u_1 and u_2, with $u_1 \in W \setminus \{s\}$ and $u_2 \in \overline{W} \setminus \{t\}$.

The edge sets $F_i = F'_i \cup \delta(z)$, for all $i \in \{1, \ldots, m'\}$, induce solutions of the TNHNDP. Indeed, as F'_i, $i \in \{1, \ldots, m'\}$ is solution of the 1HPP on $G - z$, there exists an *st*-path of length at most 3, say (s, u, v, t) (here u and v may be the same node), in the subgraph of $G - z$ induced by F'_i. Moreover, as G is complete, the edges sz and zt are in G, and (s, z, t) forms an *st*-path of length 2 in G. Finally, as z is not a node of $G - z$, u, and v are necessarily different from z, and hence, the paths (s, u, v, t) and (s, z, t) are node-disjoint. Thus, F_i induces a solution of the TNHNDP on G. Furthermore, the incidence vectors of F_i, $i \in \{1, \ldots, m'\}$, satisfy $x(\delta_{G-z}(W)) = 1$.

Let a' and b' be the restriction on $E \setminus \delta(z)$ of a and b, respectively. Thus, we have $a' x^{F_i} = \alpha$, for $i = 1, \ldots, m'$. Therefore, $b' x^{F_i} = \beta$, for $i = 1, \ldots, m'$. As x^{F_i}, $i = 1, \ldots, m'$, are affinely independent and $\alpha \neq 0$, it follows that $x^{F_i} \neq 0$, $i = 1, \ldots, m'$, and hence, x^{F_i}, $i = 1, \ldots, m'$, are linearly independent. Consequently, a is the unique solution of the system $a x^{F_i} = \alpha$, for $i = 1, \ldots, m'$. Let ρ be such that $\beta = \rho\alpha$. It then follows that $b' = \rho a'$.

Now it remains to show that $b(e) = 0$ for all $e \in \delta(z)$. Consider the edge sets $F_{m'+j} = F_{m'} \setminus \{e_j\}$, for $j = 1, \ldots, |N| - 3$. Clearly, these sets induce solutions of the TNHNDP and their incidence vectors satisfy $x(\delta_{G-z}(W)) = 1$. As $a x^{F_{m'}} = a x^{F_{m'+j}} = \alpha$, it follows that $b x^{F_{m'}} = b x^{F_{m'+j}} = \beta$. This implies that $b(e_j) = 0$, for $j = 1, \ldots, |N| - 3$.

Finally, let $F_0 = F_1 \cup \{su_1, su_2\}$. Clearly, the edge sets F_0, $F_{|E|-1} = F_0 \setminus \{sz\}$ and $F_{|E|} = F_0 \setminus \{zt\}$ induce solutions of the TNHNDP and their incidence vectors satisfy

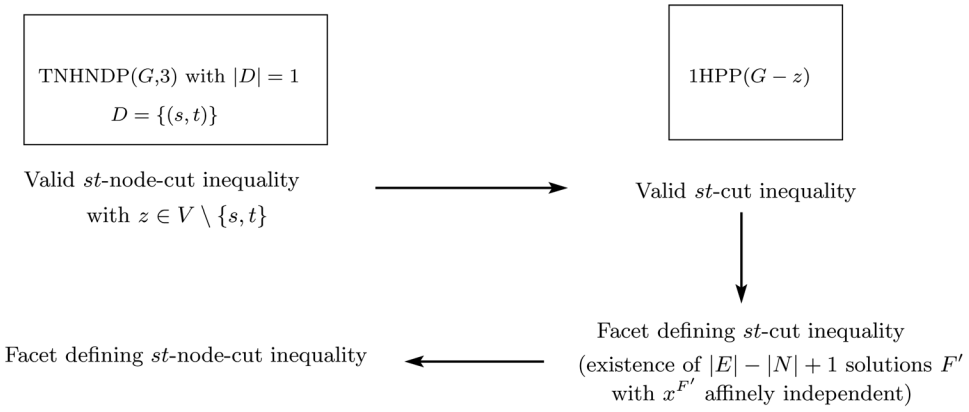

FIG. 7. Scheme of the proof of Theorem 12.

$x(\delta_{G-z}(W)) = 1$. Thus, $bx^{F_{|E|-1}} = bx^{F_{|E|}} = bx^{F_0} = \beta$, and hence, $b(sz) = b(zt) = 0$. Therefore, $b = \rho a$.

As TNHNDP(G,3) is full dimensional and $b = \rho a$, we obtain that $x(\delta_{G-z}(W)) \geq 1$ defines a facet of TNHNDP(G,3). ∎

The observation previously made for the st-cut and st-node-cut inequalities is also valid for 3-st-path-cut and 3-st-node-path-cut inequalities. The conditions under which they define facets depend on the disposition of the demands and steiner nodes over the partition inducing the 3-st-path-cut and 3-st-node-path-cut. There may exist several types of configurations for these inequalities to define facets. In the following two theorems, we give necessary and sufficient conditions for 3-st-path-cut and 3-st-node-path-cut inequalities to define facets of TNHNDP(G,3) when $|D| = 1$.

Theorem 13. *If $|D| = 1$, then every L-st-path-cut inequality defines a facet of TNHNDP(G,3) if and only if $|V_0| = 1 = |V_{L+1}|$.*

Proof. Similar to that used to prove Theorem 3.4 of [3]. ∎

Theorem 14. *If $|D| = 1$, then every L-st-node-path-cut inequality defines a facet of TNHNDP(G,3) if and only if $|V_0| = 1 = |V_{L+1}|$.*

Proof. The idea of the proof is the same as that used in proving Theorem 12. To summarize, we use the fact that a 3-st-node-path-cut inequality, $x(T_{G-z}) \geq 1$, for some 3-st-path-cut T and some node $z \in N \setminus \{s,t\}$, is valid for 1HPP($G - z$) (recall that 1HPP($G - z$) is the polytope associated with the 3-hop-constrained st-path problem on the graph $G - z$). As shown by Bendali et al. [3], a 3-st-path-cut inequality, induced by a partition $(V_0, \ldots, V_4)$, defines a facet of 1HPP($G - z$) if and only if $V_0 = \{s\}$ and $V_4 = \{t\}$. As in Theorem 12, we use this observation to show that if the face $\{x \in \text{TNHNDP}(G, 3) : ax = \alpha\}$ is contained in a facet $\{x \in \text{TNHNDP}(G, 3) : bx = \beta\}$, where $ax \geq \alpha$ denotes the inequality $x(T_{G-z}) \geq 1$, then $b = \rho a$, for some $\rho \in \mathbb{R}$. As G is complete and $|N| \geq 4$, TNHNDP(G,3) is full dimensional and we obtain that $x(T_{G-z}) \geq 1$ defines a facet of TNHNDP(G,3). ∎

The next theorem, given without proof, describes sufficient conditions for the st-jump inequalities (3.13) to define facets of TNHNDP(G,3). For the proof, the reader can refer to [8].

Theorem 15. *Let $G = (N, E)$ be a complete graph with $|N| \geq 5$. The st-jump inequality (3.13), induced by a demand $(s,t) \in D$ and a partition $(V_0, V_1, \ldots, V_4)$, defines a facet of TNHNDP(G,3) if one of the following conditions holds*

1. $|V_1| = |V_3| = 1$,
2. $|V_1| = |V_2| = |V_4| = 1$,
3. $|V_0| = |V_2| = |V_3| = 1$.

Notice that by Theorem 15, inequality (2.8) introduced in the example of Section 2 defines a facet of the TNHNDP polytope associated with the graph of Figure 4.

The last theorems of this section deal with the conditions under which the rooted partition inequalities (3.11) define facets of the TNHNDP polytope. In the following, we investigate both necessary and sufficient conditions.

Theorem 16 ([15]). *Let $t_1, \ldots, t_p$, $p \geq 2$, be a set of p destinations relative to a source node s, that is, $(s, t_i) \in D$, for all $i \in \{1, \ldots, p\}$, and let $\pi = (V_0, V_1, \ldots, V_p)$ be a partition of N such that $s \in V_0$ and $t_i \in V_i$, $i = 1, \ldots, p$. Then, the rooted partition inequality (3.11) induced by π defines a facet of TEHNDP(G,L), for any $L \geq 2$, only if p is not multiple of L.*

In fact, we can show that a rooted partition inequality is redundant with respect to the st-cut inequalities when $p \geq L$.

Theorem 17. *For any $L \geq 2$, a rooted partition inequality induced by a partition $(V_0, V_1, \ldots, V_p)$, $p \geq 2$, is redundant with respect to the st-cut inequalities when $p \leq L$.*

Proof. Note that $p \leq L$ implies that $\lceil \frac{p}{L} \rceil = 1$. Also, as each node set V_i, $i = 0, \ldots, p$, contains a terminal node, the

inequalities

$$x(\delta_G(V_i)) \geq 2, \text{ for } i = 0, \ldots, p, \tag{4.14}$$

are valid for TNHNDP(*G,L*). By summing these inequalities and dividing by 2 the resulting inequality, we get

$$x(\delta_G(\pi)) \geq p + 1 = p + \left\lceil \frac{p}{L} \right\rceil,$$

which corresponds to the rooted partition inequality induced by $(V_0, V_1, \ldots, V_p)$. ∎

Corollary 18. *A rooted partition inequality induced by a partition $(V_0, V_1, \ldots, V_p)$, $p \geq 2$, defines a facet of TNHNDP(G,3), different from the st-cut inequalities, only if $p \geq 4$ and p is not multiple of 3.*

The next theorem describes further necessary conditions for (3.11) to be a facet.

Theorem 19. *Let $t_1, \ldots, t_p$, $p \geq 4$, be a set of p destinations of a given source node s, and let $\pi = (V_0, V_1, \ldots, V_p)$ be a partition of N such that $s \in V_0$ and $t_i \in V_i$, $i = 1, \ldots, p$. Then, the rooted partition inequality (3.11) induced by π defines a facet of TNHNDP(G,3), only if $|V_i| \neq 2$ or $|V_i \cap R| \neq 2$, for all $i \in \{1, \ldots, p\}$ (recall that R denotes the set of terminals of the graph).*

Proof. Suppose that the conditions of the theorem are not satisfied by a node set V_{i_0}, with $i_0 \in \{1, \ldots, p\}$, that is, $|V_{i_0} \cap R| = |V_{i_0}| = 2$. We will show that in this case the inequality (3.11) induced by π is dominated by another rooted partition inequality and the trivial inequalities.

As we have assumed that $|V_{i_0} \cap R| = |V_{i_0}| = 2$, the node set V_{i_0} is composed of two terminal nodes, say $t^1_{i_0}$ and $t^2_{i_0}$, so $V_{i_0} = \left\{t^1_{i_0}, t^2_{i_0}\right\}$. Also, as the graph G is assumed to be complete and simple, there exists a single edge between $t^1_{i_0}$ and $t^2_{i_0}$. Now consider the partition $\pi' = (V'_0, V'_1, \ldots, V'_p, V'_{p+1})$ such that

$$\begin{aligned}
&V'_i = V_i, \text{ for all } i \in \{1, \ldots, i_0 - 1\},\\
&V'_{i_0} = \left\{t^1_{i_0}\right\},\\
&V'_{i_0+1} = \left\{t^2_{i_0}\right\},\\
&V'_i = V_{i-1}, \text{ for all } i \in \{i_0 + 2, \ldots, p + 1\}.
\end{aligned}$$

Clearly, π' induces a rooted partition and, obviously, by summing the inequalities

$$\begin{aligned}
&x(\delta(\pi')) \geq p + 1 + \left\lceil \frac{p+1}{L} \right\rceil \text{ (rooted partition inequality}\\
&\qquad\qquad \text{induced by } \pi'),\\
&-x(t^1_{i_0} t^2_{i_0}) \geq -1,
\end{aligned}$$

we obtain

$$x(\delta(\pi)) \geq p + \left\lceil \frac{p+1}{L} \right\rceil \geq p + \left\lceil \frac{p}{L} \right\rceil.$$

This latter inequality dominates the inequality (3.11) induced by π, which, hence, cannot define a facet of TNHNDP(*G,L*). ∎

The next theorem gives sufficient conditions for inequalities (3.11) to define facets.

Theorem 20. *Let $t_1, \ldots, t_p$, $p \geq 4$, be a set of p destinations of a given source node s, and let $\pi = (V_0, V_1, \ldots, V_p)$ be a partition of N such that $s \in V_0$ and $t_i \in V_i$, $i = 1, \ldots, p$. Then, the rooted partition inequality (3.11), induced by π, defines a facet of TNHNDP(G,3) if $|V_i| = 1$ for all $i \in \{1, \ldots, p\}$ and p is not a multiple of 3.*

Proof. Suppose that $|V_i| = 1$ for all $i = 1, \ldots, p$ and p is not a multiple of 3. Without loss of generality, we will assume that $E(V_0) \neq \emptyset$. Let us denote inequality (3.11) by $a^T x \geq \alpha$, and let $b^T x \geq \beta$ be a facet defining inequality of TNHNDP(*G*,3) such that

$$\begin{aligned}
&\left\{x \in \text{TNHNDP}(G, 3) : a^T x = \alpha\right\}\\
&\quad \subseteq \left\{x \in \text{TNHNDP}(G, 3) : b^T x = \beta\right\}.
\end{aligned}$$

We will show that $b = \rho a$ for some $\rho \neq 0$. ∎

Let $p = 3q + r$, where $q \geq 1$ is an integer and r is either 1 or 2. We distinguish two cases.

Case 1. $r = 1$.

Consider the edge sets

$$\begin{aligned}
F_0 = &\bigcup_{i=0}^{q-1} \{st_{3i+1}, t_{3i+1}t_{3i+2}, t_{3i+2}t_{3i+3}, t_{3i+3}s\}\\
&\cup \left\{st_{3q+1}, t_{3q+r}t_1\right\} \cup E(V_0),
\end{aligned}$$

and

$$F_e = (F_0 \setminus \left\{t_p t_1\right\}) \cup \{e\}, \text{ for all } e \in \delta(t_p) \setminus \left\{st_p, t_p t_1\right\}$$

(see Fig. 8 for an illustration of F_0).

Observe that the set F_0 is a collection of q cycles of length 4, each cycle containing s and three destination nodes, together with the edges $t_1 t_p$ and $t_p s$. Moreover, these cycles cover all the destination nodes. Hence, F_0 induces a solution of the TNHNDP. Furthermore, $|F_0 \cap \delta(V_0, V_1, \ldots, V_p)| = 4q + 2 (= \alpha)$, implying that x^{F_0} satisfies (3.11) with equality. Similarly, the sets F_e also induce solutions of the TNHNDP. As for F_0, each destination node belongs to a cycle of length ≤ 4 that contains the source node s. Moreover, as $|F_e \cap \delta(V_0, V_1, \ldots, V_p)| = |F_0 \cap \delta(V_0, V_1, \ldots, V_p)| = 4q + 2$, x^{F_e} also satisfies (3.11) with equality, for all $e \in \delta(t_p) \setminus \left\{st_p, t_p t_1\right\}$. Consequently, we have $b^T x^{F_0} - b^T x^{F_e} = b(e) - b(t_1 t_p) = 0$. Thus,

$$b(e) = b(t_1 t_p) \quad \text{for all } e \in \delta(t_p) \setminus \left\{st_p, t_p t_1\right\}. \tag{4.15}$$

Furthermore, let $F_1 = (F_0 \setminus \left\{st_p\right\}) \cup \left\{t_p t_{p-1}\right\}$. Note that in the graph induced by F_1, t_p belongs to a cycle of length ≤ 4,

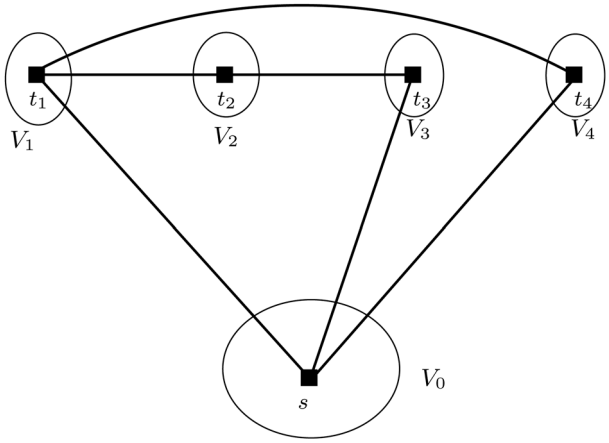

FIG. 8. Edge set F_0 in the case where $r = 1$ and $q = 1$.

containing s and two further destination nodes. Therefore, F_1 induces a solution of the TNHNDP. Its incidence vector satisfies $a^T x \geq \alpha$ with equality. As $b^T x^{F_1} = \beta$, we obtain $b(st_p) = b(t_p t_{p-1})$. This together with (4.15) yields

$$b(e) = b(e') \quad \text{for all } e, e' \in \delta(t_p).$$

By exchanging the role of t_p and t_i's, $i \neq p$, we get

$$b(e) = \rho \quad \text{for all } e \in \delta(V_0, \ldots, V_p), \text{ for some } \rho \in \mathbb{R}.$$

Now let $e \in E(V_0)$. Obviously, $F_0 \setminus \{e\}$ induces a solution of the TNHNDP whose incidence vector satisfies $a^T x \geq \alpha$ with equality. Thus, $b(e) = b^T x^{F_0} - b^T x^{F_0 \setminus \{e\}} = 0$, for every $e \in E(V_0)$.

Case 2: $r = 2$.

In this case, we let

$$F_0' = \bigcup_{i=0}^{q-1} \{st_{3i+1}, t_{3i+1}t_{3i+2}, t_{3i+2}t_{3i+3}, t_{3i+3}s\} \cup \left\{st_{3q+1}, t_{3q+r}t_1, t_{3q+1}t_{3q+2}\right\} \cup E(V_0) \qquad (4.16)$$

and

$$F_e' = (F_0' \setminus \left\{t_p t_1\right\}) \cup \{e\}, \text{ for some } e \in \delta(t_p)$$

(see Fig. 9 for an illustration of F_0').

Similarly, we can see that, F_0' and F_e' induce solutions of the TNHNDP whose incidence vectors satisfy $a^T x \geq \alpha$ with equality. Hence, we have $b^T x^{F_0'} - b^T x^{F_e'} = b(e) - b(t_1 t_p) = 0$. In consequence,

$$b(e) = b(e') \quad \text{for all } e, e' \in \delta(t_p).$$

By exchanging the role of t_p and t_i's, $i \neq p$, we get

$$b(e) = \rho \quad \text{for all } e \in \delta(V_0, \ldots, V_p), \text{ for some } \rho \in \mathbb{R}.$$

Now for an edge $e \in E(V_0)$, the edge set $F_0 \setminus \{e\}$ induces a solution of the TNHNDP whose incidence vector satisfies

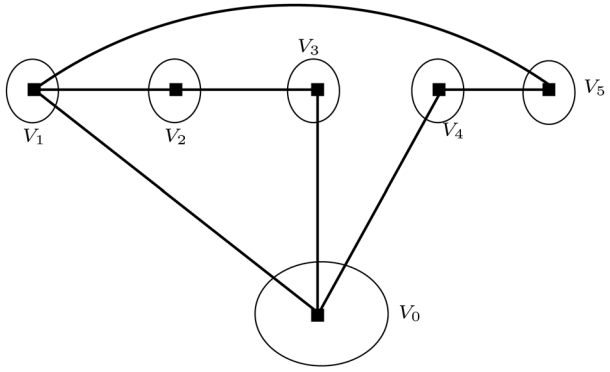

FIG. 9. Edge set F_0 in the case where $r = 2$ and $q = 1$.

$a^T x \geq \alpha$ with equality. Thus, $b(e) = b^T x^{F_0'} - b^T x^{F_0' \setminus \{e\}} = 0$, for every $e \in E(V_0)$.

Consequently, from Cases 1 and 2, we have

$$b(e) = \begin{cases} \rho & \text{for all } \delta(V_0, \ldots, V_p), \\ 0 & \text{for all } E(V_0). \end{cases}$$

Thus $b = \rho a$, and the proof is complete.

5. BRANCH-AND-CUT ALGORITHM

In this section, we devise a Branch-and-Cut algorithm for solving the TNHNDP when $L = 3$. We first describe the general framework of the algorithm. Then, we discuss two main ingredients of the algorithm, namely separation routines for the different inequalities involved in the algorithm and a primal heuristic.

5.1. *General Framework*

Our Branch-and-Cut algorithm for the TNHNDP starts by solving the linear relaxation of the natural formulation of the problem, that is,

$$\min \left\{wx : x \text{ satisfies (2.1)– (2.6) and } x \in \mathbb{R}^E\right\}. \qquad (5.17)$$

To solve this linear relaxation, we use the so-called *cutting plane method* as the natural inequalities (2.1)–(2.4) are exponential in number. Recall that the cutting plane method consists of solving a linear program (LP) containing a large number of constraints by solving a series of LPs containing a subset of constraints of the original LP. In our case, the algorithm starts with an LP containing a small number of natural constraints (2.1)–(2.4) and, iteratively, adds constraints (2.1)–(2.4) that are violated by the optimal solution, say x^*, of the current LP. The algorithm stops when all the natural inequalities are satisfied by x^*. In this latter case, x^* is the optimal solution of problem (5.17). To find inequalities of type (2.1)–(2.4) which are violated by the solution of the current LP, if there are any, one has to solve the so-called separation problem associated with inequalities (2.1)–(2.4). Recall that *the separation problem* associated with a family of

inequalities $\mathcal{F}$ and a solution $\bar{x}$ is to verify if $\bar{x}$ satisfies all the inequalities of $\mathcal{F}$, and if not, to exhibit at least one of them which is violated by $\bar{x}$. An algorithm solving a separation problem is called a *separation algorithm*.

If x^* satisfies all the natural inequalities and is integral, then it is optimal for the problem (2.7). If x^* is fractional, then we strengthen the linear relaxation of the problem by adding further valid inequalities. To do this, we also add in the cutting plane phase the rooted partition and double cut inequalities (3.11) and (3.12), respectively. Notice that the *st*-jump inequalities (3.13) are not taken into account in the algorithm because we did not find any of them violated in the preliminary tests we did for $|D| \geq 2$.

The separation of the different inequalities used in the Branch-and-Cut algorithm are performed in the following order

1. *st*-cut and *L-st*-path-cut inequalities,
2. *st*-node-cut and *L-st*-node-path-cut inequalities,
3. rooted partition inequalities (only for rooted instances),
4. double cut inequalities.

In this order, we separate the natural inequalities first as they are necessary in the natural formulation of the TNHNDP. Then, the rooted partition inequalities are separated before double cut inequalities because, in our preliminary tests, rooted partitions inequalities appeared more than double cut inequalities.

We apply the rooted partition inequalities (3.11) for the rooted TNHNDP (i.e., when the set of demands is rooted in a single node) and do not apply them when arbitrary demands are considered.

All the inequalities added during the algorithm are global (i.e., valid at every node of the Branch-and-Cut tree) and many of them can be added at each iteration. Furthermore, we move to the next class of inequalities only if we do not find any violated inequalities in the current class. All the generated inequalities are stored in a dynamic pool (managed by ABACUS), so the inequalities in the pool can be removed from the current LP when they are not active. Also, the inequalities in the pool are separated before we separate the classes of inequalities according to the order given above.

In the following, we present the separation algorithms we have devised for the natural inequalities (2.1)–(2.4), the rooted partition inequalities (3.11), and the double cut inequalities (3.12).

5.2. *Separation Algorithms*

First, we consider natural inequalities, that is to say the *st*-cut and the 3-*st*-path-cut inequalities, on one hand, and the *st*-node-cut and the 3-*st*-node-path-cut inequalities on the other hand.

We give the following theorem showing that the *st*-cut and the 3-*st*-path cut inequalities (2.1) and (2.2) can be separated in polynomial time.

Theorem 21. *The separation problem of the st-cut and the 3-st-path-cut inequalities (2.1) and (2.2) can be solved using an $O(|D|(|E|^2|N| + |N||E|\log|E|))$-time algorithm.*

Proof. Let $\bar{x} \in \mathbb{R}^E$ be the solution for which we are separating the natural inequalities (2.1)–(2.4). To separate them, we consider the following graph transformation from [3] (see also [6]). Let $(s,t) \in D$ and let $V_{st} = N \setminus \{s,t\}$, V'_{st} be a copy of V_{st} and $\widetilde{N}_{st} = V_{st} \cup V'_{st} \cup \{s,t\}$. The copy in V'_{st} of a node $u \in V_{st}$ will be denoted by u'. From G and (s,t), we build the directed graph $\widetilde{G}_{st} = (\widetilde{N}_{st}, \widetilde{A}_{st})$. Its arc set $\widetilde{A}_{st}$ is obtained as follows. For an edge of the form $st \in E$, we add an arc (s,t) in $\widetilde{A}_{st}$. For each edge $su \in E$, $u \neq t$, (resp., $vt \in E$, $v \neq s$), we add in $\widetilde{A}_{st}$ an arc (s,u), $u \in V_{st}$ (resp. (v',t), $v' \in V'_{st}$). For each edge $uv \in E$, with $u,v \notin \{s,t\}$, we add two arcs (u,v') and (v,u') in $\widetilde{A}_{st}$, with $u,v \in V_{st}$ and $u',v' \in V'_{st}$. Finally, for each node $u \in N \setminus \{s,t\}$, we add an arc (u,u') in $\widetilde{A}_{st}$ (see Fig. 10 for an illustration).

Notice that for each $(s,t) \in D$, $|\widetilde{N}_{st}| = 2|N| - 2$ and $|\widetilde{A}_{st}| = 2|E| - |\delta(s)| - |\delta(t)| + |[s,t]|$.

Bendali et al. [3] (see also [6]) showed that there is a one-to-one correspondence between the *st*-cuts and the 3-*st*-path-cuts in G and the *st*-dicuts in $\widetilde{G}_{st}$ which do not contain arcs of the form (u,u'), for all $u \in N \setminus \{s,t\}$. Moreover, if each arc $a \in \widetilde{A}_{st}$, corresponding to an edge $e \in E$, is assigned the capacity $\widetilde{c}(a) = \bar{x}(e)$ and each arc of the form (u,u') is assigned an infinite capacity, then the weight of an *st*-cut or 3-*st*-path-cut in G with respect to $\bar{x}$ is the same as that of the corresponding *st*-dicut in $\widetilde{G}_{st}$ with respect to capacity vector $\widetilde{c}$. Thus, for a given $(s,t) \in D$, there is an *st*-cut or 3-*st*-path-cut inequality violated by $\bar{x}$ if and only if there is an *st*-dicut in $\widetilde{G}_{st}$ whose capacity is < 2. If $\widetilde{W}^*_{st} \subseteq \widetilde{N}_{st}$ is a node set inducing a minimum capacity *st*-dicut of $\widetilde{G}_{st}$, with respect to $\widetilde{c}$, then there is an *st*-cut or 3-*st*-path-cut inequality violated by $\bar{x}$ if $\widetilde{c}(\delta^+_{\widetilde{G}_{st}}(\widetilde{W}^*_{st})) < 2$. If $\widetilde{c}(\delta^+_{\widetilde{G}_{st}}(\widetilde{W}^*_{st})) \geq 2$, then every *st*-cut or 3-*st*-path-cut inequality is satisfied by $\bar{x}$. Therefore, the separation problem for the *st*-cut and the 3-*st*-path-cut inequalities reduces to computing a minimum *st*-dicut in $\widetilde{G}_{st}$ with respect to the capacity vector $\widetilde{c}$. By the Max Flow-Min Cut Theorem, computing a minimum weight *st*-dicut in $\widetilde{G}_{st}$ is equivalent to computing a maximum flow separating s and t in $\widetilde{G}_{st}$, which can be done in polynomial time.

Now if $\delta^+_{\widetilde{G}_{st}}(\widetilde{W})$ is an *st*-dicut of $\widetilde{G}_{st}$ which does not contain any arc of the form (u,u') and is such that $\widetilde{c}(\delta^+_{\widetilde{G}_{st}}(\widetilde{W})) < 2$, then we need to identifiy if $\delta^+_{\widetilde{G}_{st}}(\widetilde{W})$ corresponds to an *st*-cut or a 3-*st*-path-cut in the original graph G. For this, one can apply the following procedure. First remove from G the edges corresponding to the arcs of $\delta^+_{\widetilde{G}_{st}}(\widetilde{W})$. Then, compute the shortest path (in terms of number of edges) between s and every node of $N \setminus \{s\}$ and let $l(u)$ denote the length of a shortest path between s and node $u \in N \setminus \{s\}$. If $l(t)$ is infinite, then $\delta^+_{\widetilde{G}_{st}}(\widetilde{W})$ corresponds to an *st*-cut of G. If $l(t)$ is finite, then $\delta^+_{\widetilde{G}_{st}}(\widetilde{W})$ corresponds to a 3-*st*-path-cut of G. In this case, the partition $(V_0, \ldots, V_4)$ inducing the 3-*st*-path-cut is such

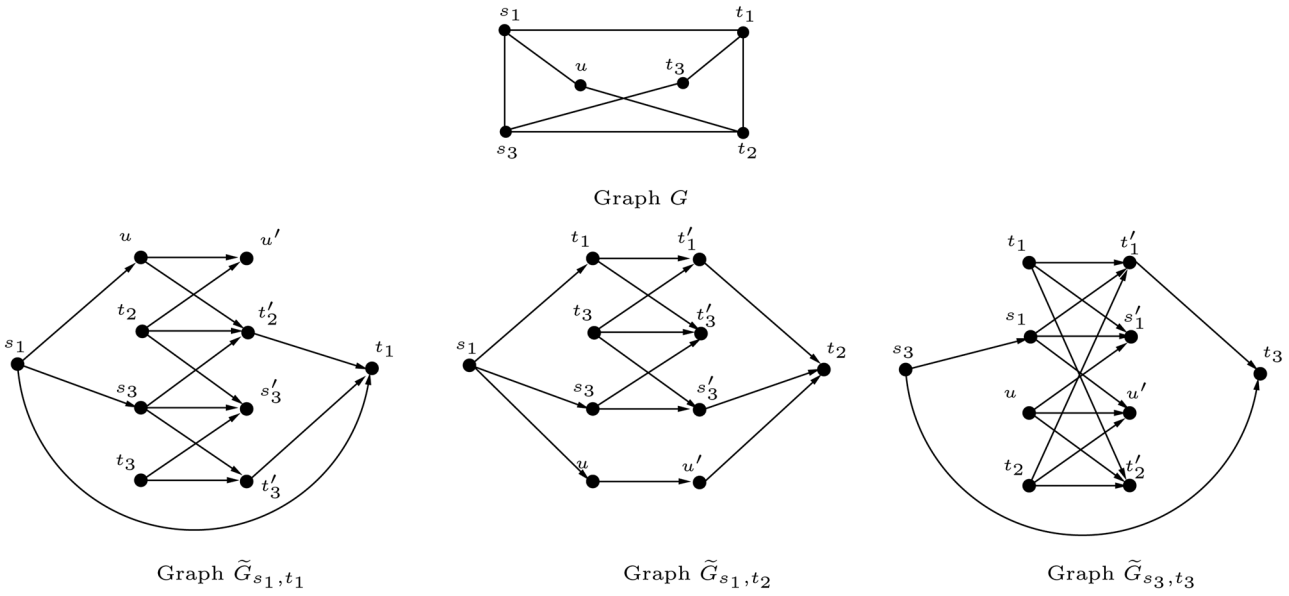

FIG. 10. Construction of graphs $\widetilde{G}_{st}$ with $D = \{(s_1, t_1), (s_1, t_2), (s_3, t_3)\}$.

that $V_0 = \{s\}$, $V_i = \{u \in N \setminus \{s\} : l(u) = i\}$, for $i = 1, 2, 3$, and $V_4 = N \setminus (\bigcup_{i=0}^{3} V_i)$.

The maximum flow computation in $\widetilde{G}_{st}$ can be handled by the Edmonds–Karp algorithm [10] which runs in $O(|\widetilde{A}_{st}|^2|\widetilde{N}_{st}|) = O(|E|^2|N|)$ time. The computation of the shortest paths (to distinguish whether we have found an st-cut or a 3-st-path-cut) can be done using Dijkstra's algorithm [9], which can be implemented to run in $O(|N||E| \log |E|)$ time. As this procedure is performed $|D|$ times (one for each demand), the whole separation algorithm can be implemented to run in $O(|D|(|E|^2|N| + |N||E| \log |E|))$ time, and hence is polynomial. ∎

The next theorem shows that the separation problem for the st-node-cut and the 3-st-node-path-cut inequalities (2.3) and (2.4) can also be solved in polynomial time.

Theorem 22. *The separation problem for the st-node-cut and the 3-st-node-path-cut inequalities (2.3) and (2.4) can be solved using an $O(|D||N|(|E|^2|N| + |N||E| \log |E|))$-time algorithm.*

Proof. For the st-node-cut and the 3-st-node-path-cut inequalities (2.3) and (2.4) induced by a node $z \in N \setminus \{s, t\}$, the separation problem can be addressed using a similar algorithm as that described in Theorem 21 but on the graph $G - z$. We build for each $(s, t) \in D$, the graph $\widetilde{G}^z_{st}$ from $G - z$ and (s, t), and check if the weight of a minimum st-dicut in $\widetilde{G}^z_{st}$ is < 1 or not. If we get an st-cut $\delta^+_{\widetilde{G}^z_{st}}(\widetilde{W})$ of $\widetilde{G}^z_{st}$ whose weight < 1, then we identify if it is a st-node-cut or a 3-st-node-path-cut in a similar way as in Theorem 21. In fact, if $\delta^+_{\widetilde{G}^z_{st}}(\widetilde{W})$ corresponds to an st-dicut of $G - z$, then it is an st-node-cut of G with respect to node z, otherwise, it is a 3-st-node-path-cut of G with respect to node z.

This procedure is repeated for every node $z \in N \setminus \{s, t\}$ and every demand $(s, t) \in D$. Thus, the whole separation algorithm for inequalities (2.3) and (2.4) can be implemented to run in $O(|D||N|(|E|^2|N| + |N||E| \log |E|))$ time. ∎

Now we consider the double cut inequalities (3.12). To the best of our knowledge, the complexity of the separation problem associated with the double cut inequalities (3.12) is still an open question. To separate them, we use the separation heuristic developed in [6]. This heuristic is implemented to run in $O(|N|^3 \log |N| \frac{(2|N|+|D_{\text{source}}|+|D_{\text{dest}}|)^2}{(|N|-1)(|N|+|D_{\text{source}}|+|D_{\text{dest}}|)})$ time (here D_{source} and D_{dest} denote the sets of nodes which are, respectively, the source and destination in a demand), which is polynomial.

We discuss now the separation problem of the rooted partition inequalities (3.11). Recall that a rooted partition inequality is induced by a set of p terminals $t_1, \ldots, t_p$ which are destinations relative to a source node s and a partition $(V_0, V_1, \ldots, V_p)$ such that $s \in V_0$ and $t_i \in V_i$, $i = 1, \ldots, p$.

Notice that Huygens et al. [15] studied the separation problem for the rooted partition inequalities for the TEHNDP. In particular, they showed that the separation problem associated with inequalities (3.11) can be solved in polynomial time when $L = 2$, by using a result from Goemans and Ramakrishnan [11].

For the case when $L \geq 3$, the complexity of the separation problem of inequalities (3.11) is still unknown. Therefore, for our purpose, we have devised a heuristic algorithm for finding violated rooted partition inequalities when $L = 3$. This heuristic looks in particular for inequalities (3.11) induced by a partition $(V_0, V_1, \ldots, V_p)$ with $p \geq 2$, $p(\text{mod}3) \neq 0$ and $|V_i| = 1$, for all $i \in \{1, \ldots, p\}$. As shown in Theorem 20, such inequalities define facets of TNHNDP(G,3).

Our heuristic is based on the computation of a maximum flow in a special directed graph where the arc flows are subject

to upper and lower bounds. First, observe that the inequality (3.11) induced by $(V_0, \ldots, V_p)$, $p \geq 2$, with $s \in V_0$ and $p(\text{mod}3) \neq 0$, is equivalent to

$$x(\delta(V_0)) + \sum_{u \in N \setminus V_0} x(\delta(u)) \geq \frac{8}{3}p + 2\alpha \tag{5.18}$$

with

$$\alpha = \begin{cases} \alpha_1 = \frac{2}{3} & \text{if } p \pmod 3 = 1, \\ \alpha_2 = \frac{1}{3} & \text{if } p \pmod 3 = 2. \end{cases}$$

To see this, observe first that

1. $\left\lceil \frac{4p}{3} \right\rceil$ is obtained by adding a scalar α to $\frac{4p}{3}$ where α is either $\frac{2}{3}$ or $\frac{1}{3}$, depending on whether $p(\text{mod}3) = 1$ or $p(\text{mod}3) = 2$, respectively;
2. $x(\delta(V_0, \ldots, V_p)) = \frac{1}{2}\sum_{i=0}^{p} x(\delta(V_i)) = \frac{1}{2}[x(\delta(V_0)) + \sum_{i=1}^{p} x(\delta(V_i))], = \frac{1}{2}[x(\delta(V_0)) + \sum_{i=1}^{p} x(\delta(t_i))]$.

By combining points 1 and 2, and multiplying the resulting inequality by 2, we can write (3.11) as

$$x(\delta(V_0)) + \sum_{i=1}^{p} x(\delta(t_i)) \geq \frac{8}{3}p + 2\alpha.$$

Now as $p = |N \setminus V_0|$ and, as G is undirected, $x(\delta(V_0)) = x(\delta(N \setminus V_0))$, inequality (5.18) is equivalent to

$$x(\delta(N \setminus V_0)) \geq \sum_{u \in N \setminus V_0} y_u + 2\alpha, \tag{5.19}$$

where $y_u = \frac{8}{3} - x(\delta(u))$, for all $u \in N \setminus V_0$.

Now we describe our separation algorithm. Let $\bar{x} \in \mathbb{R}^E$ which satisfies the trivial inequalities and let $\bar{y} \in \mathbb{R}^V$ with $\bar{y}_u = \frac{8}{3} - \bar{x}(\delta(u))$, for all $u \in V$. Our algorithm starts by contracting with node s all the steiner nodes of G, every terminal $u \in R$ such that $(s, u) \notin D$, and every terminal u such that $(s, u) \in D$ and $\bar{y}_u < 0$ (recall that R denotes the set of terminal nodes of the graph). For convenience, the supernode obtained after these contractions will still be denoted by s. We also denote by $\overline{N}$ the remaining node set after all the contractions (note that $\overline{N}$ is composed of the supernode s and the terminal nodes u such that $(s, u) \in D$ and $\bar{y}_u \geq 0$).

To find violated rooted partition inequalities, we look for violated inequalities (5.19) by looking for a feasible flow with lower bounds in a directed graph $\overline{G} = (\overline{N}, \overline{A})$. Each arc a of $\overline{G}$ is associated with a pair (L_a, U_a), where L_a and U_a are, respectively, the lower and upper bounds of the flow on arc a. For convenience, they will be called capacities. The graph $\overline{G}$ is constructed as follows.

For every edge $uv \in E$ with $\bar{x}(uv) > 0$, we add two arcs (u, v) and (v, u) in $\overline{G}$ with capacities $(0, \bar{x}(uv))$. For every node $u \in \overline{N} \setminus \{s\}$, we add an arc (s, u) in $\overline{G}$ with capacities $(\bar{y}_u, +\infty)$. Finally, we add an arc (s, u_0) for some node $u_0 \neq s$, with capacities $(0, +\infty)$. This latter arc transforms the feasible flow problem into a feasible circulation problem (see [2] for more details). Figure 11 gives an illustration of this graph transformation.

As the flow on each arc of the graph is subject to a lower bound, a flow from a terminal, say $t_1 \in \overline{N} \setminus \{s\}$, to s in $\overline{G}$ may not be feasible with respect to these lower bounds. As shown in [2], there exists a feasible flow in $\overline{G}$ if and only if every cut of $\overline{G}$ has a non-negative capacity, where the capacity of a cut induced by a node set $W \subseteq \overline{N}$ is defined by $\sum_{a \in \delta^+_{\overline{G}}(W)} U_a - \sum_{a \in \delta^-_{\overline{G}}(W)} L_a$.

It is not hard to see that the capacity of the cut induced by W is

$$\sum_{a \in \delta^+_{\overline{G}}(W)} U_a - \sum_{a \in \delta^-_{\overline{G}}(W)} L_a = \sum_{e \in \delta_G(W)} \bar{x}(e) - \sum_{t \in W} \bar{y}_t.$$

For our separation algorithm, we distinguish two cases.

Case 1. If there is no feasible flow from t_1 to s, then there exists a node set $W \subseteq \overline{N}$ with $W = \{t_1, t_2, \ldots, t_p\}$, $s \in N \setminus W$ and whose capacity is such that

$$\sum_{a \in \delta^+_{\overline{G}}(W)} U_a - \sum_{a \in \delta^-_{\overline{G}}(W)} L_a = \sum_{e \in \delta_G(W)} \bar{x}(e) - \sum_{t \in W} \bar{y}_t < 0 < 2\alpha.$$

As $s \notin W$, inequality (5.19) induced by W is hence violated by $\bar{x}$. From W, we build a rooted partition $\pi = (V_0, V_1, \ldots, V_p)$ with $V_0 = N \setminus W$, $V_i = \{t_i\}$, $i = 1, \ldots, p$. Obviously, if $|W|(\text{mod}3) \neq 0$, then the rooted partition inequality (3.11) induced by π is violated by $\bar{x}$.

Case 2. If there exists a feasible flow from t_1 to s, then we compute a maximum feasible flow. We then look for a minimum capacity cut in $\overline{G}$ separating t_1 and s. As shown in [2], a minimum capacity cut in $\overline{G}$ can be obtained by computing the value of a maximum feasible flow from t_1 to s. Let f denote the value of the minimum capacity of a cut and let W be a node set inducing such a minimum capacity cut. If $|W|(\text{mod}3) = 0$, then we choose another terminal, say t_2, and repeat the procedure. Observe that, as there exists a feasible flow from t_1 to s, there also exists a feasible flow from any node of $\overline{G}$ to s (see [2]). If $|W|(\text{mod}3) \neq 0$, then we check if $f < 2\alpha_1$ (resp., $f < 2\alpha_2$) for $|W|(\text{mod}3) = 1$ (resp., $|W|(\text{mod}3) = 2$). If it is the case, then we have found a violated rooted partition inequality. This is built from the node set W as in Case 1.

We repeat the above procedure for each terminal until we find a violated rooted partition inequality or all the terminals of $\overline{N}$ have been selected and no violated inequality has been found.

The separation algorithm is summarized in Algorithm 1 below.

Notice that, as mentioned in [2], computing a maximum feasible flow, if there is any, can be solved in polynomial time by computing two classical maximum flows in $\overline{G}$ (that is, flows where the arcs are not subject to lower bounds). These maximum flows can be computed using Goldberg and

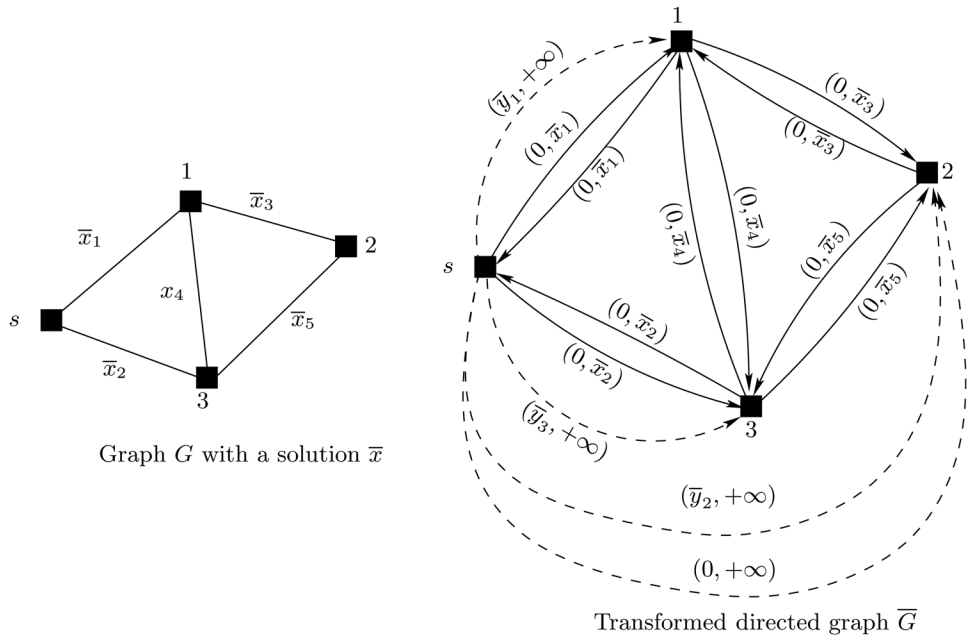

FIG. 11. A graph G, a solution $\overline{x}$ and the corresponding graph $\overline{G}$ with its arc flow upper and lower bounds.

Tarjan's algorithm [12] which runs in $O(|\overline{N}||\overline{A}| \log \frac{|\overline{N}|^2}{|\overline{A}|}) = O(|N|(|E| + |N|) \log \frac{|N|^2}{|E|+|N|})$ time.

The algorithm described in [2] produces the value of a maximum feasible flow as well as a node set W inducing a cut whose capacity, $\sum_{a \in \delta^+_{\overline{G}}(W)} U_a - \sum_{a \in \delta^-_{\overline{G}}(W)} L_a$, is minimum. In the case where there is no feasible flow, the algorithm in [2] produces a node set W inducing a cut with negative capacity, by computing a maximum flow and a set of augmenting paths. This can be done using Goldberg and Tarjan's algorithm [12] for the maximum flow and the Edmonds–Karp algorithm [10] for augmenting paths. Therefore, our separation heuristic for the rooted partition inequalities can be implemented to run in $O(|D|[|N|(|E|+|N|) \log \frac{|N|^2}{|E|+|N|} + (|E| + |N|)^2|N|])$ time, which is polynomial.

5.3. *Primal Heuristic*

An important issue in the efficiency of the Branch-and-Cut algorithm is computing a good upper bound at each node of the Branch-and-Cut tree. We use a primal heuristic described in Algorithm 2 to find upper bounds. Let $\overline{x} \in \mathbb{R}^E$ be the current solution and assume that $\overline{x}$ satisfies the trivial inequalities and is fractional. We start by fixing a threshold value α (α is set to 0.80 at the beginning) and removing from G every edge $e \in E$ such that $\overline{x}(e) \leq \alpha$. Then, we check if the resulting graph, denoted by $G(\overline{x}, \alpha)$, induces a feasible solution of the TNHNDP. To do this, we compute, for every $(s,t) \in D$, the graph $\widetilde{G}_{st}$ (see proof of Theorem 21) obtained from $G(\overline{x}, \alpha)$, and compute the number of node-disjoint st-dipaths in $\widetilde{G}_{st}$, denoted by λ_{st}. If $\lambda_{st} \geq 2$ for all $(s,t) \in D$, then $G(\overline{x}, \alpha)$ induces a feasible solution of the TNHNDP. If not, then we decrease the threshold value α by 0.05 and repeat the same procedure. The procedure is repeated until $G(\overline{x}, \alpha)$ induces a feasible solution. Notice that, there always exists a value of $\alpha \leq 0.80$ for which $G(\overline{x}, \alpha)$ induces a feasible solution. In fact, for $\alpha = 0$, $G(\overline{x}, \alpha)$ is exactly the original graph G, which is assumed to induce a feasible solution of the problem. Thus, in the worst case, the algorithm will produce a feasible solution when α is decreased to 0.

After finding a feasible solution, with a given threshold value α_0, we try to improve the solution by removing from $G(\overline{x}, \alpha_0)$ all the edges that are not used in any 3-st-path met during the feasibility test phase of the algorithm. The complete algorithm is summarized in Algorithm 2 below.

To compute the number of node-disjoint st-dipaths in $\widetilde{G}_{st}$, for every $(s,t) \in D$, we use a variant of the Edmonds–Karp algorithm [10]. We recall that the Edmonds–Karp algorithm computes a maximum st-flow in a directed graph by iteratively computing a set of augmenting st-dipaths. Moreover, by assigning a unit capacity to each arc of the graph, the obtained augmenting st-dipaths are mutually edge-disjoint. To compute the number of node-disjoint st-dipaths in $\widetilde{G}_{st}$, we modify the Edmonds–Karp algorithm in the following way. First, we give unit capacity to each arc of $\widetilde{G}_{st}$. Then, we compute an augmenting st-dipath in $\widetilde{G}_{st}$ using the same procedure as that of Edmonds–Karp. If this path is of the form (s, t), then we look for another augmenting path. If this st-dipath is of the form (s, u, v', t) (note that v may be equal to u), then we remove u and v', and the corresponding nodes u' and v, from $\widetilde{G}_{st}$. Notice that in this way, any augmenting st-dipath obtained by the algorithm will be node-disjoint from the others. As, by its construction, any st-dipath in $\widetilde{G}_{st}$ corresponds to a 3-st-path of G, the number of node-disjoint st-dipaths of $\widetilde{G}_{st}$ will correspond to the number of node-disjoint 3-st-paths of G.

It is not hard to see that the main issue in the algorithm is the computation of the number of node-disjoint st-dipaths

Algorithm 1 Separation algorithm for rooted partition inequalities

Data: An undirected graph $G = (N, E)$, demands $D \subseteq N \times N$, a fractional solution $\overline{x}$
Result: A set $\mathcal{F}^*$ of violated rooted partition inequalities

begin
  $\mathcal{F} \leftarrow \emptyset$;
  $\alpha_1 \leftarrow \frac{2}{3}$;
  $\alpha_2 \leftarrow \frac{1}{3}$;
  Compute the vector $\overline{y} \in \mathbb{R}^V$;
  In graph G, contract with node s every steiner node and every terminal $u \in R$ such that $\overline{y}(u) < 0$;
  Build the directed graph $\overline{G}$ and set up flow upper and lower bounds on the arcs of $\overline{G}$ as defined above;
  for *each* $t \in R$ **do**
    Compute a feasible flow in G from t to s;
    if *there is no feasible flow* **then**
      Let $W = \{t_1, ..., t_p\}$, $p \leq |D|$, be a node set obtained at Step 9;
      `/*There exists a violated rooted partition inequality*/`
      Build the partition $\pi = (V \setminus W, \{t_1\}, ..., \{t_p\})$;
      if $\pi \notin \mathcal{F}$ **then**
        $\mathcal{F} \leftarrow \mathcal{F} \cup \{\pi\}$;
    else
      Compute a maximum feasible flow in G from t to s. Let f be the value of such maximum flow;
      Let $W = \{t_1, ..., t_p\}$ be a node set whose capacity equals f;
      if *(*$|W| \pmod 3 = 1$ *and* $f < 2\alpha_1$*) or (*$|W| \pmod 3 = 2$ *and* $f < 2\alpha_2$*)* **then**
        `/*There exists a violated rooted partition inequality*/`
        Build the partition $\pi = (V \setminus W, \{t_1\}, ..., \{t_p\})$;
        if $\pi \notin \mathcal{F}$ **then**
          $\mathcal{F} \leftarrow \mathcal{F} \cup \{\pi\}$;
  Return $\mathcal{F}$;
end

in $\widetilde{G}_{st}$ for every $(s,t) \in D$ and each value of $\alpha \leq 0.80$. This can be done in at most $(|N| - 1)(|N| - 2)$ iterations for each $(s,t) \in D$ and $\alpha \leq 0.80$. Moreover, the number of iterations of the main loop of Algorithm 2 is bounded by $16 = 0.80/0.05$. Thus, Algorithm 2 runs in $O(|D||N|^2)$ time, which is polynomial.

6. COMPUTATIONAL RESULTS

The Branch-and-Cut algorithm presented in the previous section has been implemented in C++, using ABACUS 3.0 [1] to manage the Branch-and-Cut tree and CPLEX 12.2 as LP-solver.

Algorithm 2 Primal Heuristic Algorithm for TNHNDP

Data: An undirected graph $G = (N, E)$, demands $D \subseteq N \times N$, a fractional solution $\overline{x}$
Result: An Upper Bound UB for the TNHNDP

begin
  $\alpha \leftarrow 0.80$;
  FeasibleSolutionFound $\leftarrow$ *False*;
  repeat
    Build the graph $G(\overline{x}, \alpha)$ by removing from G every edge e such that $\overline{x}(e) \leq \alpha$;
    `/*Check if G(x,α) induces a feasible solution*/`
    for *each demand* $(s,t) \in D$ **do**
      Build the directed graph $\widetilde{G}_{st}$ from $G(\overline{x}, \alpha)$;
      Compute λ_{st}, the number of node-disjoint st-dipaths in $\widetilde{G}_{st}$;
      for *all* $a \in \widetilde{A}_{st}$ **do**
        if *a is used in a node-disjoint st-dipath obtained at Step 9* **then**
          Label the edge e of G corresponding to arc a;
      if $\lambda_{st} < 2$ **then**
        Go to Step 19;
    for *each edge e of* $G(\overline{x}, \alpha)$ **do**
      Remove from $G(\overline{x}, \alpha)$ every edge that is not labelled;
    FeasibleSolutionFound $\leftarrow$ *True*;
    $UB \leftarrow$ the total weight of $G(\overline{x}, \alpha)$;
    $\alpha \leftarrow \alpha - 0.05$;
  until *FeasibleSolutionFound* = *True*;
  return UB;
end

It was tested on a Xeon Quad-Core E5507 machine with a 2.27 GHz processor and 8 GB RAM, running under Linux. The maximum CPU time has been fixed to 5 h. The test problems are composed of complete graphs coming from TSPLIB [16] or that are randomly generated on a 250×250 grid. In both cases (TSPLIB and random instances), the nodes of the graphs are assigned coordinates in the plane. The weight of each edge is taken as the Euclidean distance between the end nodes of the edge. The demands used in these tests are randomly generated. Each set of demands is either rooted, that is, of the form $\{(s, t_i) : i = 1, \ldots, d\}$ (s is the root node of the demands), or arbitrary.

For each instance, the Branch-and-Cut algorithm has been run twice. The first run (run 1) is performed with all the inequalities listed in Section 5.1. The second run (run 2) is done with only the natural inequalities (st-cut, st-node-cut, L-st-path-cut and L-st-node-path-cut inequalities).

For the random problems, five instances have been tested and the results given represent averages. Finally, the instances having graphs with up to 30 nodes will be considered as small size instances, while those having graphs with more than 30 nodes will be considered as large size instances.

The computational results are given in Tables 1–4. Each instance is described by the number of nodes of the graph and the number of demands. The number of nodes is preceded either by "r" if the demands are rooted or "a" if they are not rooted. The entries of the various tables presented below are:

$\|N\|$	the number of nodes of the graph,
$\|D\|$	the number of demands,
NC	the number of generated *st*-cut inequalities,
NNC	the number of generated *st*-node-cut inequalities,
LPC	the number of generated *L-st*-path-cut inequalities,
LPNC	the number of generated *L-st*-node-path-cut inequalities,
RP	the number of generated rooted partition inequalities,
DC	the number of generated double cut inequalities,
COpt	value of the best upper bound obtained,
Gap1	the relative error 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 relative error 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 relative error between the best upper bound and the best lower bound of the Branch-and-Cut tree in the first run,
Gt2	the relative error between the best upper bound and the best lower bound of the Branch-and-Cut tree in the second run,
Tree1	the number of nodes in the Branch-and-Cut tree in the first run,
Tree2	the number of nodes in the Branch-and-Cut tree in the second run,
CPU1	total CPU time of the first run in hours:min.sec.,
CPU2	total CPU time of the second run in hours:min.sec.

Note that a value of ∞ for Gap1 or Gap2 means that the algorithm spends all the CPU time (5 h) at the root node of the Branch-and-Cut tree. Also, a value of 0 for Gt1 or Gt2 means that the algorithm has obtained the optimal solution within the CPU time limit while a non-zero value indicates that the optimal solution has not been reached. Finally, a value of ∞ for Gt1 and Gt2 means that the algorithm ends without finding any feasible solution (upper bound is ∞) and that all the CPU time has been spent at the root node of the Branch-and-Cut tree.

Our first series of computations concerns the rooted demands for both TSPLIB and random instances. The results are given in Tables 1 and 2.

We can see that, at the first run, for rooted TSPLIB instances (Table 1), the algorithm has solved to optimality 6 instances out of 15, with graphs having up to 30 nodes and with 10 demands, and the CPU time varies from 1 s to 27 min. The gap achieved between the best upper bound (that is, the optimal solution) and the value at the root node of the Branch-and-Cut tree (Gap1) is relatively small (less than 8%) for these instances. For the instances that have not been solved to optimality, the value of the gaps (Gap1 and Gt1) are relatively high: more than 20% for Gap1 and more than 11% for Gt1. For some instances (like r-58-40) Gap1 and Gt1 reach, respectively, 76.13% and 75.31%. The number of nodes in the Branch-and-Cut tree is relatively small (less than 100 nodes in 5 h of computations) for those instances. For one instance, r-58-40, the algorithm has not been able to get even a feasible solution and spends all the CPU time at the root node of the Branch-and-Cut tree (Gap1 and Gt1 are ∞). Table 1 also shows that a large number of natural inequalities, especially 3-*st*-path-cut inequalities, have been generated during the resolution process. Rooted partition inequalities have also been generated for 14 instances out of 15 and no double cut inequalities have been generated for any of the instances.

For the rooted random instances (Table 2), the observations are quite similar. Here 5 instances out of 16 have been solved to optimality with average CPU time between 1 s and 56 min for graphs with fewer than 30 nodes. The average gap Gap1 is less than 10%. For the instances not solved to optimality, the average gaps Gap1 are relatively high (more than 20%), and the number of nodes in the Branch-and-Cut tree is, in general, relatively small. Here also, we observe that a large number of natural inequalities are generated during the resolution process and rooted partition inequalities have been generated for all the instances. A few number of double cut inequalities have been generated.

The Branch-and-Cut algorithm has also been run for TSPLIB and random instances with arbitrary demands (Tables 3 and 4 below). Recall that the rooted partition inequalities have not been applied for arbitrary instances. Thus, the corresponding column does not appear in Tables 3 and 4.

As for the rooted demands, we can see that the algorithm has solved to optimality the instances with graphs having fewer than 30 nodes. For the instances not solved to optimality, the gap Gap1 is relatively high (more than 19%) for both TSPLIB and random instances. We also notice that a large number of natural inequalities are generated and, contrary to the rooted instances, several double cut inequalities have been generated during the resolution process.

Our next series of experiments aims to check the efficiency of the different classes of inequalities used in the Branch-and-Cut algorithm. For this, we have first run the algorithm without the rooted partition and double cut inequalities and using the natural inequalities (i.e., run 2) for all the instances and compared the results with those obtained in run 1. The comparison between run 2 and run 1 for the rooted TSPLIB instances (Table 1) shows that, for the instances solved to optimality in run 1, the performance of the algorithm is decreased in run 2. In fact, the CPU time CPU2 is greater for all these instances. Also, in general, the gap Gap2 and the number of nodes in the Branch-and-Cut tree Tree2 are also increased with respect to Gap1 and Tree1. Moreover, for one instance r-30-10, the algorithm in run 2 has not been able to solve the problem to optimality within the CPU time limit while it was able to solve it to optimality in run 1. For the rooted random instances (Table 2) and for the arbitrary instances (Tables 3

TABLE 1. Results for TSPLIB instances with $L = 3$ and rooted demands

$\|N\|$	$\|D\|$	NC	NNC	LPC	LPNC	RP	DC	COpt	Gap1	Gap2	Gt1	Gt2	Tree1	Tree2	CPU1	CPU2
r 10	5	26	2	21	0	2	0	2,387	0.87	1.87	0	0	9	11	0:00:01	0:00:01
r 10	7	20	12	52	0	2	0	3,022	2.87	7.69	0	0	11	33	0:00:01	0:00:01
r 14	10	42	11	901	31	11	0	4,085	6.82	9.50	0	0	105	171	0:00:02	0:00:05
r 14	7	44	8	438	6	6	0	3,237	8.40	10.38	0	0	57	53	0:00:01	0:00:01
r 17	16	69	23	7,838	1,176	50	0	3,051	8.22	14.65	0	0	1,391	4,233	0:05:45	0:20:20
r 30	10	367	722	5,304	439	17	0	5,002	7.51	9.00	0	2.98	1,385	481	0:27:13	5:00:00
r 30	15	444	646	42,019	1,598	75	0	7,843	20.61	29.28	11.17	19.24	3,071	3,007	5:00:00	5:00:00
r 48	10	363	194	55,664	27	9	0	12,302	40.55	29.59	38.32	27.81	17	15	5:00:00	5:00:00
r 48	15	261	177	23,285	230	21	0	13,712	38.70	45.07	33.51	40.96	55	45	5:00:00	5:00:00
r 52	10	767	901	35,494	48	10	0	6,243	27.09	25.68	23.90	23.34	71	35	5:00:00	5:00:00
r 58	20	520	0	67,061	0	0	0	-	∞	∞	∞	∞	1	1	5:00:00	5:00:00
r 58	30	573	25	41,371	10	17	0	96,593	69.32	71.87	68.31	71.87	3	3	5:00:00	5:00:00
r 58	40	439	0	44,463	0	2	0	137,712	76.13	74.87	75.31	74.75	1	7	5:00:00	5:00:00
r 76	20	413	0	57,278	1	12	0	768	52.14	45.03	51.24	43.33	9	17	5:00:00	5:00:00
r 76	40	435	0	28,755	0	31	0	1,764	69.28	∞	66.75	∞	13	1	5:00:00	5:00:00

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

$\|N\|$	$\|D\|$	NC	NNC	LPC	LPNC	RP	DC	COpt	Gap1	Gap2	Gt1	Gt2	Tree1	Tree2	CPU1	CPU2
r 10	5	24	14	98	5	1	0	841	8	12	0	0	35	31	0:00:01	0:00:01
r 10	7	25	16	461	47	2	6	1,100	12	14	0	0	129	120	0:00:01	0:00:01
r 14	10	44	11	2,736	286	22	0	1,141	13	18	0	0	755	1,536	0:00:41	0:01:09
r 14	7	91	22	1,585	79	12	0	1,030	14	16	0	0	337	258	0:00:12	0:00:05
r 17	16	19	44	19,642	4,055	191	0	1,346	10	20	0	5	17,713	29,711	0:56:14	3:02:23
r 30	10	194	6	3,137	47,840	11	0	1,691	35	39	31	36	91	553	5:00:00	5:00:00
r 30	15	138	51	40,619	336	29	0	2,129	37	40	34	39	395	765	5:00:00	5:00:00
r 48	10	288	0	45,077	128	9	0	1,413	20	25	16	23	35	136	5:00:00	5:00:00
r 48	15	204	35	35,558	5	12	0	2,542	50	56	47	51	25	66	5:00:00	5:00:00
r 52	10	455	29	31,814	128	16	0	1,434	30	36	26	33	59	129	5:00:00	5:00:00
r 52	20	424	87	31,142	97	23	0	2,884	55	64	53	61	187	381	5:00:00	5:00:00
r 58	20	390	20	85,628	15	15	0	2,788	49	53	48	51	25	12	5:00:00	5:00:00
r 58	30	318	0	30,475	0	9	0	-	∞	∞	∞	∞	1	1	5:00:00	5:00:00
r 58	40	154	0	26,186	30	32	0	11,837	81	85	81	83	49	81	5:00:00	5:00:00
r 76	20	622	0	65,835	0	7	0	3,143	53	57	52	55	7	7	5:00:00	5:00:00
r 76	40	709	0	76,057	0	18	0	13,361	83	88	83	88	9	41	5:00:00	5:00:00

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

$\|N\|$	$\|D\|$	NC	NNC	LPC	LPNC	DC	COpt	Gap1	Gap2	Gt1	Gt2	Tree1	Tree2	CPU1	CPU2
a 10	5	9	2	188	1	15	3,500	9.48	9.48	0	0	49	49	0:00:01	0:00:01
a 14	10	45	13	5,975	515	23	4,421	14.70	14.70	0	0	563	575	0:01:08	0:01:29
a 14	7	4	0	191	0	0	3,938	2.14	2.14	0	0	9	9	0:00:01	0:00:01
a 17	45	516	80	47,865	4,021	61	2,707	21.71	30.65	0	14.97	1,989	795	0:42:11	5:00:00
a 17	8	30	6	17,260	671	68	3,233	14.18	14.18	0	0	1,789	2,259	0:14:51	0:20:03
a 30	10	71	2	33,687	11	0	7,308	19.04	19.04	15.64	15.64	41	41	5:00:00	5:00:00
a 30	15	19	0	40,863	11	0	13,882	43.97	43.97	43.00	42.91	91	101	5:00:00	5:00:00
a 48	10	121	16	27,165	31	0	17,824	42.14	42.14	39.99	40.01	19	13	5:00:00	5:00:00
a 48	15	65	0	24,306	0	0	22,731	58.90	58.90	58.79	58.79	1	1	5:00:00	5:00:00
a 52	10	194	4	23,460	3	0	7,268	19.15	19.15	15.87	15.87	35	35	5:00:00	5:00:00
a 52	20	60	0	29,551	0	0	21,250	56.35	56.35	56.14	56.11	29	35	5:00:00	5:00:00

and 4), the observations are the same. The CPU time CPU2, the gap Gap2 and the number of nodes in the Branch-and-Cut tree Tree2, generally increase in run 2 compared to run 1. This shows that, as expected, using the natural inequalities only is not effective for solving the TNHNDP and additional inequalities (mainly rooted partition inequalities for rooted demands and double cut inequalities for arbitrary demands) may considerably improve the algorithm for instances with graphs having less than 30 nodes.

For the instances that are not solved to optimality in run 1, the influence of the rooted partition and double cut inequalities is less clear, mainly for TSPLIB instances (Tables 1 and 3). First, it can be noticed that for TSPLIB instances with arbitrary demands (Table 3), the results obtained in runs 1

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

$\|N\|$	$\|D\|$	NC	NNC	LPC	LPNC	DC	COpt	Gap1	Gap2	Gt1	Gt2	Tree1	Tree2	CPU1	CPU2
a 10	5	6	0	195	8	7	1,063	9	10	0	0	47	91	0:00:01	0:00:01
a 14	10	49	20	9,301	1,099	33	1,179	19	22	0	0	977	1,454	0:02:47	0:06:40
a 14	7	6	0	12,043	834	69	1,283	15	18	0	0	3,045	2,067	0:09:13	0:07:57
a 17	45	575	54	118,509	16,839	605	1,214	28	30	0	6	12,025	8,140	5:00:00	5:00:00
a 17	8	22	8	17,997	731	78	1,315	12	21	0	14	2,519	279	0:16:38	3:21:31
a 30	10	43	0	42,166	44	3	1,778	21	36	18	33	133	711	5:00:00	5:00:00
a 30	15	10	0	38,339	11	0	4,722	56	54	56	54	15	15	5:00:00	5:00:00
a 48	10	79	0	33,040	7	0	2,350	36	∞	32	∞	97	1	5:00:00	5:00:00
a 48	15	40	0	42,978	2	0	9,906	80	80	80	80	5	10	5:00:00	5:00:00
a 52	10	61	0	25,105	7	0	2,412	46	46	45	45	17	16	5:00:00	5:00:00
a 52	20	64	0	67,952	0	0	-	∞	∞	∞	∞	1	1	5:00:00	5:00:00

TABLE 5. Results for TSPLIB instances with $L = 3$, rooted demands and without the primal heuristic

$\|N\|$	$\|D\|$	Copt	Gap1	Gap4	Gt1	Gt4	Tree1	Tree4	CPU1	CPU4
r 10	5	2,387	0.87	0.87	0	0	9	9	0:00:01	0:00:01
r 10	7	3,022	2.87	2.87	0	0	11	11	0:00:01	0:00:01
r 14	10	4,085	6.82	6.82	0	0	105	117	0:00:02	0:00:04
r 14	7	3,237	8.40	8.4	0	0	57	59	0:00:01	0:00:01
r 17	16	3,051	8.22	8.22	0	0	1,391	2,317	0:05:45	0:12:27
r 30	10	5,002	7.51	7.51	0	0	1,385	1,385	0:27:13	0:27:10
r 30	15	7,843	20.61	∞	11.17	∞	3,071	5,485	5:00:00	5:00:00
r 48	10	12,302	40.55	∞	38.32	∞	17	17	5:00:00	5:00:00
r 48	15	13,712	38.70	∞	33.51	∞	55	55	5:00:00	5:00:00
r 52	10	6,243	27.09	∞	23.90	∞	71	71	5:00:00	5:00:00
r 58	30	96,593	69.32	∞	68.31	∞	3	3	5:00:00	5:00:00
r 76	20	768	52.14	∞	51.24	∞	9	9	5:00:00	5:00:00
r 76	40	1,764	69.28	∞	66.75	∞	13	13	5:00:00	5:00:00

and 2 for the instances that are not solved to optimality in run 1, are quite similar. This is explained by the fact that no double cut inequality is generated for these instances, and that rooted partition inequalities are not used for arbitrary demands. Now, for the rooted TSPLIB instances not solved to optimality in run 1 (Table 1), we observe, for some of them (e.g., r-30-15), that the gaps Gap and Gt are better in run 1 than in run 2, while for some other instances (e.g., r-52-10), the results are better when rooted partition inequalities are not used. However, we can remark that for the instances where run 1 produces better results than run 2, several rooted partition inequalities have been generated during run 1. Moreover, for one instance (r-76-40), the algorithm was not able to find even a feasible solution after the CPU time limit, when the rooted partition and double cut inequalities are not used. This tends to show that the rooted partition and double cut inequalities are effective in solving the problem.

We have also compared the efficiency of the rooted partition inequalities against the double cut inequalities. For this, we have solved the problem without the rooted partition inequalities and using the natural and double cut inequalities, for the same instances as those of Table 1. We first observe that no double cut inequalities have been generated when only rooted partition inequalities are not used, and that the results are quite similar to those obtained in run 2 (remind that run 2 is the algorithm without both rooted partition and double cut inequalities). This clearly shows that the rooted partition inequalities are more effective than the double cut inequalities, for rooted instances.

As for the inequalities, we have also checked the efficiency of the primal heuristic that has been used in the computations and described by Algorithm 2. For this, we have run, in run 4, our Branch-and-Cut algorithm after deactivating the primal heuristic, and compared the results obtained to those obtained during run 1. Namely, we have compared the values achieved, during runs 1 and 4, for the gaps Gap and Gt, the number of nodes in the Branch-and-Cut tree and the total CPU time. Recall that the primal heuristic is called with a fractional solution $\bar{x}$ satisfying the natural inequalities and when no rooted partition and double cut inequalities violated by $\bar{x}$ are found during the separation phase. The test has been done for TSPLIB instances and both rooted and arbitrary demands. We have considered only instances for which the number of nodes in the Branch-and-Cut tree achieved in run 1 (Tree1) is at least 2. Indeed, for instances having Tree1 equal to 1, the Branch-and-Cut algorithm spends all the CPU time in the cutting plane phase at the root node of the tree, and stops before calling the primal heuristic. Thus, for these instances, the results are the same with and without using the primal heuristic. The results of this run are given in Tables 5 and 6 for rooted and arbitrary demands, respectively.

From Tables 5 and 6, we can see first that instances that are solved to optimality in run 1 are also solved to optimality in run 4 (i.e., without the primal heuristic). Indeed, the gaps

TABLE 6. Results for TSPLIB instances with $L = 3$, arbitrary demands and without the primal heuristic

$\|N\|$	$\|D\|$	Copt	Gap1	Gap4	Gt1	Gt4	Tree1	Tree4	CPU1	CPU4
a 10	5	3,500	9.48	9.48	0	0	49	49	00:0:01	0:00:01
a 14	10	4,421	14.70	14.7	0	0	563	563	0:01:08	0:01:28
a 14	7	3,938	2.14	2.14	0	0	9	9	0:00:01	0:00:01
a 17	45	2,707	21.71	21.71	0	0	1,989	2,045	0:42:11	0:55:52
a 17	8	3,233	14.18	14.18	0	0	1,789	1,795	0:14:51	0:19:40
a 30	10	7,308	19.04	∞	15.64	∞	41	41	5:00:00	5:00:00
a 30	15	13,882	43.97	∞	43.00	∞	91	91	5:00:00	5:00:00
a 48	10	17,824	42.14	∞	39.99	∞	19	19	5:00:00	5:00:00
a 52	10	7,268	19.15	∞	15.87	∞	35	35	5:00:00	5:00:00
a 52	20	21,250	56.35	∞	56.14	∞	29	29	5:00:00	5:00:00

TABLE 7. Results for TSPLIB instances with $L = 3$, rooted demands and all inequalities separated simultaneously

$\|N\|$	$\|D\|$	NC	NNC	LPC	LPNC	RP	DC	COpt	Gap3	Gap1	Gt3	Gt1	Tree3	Tree1	CPU3	CPU1
r 10	5	26	67	32	13	4	0	2,387	1.87	0.87	0	0	9	9	0:00:01	0:00:01
r 10	7	17	63	59	32	4	0	3,022	7.69	2.87	0	0	11	11	0:00:01	0:00:01
r 14	10	41	270	824	357	22	0	4,085	9.50	6.82	0	0	105	105	0:00:07	0:00:02
r 14	7	30	158	390	191	12	0	3,237	10.38	8.40	0	0	57	57	0:00:01	0:00:01
r 17	16	42	545	8,878	4,992	108	0	3,051	14.65	8.22	0	0	2,267	1,391	0:52:39	0:05:45
r 30	10	242	2,864	3,676	3,835	32	0	5,002	9.00	7.51	0.98	0	987	1,385	5:00:00	0:27:13
r 30	15	262	2,580	10,222	5,317	86	0	8,477	30.14	20.61	19.44	11.17	781	3,071	5:00:00	5:00:00
r 48	10	254	4,092	3,602	12,041	20	0	11,114	35.37	40.55	28.38	38.32	75	17	5:00:00	5:00:00
r 48	15	197	6,577	9,342	10,237	56	0	12,731	33.98	38.70	27.66	33.51	89	55	5:00:00	5:00:00
r 52	10	539	9,477	1,994	11,985	20	0	6,243	27.09	27.09	24.35	23.9	47	71	5:00:00	5:00:00
r 58	20	322	2,138	7,462	25,406	42	0	60,925	56.55	∞	55.15	∞	13	1	5:00:00	5:00:00
r 58	30	369	1,478	15,719	23,858	2	0	105,343	71.87	69.32	71.27	68.31	1	3	5:00:00	5:00:00
r 58	40	272	719	11,895	20,626	2	0	137,712	76.13	76.13	75.46	75.31	1	1	5:00:00	5:00:00
r 76	20	428	3,913	6,603	24,175	24	0	775	54.54	52.14	51.69	51.24	11	9	5:00:00	5:00:00
r 76	40	352	1,464	12,519	26,366	34	0	1,686	67.86	69.28	65.30	66.75	9	13	5:00:00	5:00:00

Gt1 and Gt4 are both equal to 0 for these instances. We can also see that the number of nodes in the Branch-and-Cut tree as well as the total CPU time are either the same or increased, when the primal heuristic is not used. For example, for instance r-17-16 in Table 5 the number of nodes in the Branch-and-Cut tree increases from 1,391 to 2,317, and the total CPU time increases from 5 min and 45 s to 12 min. The same observation holds for instance a-17-45 in Table 6. This latter observation shows that using the primal heuristic permits one to speed up the resolution of the problem for these instances.

For instances that are not solved to optimality in run 1, the Branch-and-Cut algorithm is still not able to solve them to optimality when the primal heuristic is not used. Moreover, we can see in both Tables 5 and 6 that for these instances, the Branch-and-Cut algorithm is not able to find a feasible solution after 5 h of computation (Gap4 and Gt4 are both ∞). Furthermore, the heuristic produces, in some cases, upper bounds of good quality, as for instances like r-30-15 in Table 5 and a-30-10 and a-52-10 in Table 6, the gap Gap1 is $\leq 20\%$. Therefore, the primal heuristic developed here is useful in the Branch-and-Cut algorithm.

We conclude this computational study by addressing an issue met when solving the TNHNDP in run 1, especially for large size instances. Indeed, we have observed that the Branch-and-Cut algorithm spends a lot of time in solving the different LPs it creates. This reduces the total CPU time spent for the exploration of the Branch-and-Cut tree and may prevent finding good solutions in short time. It may also prevent finding even feasible solutions as, in some cases, the algorithm spends all the CPU time at the root node of the Branch-and-Cut tree (see for example instances r-58-20 and r-58-40 in Table 1 and instance a-48-15 in Table 3). To address this issue, we have run the Branch-and-Cut algorithm by separating the natural inequalities, rooted partition, and double cut inequalities simultaneously, instead of separating them according to the order described in Section 5.1. In this way, one may reduce the number of linear programs to solve during the entire algorithm and reduce the total CPU time allowed to solve LPs and allow more CPU time for the exploration of the Branch-and-Cut tree. Also, one may strengthen the linear relaxation of the problem. This separation strategy composes run 3, and the results for TSPLIB instances are presented in Tables 7 and 8. Notice that the number of inequalities presented in Tables 7 and 8 are those obtained for each instance in run 3.

The comparison between run 3 and run 1 for rooted instances (Table 7) shows that for the instances solved to optimality at run 1, the CPU time increases at run 3. Also, for one instance, r-30-10, the algorithm was not able to solve the problem within the CPU time limit while it is solved to optimality in run 1. Moreover, the gap Gap3 increases when all the

TABLE 8. Results for TSPLIB instances with $L = 3$, arbitrary demands and all inequalities separated simultaneously

$\|N\|$	$\|D\|$	NC	NNC	LPC	LPNC	DC	COpt	Gap3	Gap1	Gt3	Gt1	Tree3	Tree1	CPU3	CPU1
a 10	5	5	25	164	84	15	3,500	9.48	9.48	0	0	49	47	0:00:01	0:00:01
a 14	10	41	144	4,933	1,237	23	4,421	14.7	14.70	0	0	563	563	0:01:08	0:01:35
a 14	7	4	26	167	155	0	3,938	2.14	2.14	0	0	9	7	0:00:01	0:00:01
a 17	45	430	540	41,074	8,024	69	2,707	21.71	21.71	0	0	1,989	2,211	0:42:11	0:58:50
a 17	8	21	232	16,157	3,386	73	3,233	14.18	14.18	0	0	1,789	1,799	0:14:51	0:37:29
a 30	10	56	317	2,909	2,487	0	6,307	19.04	6.19	15.64	0	41	97	5:00:00	0:10:47
a 30	15	17	218	51,328	2,967	0	14,031	43.97	44.57	43.00	43.65	91	83	5:00:00	5:00:00
a 48	10	85	1,278	39,050	10,117	0	17,824	42.14	42.14	39.99	39.39	19	25	5:00:00	5:00:00
a 48	15	43	579	15,151	9,471	0	33,525	58.90	58.90	58.79	58.79	1	7	5:00:00	5:00:00
a 52	10	147	1,503	22,319	5,851	0	7,268	19.15	19.15	15.87	17.69	35	15	5:00:00	5:00:00
a 52	20	61	561	21,826	9,109	0	20,401	56.35	54.53	56.14	54.09	29	57	5:00:00	5:00:00

inequalities are generated simultaneously. For the instances that are not solved to optimality in run 1, the gaps Gap1 and Gap3 are quite close except for three instances, r-30-15, r-48-10, and r-48-15, for which Gap3 is higher than Gap1. This implies that separating all the inequalities simultaneously seems to be less effective than separating the inequalities in a predefined order. Notice that for one instance, r-58-20, the algorithm at run 3 has obtained a feasible solution while in run 1, the total CPU time has been spent at the root node of the Branch-and-Cut tree and no feasible solution has been found. This let us conclude that, as expected, run 3 may improve the resolution process for some instances. However, the number of nodes in the Branch-and-Cut tree is very small and the gap Gap3 is relatively high. Hence, we conclude that this improvement is not very significant.

For arbitrary demands (Table 8), the observations are the same. For the instances which are solved to optimality in run 1, the CPU time increases in run 3, and Gap3 is the same as Gap1 for all these instances except for instance a-30-10 where Gap3 equals 19.04% and Gap1 equals 6.19%. Moreover, for this latter instance, the algorithm obtains the optimal solution in run 1, while in run 3, only an upper bound is obtained.

All these observations lead to the conclusion that separating the inequalities simultaneously does not help to reduce the total CPU time used for solving the LPs in our Branch-and-Cut algorithm. It may even decrease the performance of the algorithm in several cases.

7. CONCLUSION

In this article, we have studied the Two Node-Disjoint Hop-Constrained Survivable Network Design Problem (TNHNDP) in the case where the hop constraint is $L = 3$. We have presented an integer programming formulation for the problem and studied the associated polytope. We then have described several classes of valid inequalities and investigate conditions under which they define facets of the polytope. We have also discussed the separation problem associated with the so-called natural inequalities, as well as the rooted partition and double cut inequalities, and devised a Branch-and-Cut algorithm to solve the problem.

The computational study realized in this article shows that the Branch-and-Cut algorithm is effective in solving the problem for several instances with both rooted and arbitrary sets of demands. It also shows that for large size instances, the problem is still difficult to solve. The experiments further point out that both rooted partition and double cut inequalities (3.11) and (3.12) are effective in solving the problem in general. Moreover, the rooted partition inequalities are more effective than the double cut inequalities for solving the rooted TNHNDP.

The theoretical and computational study conducted here also shows that solving large size instances by a Branch-and-Cut algorithm is still a challenging task and several issues need to be addressed for an efficient resolution of the problem. Among these, the reduction of the gap achieved between the root node of the Branch-and-Cut tree and the best upper bound may have a significant impact on the resolution of the problem, by decreasing the total CPU time as well as the number of nodes in the Branch-and-Cut tree. This can be addressed by investigating more deeply the polytope of the problem and finding more facet defining inequalities, and devising efficient separation algorithms for the concerned inequalities.

Another issue to be addressed is the large number of natural inequalities separated during the execution of the algorithm, and the CPU time spent by the algorithm in solving each LP. As previously mentioned, the excessive CPU time spent in solving each LP and separating the natural inequalities may prevent the Branch-and-Cut algorithm from a good exploration of the Branch-and-Cut tree and finding good feasible solutions. The different separation strategies tested in this study have not improved the total CPU time. Even the simultaneous separation of all the inequalities, sometimes used in the literature to reduce the total number of LPs that are solved, has failed in improving the total CPU time. For this latter point, one may investigate other methods, like decomposition methods, to efficiently solve each LP of the Branch-and-Cut algorithm.

It could also be interesting, for both theoretical and practical purposes, to investigate in some special cases the structural properties of the solutions of the TNHNDP. For example, one can consider that the edge weights satisfy

the triangle inequalities. In fact, many real networks have this property. One can also consider the problem in special classes of graphs like series-parallel or Halin graphs. This can lead to efficient primal heuristics for the problem in the above cases or even produce new valid inequalities for a Branch-and-Cut algorithm in general. These two points can also be used to reduce the gap and the total CPU time spent by a Branch-and-Cut algorithm in solving the problem.

Finally, the case where $L = 4$ should also be addressed since in practice, the QoS-constraint in telecommunication networks may be ensured by paths with more than 3 hops. To the best of our knowledge, few polyhedral results are known for that case even if the natural formulation (2.7) is valid for the TNHNDP with $L = 4$. More generally, there are few results on both TEHNDP and TNHNDP for $L \geq 4$. As an illustration, a natural formulation even for $L = 5$ for both TNHNDP and TEHNDP is still unknown in the literature. Thus, it would be interesting to study, from both theoretical and computational points of view, the TNHNDP (and the TEHNDP) for $L = 4$ in particular, and for $L \geq 4$ in general.

ACKNOWLEDGMENT

The authors would like to thank the anonymous referees for their valuable comments that considerably improved the presentation of the paper.

REFERENCES

[1] ABACUS—A Branch-And CUt System, webpage. Available at: http://www.informatik.uni-koeln.de/abacus/.

[2] R.K. Ahuja, T.L. Magnanti, and J.B. Orlin, Network flows: Theory, algorithms, and applications, Prentice Hall, New Jersey, 1993.

[3] F. Bendali, I. Diarrassouba, A.R. Mahjoub, and J. Mailfert, The k edge-disjoint 3-hop-constrained paths polytope, Discrete Optim 7 (2010), 222–233.

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

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

[6] I. Diarrassouba, Survivable network design problems with high connectivity requirements, PhD Thesis, Université Blaise Pascal, France, 2009.

[7] I. Diarrassouba, V. Gabrel, L. Gouveia, A.R. Mahjoub, and P. Pesneau, Integer programming formulations for the k-edge-connected 3-hop-constrained network design problem, Networks 67 (2016), 148–169.

[8] I. Diarrassouba, H. Kutucu, and A.R. Mahjoub, Two node-disjoint hop-constrained survivable network design and polyhedra, Technical Report, Cahier du LAMSADE N^o 332, Université Paris Dauphine, 2013.

[9] E.W. Dijkstra, A note on two problems in connexion with graphs, Numer Math 1 (1959), 269–271.

[10] J. Edmonds and R.M. Karp, Theoretical improvements in algorithmic efficiency for network flow problems, J ACM 19 (1972), 248–264.

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

[12] A. Goldberg and R.E. Tarjan, A new approach to the maximum-flow problem, J ACM 35 (1988), 921–940.

[13] 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, G. Anandalingam and S. Raghavan (Editors), Springer, Berlin, 2005, Vol. 33, pp. 167–180.

[14] D. Huygens and A.R. Mahjoub, Integer programming formulations for the two 4-hop-constrained paths problem, Networks 49 (2007), 135–144.

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

[16] TSPLIB, webpage. Available at: http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/.